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

Automatische Erkennung von destruktiven Teaminteraktionen aus Audiosignalen

Automatic Recognition of Destructive Team Interactions from Audio Signals

Bachelorarbeit

im Studiengang Informatik

von

Cedric Fauth

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

Hannover, 20.01.2022

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 20.01.2022

Cedric Fauth

Zusammenfassung

Automatische Erkennung von destruktiven Teaminteraktionen aus Audiosignalen

In einem Softwareprojekt spielt die Zusammenarbeit des Teams eine essentielle Rolle für dessen Erfolg. Teaminteraktionen können dazu beitragen den Teamgeist zu stärkten, jedoch verläuft nicht jede Teamkommunikation reibungslos. *Destruktive* Interaktionen während Teammeetings sind keine Seltenheit und können mithilfe eines Kodierungsschemas wie *act4teams* oder *act4teams-SHORT* analysiert werden. Durch diese Analysen ist es möglich, den Teammitgliedern Feedback zu geben, damit diese ihr Verhalten reflektieren können. Jedoch ist die Analyse eines Meetings sehr zeitaufwendig und erfordert oftmals den Einsatz geschulter Psychologen.
In dieser Arbeit werden verschiedene Klassifikatoren getestet, um automatisch *destruktive* Interaktionen anhand von Audiosignalen zu erkennen und genauer einzuordnen. Der verwendete Datensatz besteht aus Audioaufzeichnungen von Softwareprojekten. Aus diesen Audiospuren werden aussagekräftige Merkmale ausgewählt und extrahiert. Danach erfolgt eine binäre Klassifizierung zwischen *destruktiven* und *nicht-destruktiven* Interaktionen. Im zweiten Schritt erfolgt eine detailliertere Multi-Class Klassifizierung der bereits erkannten *destruktiven* Interaktionen. Die sorgfältig ausgewählten Klassifikatoren beinhalten sowohl klassische Methoden wie SVMs, Logistic Regression, Random Forests und Naive Bayes, aber auch rekurrente neuronale Netze. Diese werden anhand von Metriken wie Precision, Recall und Balanced Accuracy auf Testdaten ausgewertet. Es wird gezeigt, dass die SVM sowohl in der binären, als auch Multi-Class Klassifizierung die besten Ergebnisse mit einer Balanced Accuracy von 66% respektive 36% erzielt.

Abstract

Automatic Recognition of Destructive Team Interactions from Audio Signals

In a software project, team collaboration plays an essential role in its success. Team interactions can help to strengthen the team spirit, but not every team communication runs smoothly. *Destructive* interactions during team meetings are not uncommon and can be analyzed using a coding scheme such as *act4teams* or *act4teams-SHORT*. Through these analyses, it is possible to provide feedback to team members so that they can reflect on their behavior. However, analyzing a meeting is very time consuming and often requires the use of trained psychologists.
In this work, different classifiers are tested to automatically identify *destructive* interactions based on audio signals and classify them more accurately. The dataset used consists of audio recordings of software projects. Meaningful features are selected and extracted from these audio tracks. Then, a binary classification between *destructive* and *non-destructive* interactions is performed. In the second step, a more detailed multi-class classification of the already detected *destructive* interactions is performed. The carefully selected classifiers include classical methods such as SVMs, Logistic Regression, Random Forests, and Naive Bayes, but also recurrent neural networks (RNNs). These are evaluated on test data using metrics such as Precision, Recall, and Balanced Accuracy. It is shown that SVM performs best in both binary and multi-class classification with a Balanced Accuracy of 66% and 36% respectively.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Das Gebiet der praktischen Informatik bildet den Übergang von theoretischer zu angewandter Informatik. Das Ziel der praktischen Informatik ist das Lösen von Problemen durch Anwendung theoretischer Methoden. Die Softwaretechnik ist ein Teilgebiet der praktischen Informatik und beschäftigt sich mit der Bereitstellung von Methoden und Techniken, welche für die erfolgreiche Entwicklung von Software benötigt werden [43, 55].

Heutzutage ist die Softwareentwicklung eine komplexe Aufgabe, welche durch Industrie und Wissenschaft vorangetrieben wird. Die hohen Kundenanforderungen, der Personalaufwand und die Kommunikation mit dem Kunden sowie im Unternehmen erhöhen die Komplexität der Entwicklung und des Managements. Aus diesem Grund werden Konzepte und Methoden entworfen, welche die erfolgreiche Entwicklung von Software strukturieren und absichern sollen. Beispielsweise wird der Entwicklungsprozess in mehrere Teile aufgeteilt. Angefangen mit der Anforderungsanalyse versuchen die Entwickler alle Anforderungen und Visionen des Kunden festzuhalten und in ein Modell zu übersetzen. Danach erfolgt die Umsetzung, welche kontinuierlich überprüft werden muss, um Probleme frühzeitig zu erkennen. Dabei helfen Methoden wie Scrum die Prozesse für alle Mitarbeiter unkompliziert und agil zu gestalten. Besonders das Arbeiten im Team ist aus der heutigen Softwareentwicklung nicht mehr wegzudenken. Teamarbeit beschleunigt den Entwicklungsprozess und ist für die meisten Projekte notwendig. Durch regelmäßige Meetings können die Teammitglieder ihre Fortschritte teilen, Probleme ansprechen und diskutieren. [55, 60].

Wissenschaftler beschäftigen sich seit einigen Jahren intensiv mit der Analyse von Teammeetings, da diese einen erheblichen Teil der Softwareentwicklung ausmachen [38, 60]. Dabei tritt immer mehr das Verhalten der einzelnen Teilnehmer in den Fokus. Kauffeld und Lehmann-Willenbrock [34] sowie Schneider et al. [60] fanden heraus, dass die Interaktionen und das Verhalten der Mitglieder die Stimmung des Teams beeinflussen können. Des Weiteren konnte ein Zusammenhang zwischen der Teamstimmung und dem Erfolg des

Projekts beobachtet werden.

1.1 Problemstellung

Teammeetings stellen eine sehr direkte und schnelle Art der Kommunikation innerhalb eines Softwareteams dar. Aus diesem Grund werden wichtige Entscheidungen und Probleme während eines Meetings besprochen. Jedoch verlaufen Meetings nicht immer wie geplant. Unstimmigkeiten und *destruktives* Verhalten zwischen den Teammitgliedern verursachen Spannungen und wirken sich negativ auf die Teamstimmung und letzendlich auf den Entwicklungsprozess aus [60]. Dies ist einer der Gründe, weshalb die Analyse von Meetings in den letzten Jahren zunehmend an Bedeutung gewann. Unter anderem sollen die Teilnehmer nach dem Meeting ein Feedback über ihr Verhalten und die Teamstimmung erhalten, wodurch diese besser reflektieren und Probleme im Team schneller erkennen können [60]. Kauffeld und Lehmann-Willenbrock [34] entwickelten das act4teams Schema, welches ein Modell für die Kategorisierung von Aussagen während eines Meetings beschreibt. Die Analyse eines einstündigen Videos nach dem act4teams Schema von ausgebildeten Psychologen dauert etwa zehn Stunden. Daraufhin vereinfachten Klünder et al. [38] das Schema und veröffentlichten act4teams-SHORT, welches weniger Kategorien enthält und für die Echtzeitanalyse vorgesehen ist.
Dennoch muss in beiden Fällen eine manuelle Analyse des Meetings durch eingewiesene Beobachter oder Experten erfolgen. Das ist ein aufwendiger Prozess, bei dem hohe Konzentration und viel Zeit erforderlich sind. Besonders in Unternehmen ist es eine Herausforderung diese Prozesse umzusetzen, weil entsprechende Fachkräfte benötigt werden.

1.2 Lösungsansatz

Diese Arbeit ist dem übergeordneten Themengebiet der automatischen Erkennung und Analyse von Aussagen in Meetings einzuordnen. Ziel der Arbeit ist es, auf der Grundlage von Audiospuren, *destruktive* Interaktionen innerhalb eines Meetings automatisch zu klassifizieren. Der Fokus liegt dabei nur auf *destruktiven* Interaktionen, welche sich negativ auf Team und Projekt auswirken können [34, 60].
Es werden verschiedene Klassifikationsverfahren im Bereich des maschinellen Lernens ausgewählt und getestet. Darunter fallen sowohl klassische Verfahren, als auch neuronale Netze. Zudem sollen erkannte *destruktive* Interaktionen in Unterkategorien eingeordnet werden. Die zugrundeliegenden Audioaufnahmen stammen aus Softwareprojekt-Meetings und werden mit quantitativen Methoden analysiert. Es soll geprüft werden, ob quantitative Verfahren in diesem Kontext gute Ergebnisse liefern können.

Am Ende wird eine Software in Form eines Prototyps erstellt, welche Audioausschnitte mithilfe der in dieser Arbeit trainierten Klassifizierer auswerten kann. Die Aufnahmen müssen bereits segmentiert sein, sodass diese jeweils eine Interaktion enthalten. Die Analyse erfolgt automatisch ohne menschlichen Eingriff oder zusätzliche Ressourcen. Dieser Lösungsansatz soll den Analyseaufwand von Meetings erleichtern. Die hier erarbeiteten Konzepte können zukünftig mit anderen Methoden wie Natural Language Processing kombiniert werden, um bessere Endergebnisse zu erhalten.

1.3 Struktur der Arbeit

In Kapitel 2 werden wichtige Grundlagen vermittelt, worauf weitere Konzepte und Methoden dieser Arbeit aufbauen. Es werden grundlegende Definitionen eingeführt und erklärt sowie Evaluationsmetriken genannt. Es folgt ein Vergleich und eine Abgrenzung von verwandten Arbeiten in Kapitel 3. Kapitel 4 behandelt die Vorverarbeitung und Auswahl von Merkmalen der Audiospuren. Die Klassifikatoren teilen sich in zwei Gruppen, welche unterschiedlich trainiert werden. In Kapitel 5 wird auf die Auswahl und das Training verschiedener klassischer Klassifikatoren eingegangen. Kapitel 6 behandelt Auswahl und Training rekurrenter neuronaler Netze. Anschließend werden die unterschiedlichen Klassifikatoren in Kapitel 7 ausgewertet und verglichen. Zudem wird auf die Erstellung des Prototyps eingegangen. Kapitel 8 diskutiert die Ergebnisse und zeigt Ursachen für Probleme auf. Letztendlich fasst Kapitel 9 die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf zukünftige Aufgaben.

Kapitel 2

Grundlagen

2.1 Act4teams und act4teams-SHORT

Advanced Interaction Analysis for teams (act4teams) ist ein Kodierungsschema, welches von Kauffeld und Lehmann-Willenbrock [34] zur Analyse und Einordnung von Aussagen in Teammeetings entwickelt wurde. Die Aussagen der Teammitgleider können von qualifizierten Personen wie Psychologen jeweils einem der 44 act4teams Codes zugeordnet werden. Die verschiedenen Codes sind in vier Kompetenzbereiche mit jeweiligen Unterkategorien aufgeteilt. Diese Bereiche lauten Professionelle Kompetenz, Methodenkompetenz, Sozialkompetenz und Selbstkompetenz. Mithilfe des Kodierungsschemas lassen sich einzelne Aussagen genauer auf ihre Auswirkung auf das Meeting analysieren [34].
Grundsätzlich existieren unterschiedliche Arten von Aussagen, welche auch unterschiedliche Reaktionen und Stimmungen im Team hervorrufen können. Das Schema erwies sich als hilfreich, um diese zu identifizieren und den Teamerfolg durch Feedback verbessern zu können. Schneider et al. [60] konnten bereits zeigen, dass sich proaktives Verhalten gefolgt von unterstützenden Aussagen positiv auf die Teamstimmung auswirkt und auch einen positiven Einfluss auf das Projekt hat. Allerdings muss die Analyse eines Meetings durch qualifizierte Mitarbeiter erfolgen. Ein Psychologe benötigt für die Analyse einer einstündigen Video-/Audioaufnahme etwa zehn Stunden mithilfe des act4teams Schemas [60]. Grund dafür ist vor allem die Vielzahl an act4teams Codes, welche von einer unausgebildeten Personen nicht einfach voneinander unterscheidbar sind [37].
Daraufhin entwickelte Klünder [37] das vereinfachte Kodierungsschema act4teams-SHORT, welches für die Analyse von Meetings in Echtzeit vorgesehen ist. Das Schema reduziert die 44 act4teams Codes auf 11 Kategorien, welche verständlicher und einfacher zu unterscheiden sind. Dadurch ist es auch für Mitarbeiter ohne spezielle Ausbildung möglich, nach einer kurzen Einweisung eine Meetinganalyse in Echtzeit durchzuführen.

2.2 Destruktive Interaktionen

In dieser Arbeit sollen Verfahren angewendet werden, welche automatisch zwischen *destruktiven* und *nicht-destruktiven* Interaktionen unterscheiden können. Ebenfalls ist es ein Ziel, *destruktive* Interaktionen genauer zu analysieren. Interaktionen sind definiert als aufeinander bezogene Handlungen mehrerer Personen [7]. Diese Arbeit bezieht sich hierbei nur auf verbale Interaktionen im Kontext von Entwicklermeetings. Die Definition *destruktiver* Interaktionen innerhalb dieser Arbeit basiert auf den Kategorien des act4teams und act4teams-SHORT Kodierungsschema [34, 37, 38]. Klünder [37] definiert in act4teams-SHORT die Kategorie destruktives Verhalten als Vereinigung der act4teams Codes **Tadeln/Lästern (TD)**, **Schuldigensuche (S)**, **Jammern (J)**, **Kein Interesse an Veränderungen (KI)** und **Seitengespräche (Seit)**. Damit bezieht sie sich auf Codes innerhalb der act4teams Kompetenzbereiche Sozial- und Selbstkompetenz mit ihren Unterkategorien unkollegiale Interaktion und destruktive Mitwirkung. Allerdings wählte Klünder nur fünf der insgesamt zehn Codes innerhalb dieser Unterkategorien für ihre Definition von destruktivem Verhalten aus. Bei den fünf restlichen Codes **Unterbrechung (Unt)**, **Reputation (R)**, **Abbruch (E)**, **Betonung autoritärer Elemente (AE)** und **Phrase (AL)** kann nicht ausgeschlossen werden, dass es sich um destruktives Verhalten handelt. Allerdings war die durchschnittliche Dauer von Aussagen zu diesen Codes in einem Beispieldatensatz so kurz, dass es Beobachtern zu schwer fallen würde, diese Aussagen in Echtzeit zu identifizieren. Aus diesem Grund wird in act4teams-SHORT nur eine reduzierte Anzahl an Codes für destruktives Verhalten genutzt [37].
Anders als bei act4teams-SHORT werden Interaktionen in dieser Arbeit von Computerprogrammen klassifiziert. Es existieren bereits fertige Audioaufzeichnungen als Eingabedaten. Damit sind menschliche Limitierungen während der Echtzeitanalyse wie das Kurzzeitgedächtnis hinfällig. Aus diesem Grund werden *destruktive* Interaktionen als Vereinigung aller Codes aus den Kategorien unkollegiale Interaktion und destruktive Mitwirkung definiert. *Destruktive* Interaktionen sind also die Menge der zehn folgenden act4teams Codes: **{ AE, AL, E, J, KI, R, S, Seit, TD, Unt }** . Eine Zusammenfassung der aufgeführten Codes befindet sich in Anhang A.1.

2.3 Undersampling

Datensätze werden benötigt um ein Training mittels Klassifikatoren vorzunehmen. In der Praxis kommt es häufig vor, dass Datensätze eine ungleiche Klassenverteilung besitzen. Zum Beispiel enthält der in dieser Arbeit verwendete Datensatz viel weniger *destruktive* als *nicht-destruktive* Interaktionen. Der Ursprung dieses Ungleichgewichts liegt in der Realität:

Nur ein geringer Teil aller Interaktionen innerhalb eines Meetings sind *destruktiv*, da es sich um sozio-emotionale Aussagen handelt. Der Großteil aller Aussagen bezieht sich auf konkrete Projektinhalte und ist damit meistens dem nicht-destruktivem Verhalten zuzuordnen [34, 38]. Aus diesen Gründen ist das Verhältnis der Interaktionen im Originaldatensatz nicht ausgewogen.

Undersampling ist ein Verfahren zum Ausgleich von unausgewogener Klassenverteilung in einem Datensatz. Es entfernt Daten der überlegenden Klasse und gleicht somit das Klassenverhältnis aus. Ein Nachteil ist der entstehende Informationsverlust durch das Löschen der Daten [44, 45]. In dieser Arbeit wird an einigen Stellen der *RandomUndersampler* aus der Python Bibliothek Imbalanced-Learn [44] genutzt. Das Auswählen der zu entfernenden Daten geschieht zufällig. Die Samplingrate gibt das Verhältnis zur Minderheitsklasse an. Es wird ein Sampling von 1,0 genutzt, damit ein ausgeglichenes Klassenverhältnis von 50:50 entsteht.

2.4 Kreuzvalidierung

Kreuzvalidierung (engl. Cross Validation; kurz: CV) ist ein Verfahren zur Bewertung von maschinellen Lernmethoden. „Klassischerweise“ wird ein Datensatz in Trainings- und Testmenge aufgeteilt (Train-Test-Split), um mit der Trainingsmenge einen Klassifikator zu trainieren und mit der Testmenge den Klassifikator zu bewerten. Die Testmenge wird während des Trainings nicht genutzt und enthält ausschließlich unbekannte Daten für den Klassifikator. Dadurch ist es möglich, das Verhalten des Modells auf reale und neue Daten zu testen. Außerdem kann durch den Vergleich von Metriken zwischen dem Trainings- und Testdatensatz herausgefunden werden, ob der Klassifizierer eher zum Auswendiglernen der Trainingsdaten neigt, oder passend generalisieren kann [57].

Das Problem beim Train-Test-Split ist, dass der Testdatensatz nur eine kleine Teilmenge aller Daten enthält, da die meisten Daten für das Training genutzt werden. Die Evaluation eines Klassifizierers wäre aussagekräftiger, wenn er mindestens einmal auf allen Daten getestet werden könnte. Kreuzvalidierung ist ein Verfahren, welches diesen Lösungsansatz verfolgt. Der Datensatz wird in k Teilmengen aufgeteilt. Anschließend wird eine der Teilmengen als Testdatensatz und die restlichen Teilmengen als Trainingsdatensatz genutzt. Die Klassifizierer werden anhand dieser Daten trainiert und getestet. Der Ablauf wiederholt sich k-Mal bis jede Menge einmal als Testmenge vorkam. Aus den k Ergebnissen wird ein Mittelwert gebildet. Das Ergebnis ist genauer, da die Evaluation mit mehreren Datensätzen erfolgt und alle Daten einmal im Testdatensatz vorkommen. Dieses Verfahren wird auch als k-Fold-Cross-Validation bezeichnet [40, 57, 58].

2.4.1 Stratified-k-Fold-Cross-Validation

Stratified-k-Fold-Cross-Validation erweitert das oben beschriebene Verfahren der k-Fold-Cross-Validation um eine zusätzliche Bedingung. Bei der normalen Kreuzvalidierung wird der Datensatz in k zufällige Teilmengen unterteilt. Diese enthalten auch eine zufällige Verteilung von Daten. Bei ausgewogenen Datensätzen, in denen alle zu klassifizierenden Klassen im gleichen Verhältnis vorkommen, stellt die zufällige Unterteilung in Teilmengen kein Problem dar. Allerdings werden in dieser Arbeit auch Datensätze mit ungleicher Klassenverteilung verwendet. Dadurch kann es vorkommen, dass einige Teilmengen zu viele beziehungsweise zu wenige Daten einer Klasse beinhalten und die Ergebnisse verfälscht werden. Die stratifizierte Kreuzvalidierung stellt sicher, dass das Klassenverhältnis auch innerhalb der Teilmengen ungefähr gleich bleibt und somit aussagekräftige Ergebnisse erzielt werden können [40].

2.4.2 Grid-Search-Cross-Validation

Grid-Search-Cross-Validation (GridSearchCV) ist ein Verfahren zum Anpassen und Optimieren von Hyperparametern eines Klassifizierers. Viele Klassifizierer besitzen Parameter, welche vor dem Training gewählt werden müssen und später nicht mehr änderbar sind. Diese Parameter heißen Hyperparamter. Um Hyperparamter zu optimieren, kann GridSearchCV genutzt werden. Bei diesem Verfahren wird im Voraus für jeden ausgewählten Hyperparamter eine Menge der zu testenden Werte angegeben. Anschließend startet eine Kreuzvalidierung mit jeder Kombination von Hyperparamterwerten (Kreuzprodukt der Parametermengen) [9]. Am Ende erfolgt die Rückgabe der Parameterkombination, die zum besten Ergebnis führte. Die Definition des besten Ergebnisses muss vorher mithilfe einer beliebigen Gütefunktion erfolgen. In dieser Arbeit wird der Balanced-Accuracy Score verwendet.

2.5 Rekurrente neuronale Netze

Die Forschung und Nutzung neuronaler Netze hat in den letzten Jahren vor allem durch ihre vielseitige Einsetzbarkeit und die steigende Rechenleistung von Grafikkarten stark zugenommen [63]. Ein neuronales Netz besteht mindestens aus Eingabe- und Ausgabeschicht. Diese Schichten enthalten ein oder mehrere Neuronen. Neuronen sind Einheiten, welche Eingaben verarbeiten und eine Ausgabe berechnen. Bei einem Feed-Forward Netzwerk sind benachbarte Schichten in einer Richtung miteinander verbunden. Diese Netze haben die Limitierung, dass es keine Verbindung zwischen Neuronen innerhalb einer Schicht gibt. Somit hängt die Ausgabe eines Neurons nur von den Ausgaben der verbunden Neuronen der höheren Schicht und den individuellen Gewichten des Neurons selbst zusammen. In der

Sprachanalyse zeigen Feed-Forward Netze Schwächen, weil es sich um voneinander abhängige Sequenzen von Informationen handelt [63]. Zum Beispiel ist das nächste Wort eines Satzes von den vorherigen Informationen abhängig. So ist es wahrscheinlich, dass aufgrund grammatikalischer Regeln nach einem Nomen ein Verb folgt. Feed-Forward Netze sind nicht dafür geeignet, sequenzielle Zusammenhänge zu erkennen.

Rekurrente neuronale Netze (RNNs) versuchen genau diesem Problem entgegenzuwirken. In diesen Netze können Neuronen auch innerhalb einer Schicht oder mit vorherigen Schichten verbunden sein. Durch diese Rückverbindungen entsteht eine Art Kurzzeitgedächtnis, wodurch sequenzielle Informationen besser verarbeitet werden können [20, 63]. Die Ausgabe eines Neurons hängt nicht nur von den Daten des aktuellen Zeitschritts ab, sondern auch vom Ergebnis des benachbarten Neurons, welches den vorherigen Zeitschritt verarbeitet hat. Jedoch besitzen gewöhnliche RNNs Schwachstellen. So ist es beispielsweise schwierig Informationen über mehrere Zeitschritte hinweg zu speichern bis sie benötigt werden [63].

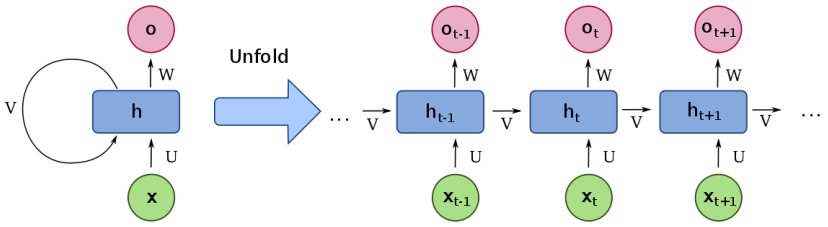


Abbildung 2.1: Kompakte (links) und ausführliche (rechts) Darstellung einer einfachen rekurrenten Schicht [32].

2.5.1 Long Short-Term Memory

Ein Long Short-Term Memory (LSTM) Netzwerk erweitert die Funktionalität eines RNN um eine Art Langzeitgedächtnis, welches sich Informationen vergangener Schritte merken kann. Die Zellen dieses Netzwerks sind deutlich komplexer aufgebaut und enthalten verschiedene Funktionen zur Speicherung und Verarbeitung von Informationen [20, 63].

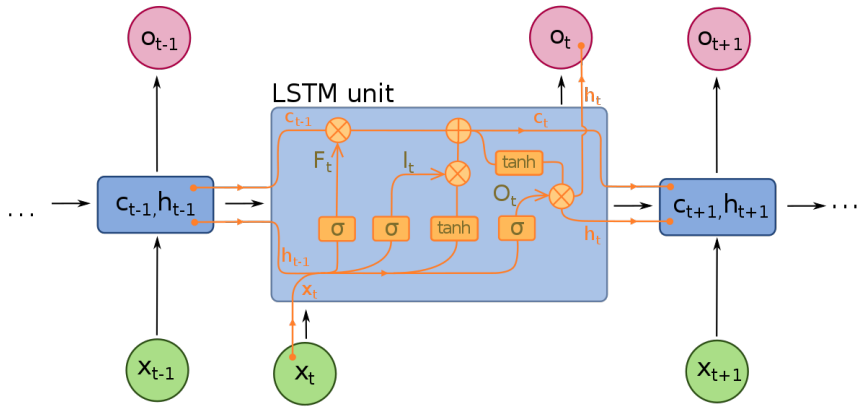


Abbildung 2.2: Aufbauschema eines LSTM Neurons [31].

2.5.2 Gated Recurrent Units

Ein Gated Recurrent Unit (GRU) besitzt ähnliche Eigenschaften wie ein LSTM. Diese Einheiten können ebenfalls Langzeitinformationen verarbeiten. Der Unterschied besteht im internen Aufbau der Zellen. GRUs sind eine relativ neue Entwicklung und bauen auf LSTMs auf, besitzen allerdings weniger Parameter [20, 63]. Sie sollen durch den vereinfachten Aufbau effizienter sein. Tests ergaben, dass sich sowohl LSTMs als auch GRUs zur Musik und Spracherkennung eignen [63].

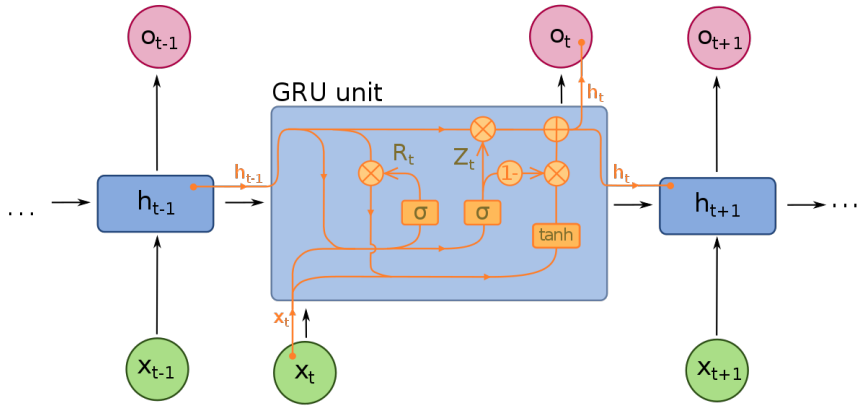


Abbildung 2.3: Aufbauschema eines GRU Neurons [30].

2.6 Evaluationsmetriken

In der Evaluation in Kapitel 7 werden verschiedene Klassifikatoren mithilfe geeigneter Metriken ausgewertet und verglichen. Die Metriken werden zuvor in diesem Kapitel erläutert.

Konfusionsmatrix

Die einfachste Form der Klassifizierung ist die binäre Klassifizierung, bei der nur zwischen zwei Klassen unterschieden wird. In dieser Arbeit sollen beispielsweise *destruktive* Interaktionen erkannt werden, deshalb sind diese als sogenannte *positive Klasse* definiert. *Nicht-destruktiven* Interaktionen zählen zur *negativen Klasse*. Ein Datum besitzt also eine von zwei Klassen und wird von einem Klassifizierer in eine der zwei Klassen eingeordnet, somit müssen vier verschiedene Fälle unterschieden werden. Anhand dieses Beispiels werden die Begriffe True-/False Positive und True-/False Negative beschrieben.

1. True Positives (TP) sind Daten, welche der positiven Klasse angehören und korrekt zugeordnet wurden.
2. False Positives (FP) sind Daten, welche der negativen Klasse angehören und fälschlicherweise der positiven Klasse zugeordnet wurden.
3. True Negatives (TN) gehören der negativen Klasse an und werden korrekt klassifiziert.
4. False Negatives (FN) gehören der positiven Klasse an und werden als negativ klassifiziert.

Die vier möglichen Kategorien können in einer Konfusionsmatrix dargestellt werden, wie in Anhang A.1 zu sehen ist [16, 65]. Aus dieser Matrix kann abgelesen werden, wie die klassifizierten Daten eingeordnet wurden. Jede Reihe stellt Daten einer echten Klasse dar. Die Spalten stehen für die vorhergesagten Klasse. Meistens werden die Felder in Prozenten dargestellt, wobei die Summe innerhalb einer Reihe 100% ergibt (alle Daten der echten Klasse). Die Konfusionsmatrix einer perfekten Klassifizierung enthält nur Einträge auf der Hauptdiagonalen.

Recall

Der Recall wird auch True-Positive Rate genannt und gibt das Verhältnis zwischen der Anzahl von True Positives und der Anzahl aller tatsächlich positiven Daten (TP+FN) an. Das Verhältnis sagt aus, wie viele der relevanten Daten tatsächlich erkannt werden. Ein Recall von 0,5 bedeutet, dass lediglich 50% der positiven Daten als positiv erkannt werden [16, 65].

$$Recall = \frac{TP}{TP + FN}$$

Precision

Precision ist das Verhältnis zwischen True Positives und allen positiv klassifizierten Daten (TP+FP). Eine Precision von 50% bedeutet, dass

ein positiv klassifiziertes Datum mit einer Wahrscheinlichkeit von 50% tatsächlich der Positiven Klasse angehört [16, 65].

$$Precision = \frac{TP}{TP + FP}$$

F1

Der F1 Wert bildet den harmonischen Durschnitt von Precision und Recall. Der Maximalwert ist 1,0, wenn sowohl Recall als auch Precision 1,0 sind. Ist eine der beiden Werte 0,0, erreicht auch F1 seinen Minimalwert von 0,0 [65].

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

False-Positive Rate

Die False-Positive Rate ist das Verhältnis von False-Positives zu allen Daten der negativen Klasse (TN+FP). Liegt dieser Wert beispielsweise bei 0,5, werden 50% der negativen Daten als positiv klassifiziert und sich damit falsch-positiv [16, 65].

$$FalsePositiveRate = \frac{FP}{TN + FP}$$

Balanced Accuracy

Balanced Accuracy bildet den Mittelwert der True-Positive Rate (Recall) und True-Negative Rate. Die True-Negative Rate gibt dabei den Anteil der True-Negatives von allen Daten der negativen Klasse an. Sie ist somit der *Recall der negativen Klasse*. Für einen Datensatz mit einem 50:50 Klassenverhältnis ist die Balanced Accuracy identisch mit der normalen Accuracy (TP+TN geteilt durch die Anzahl aller Daten). Jedoch werden bei einem unausgewogenen Klassenverhältnis alle Klassen behandelt, als besäßen sie das gleiche Verhältnis [5, 27].

$$BalancedAccuracy = \frac{TPR + TNR}{2}$$

Diese Metrik lässt sich gut auf unausgewogene Datensätze anwenden, bei denen die normale Accuracy nicht aussagekräftig ist. Wenn beispielsweise 90% der Daten der negativen Klasse angehören und ein Klassifizierer alle Daten als negativ einordnet, ist die Accuracy 90%. Dieser Wert sagt aber nichts über die Qualität des Klassifizierers aus, da die True-Positive Rate bei 0% liegt. Der Balanced Accuracy Wert für diese Klassifizierung liegt bei 50%, da alle Daten der negativen Klasse und keine Daten der positiven Klasse richtig eingeordnet wurden [5, 8].

Kapitel 3

Verwandte Arbeiten

In diesem Kapitel werden weitere Arbeiten mit verwandten oder ähnlichen Themen vorgestellt. Am Ende des Kapitels wird noch einmal auf die Abgrenzung dieser Arbeit zu den verwandten Arbeiten eingegangen.

Kauffeld und Lehmann-Willenbrock [34] konnten mithilfe des act4teams Kodierungsschemas die Kommunikation in 92 Teammeetings manuell analysieren. Dabei beobachteten sie, dass funktionale Interaktionen innerhalb der Teams mit guter Teamarbeit und hoher Produktivität zusammenhängen. Dysfunktionale Kommuniktation resultierte hingegen in schlechterer Zusammenarbeit und beeinflusste den Erfolg der Projekts. Dabei wurden jedoch nur Meetings von Produktionsteams betrachtet.

Die Studie von Cole et al. [12] kam zu dem Ergebnis, dass dysfunktionales Verhalten in Teams die Stimmung des gesamten Teams beeinflusst. Des Weiteren steht dies in einem komplexen Zusammenhang mit der Teamleistung. Schneider et al. [60] nutzten diese Erkenntnisse und analysierten, wie sich positives Verhalten auf die Teamstimmung auswirkt. Es konnte beobachtet werden, dass proaktive Aussagen, gefolgt von unterstützenden Aussagen, einen positiven Einfluss auf den *Group Affect* haben. Diese Ergebnisse bestätigen die Wichtigkeit funktionierender Teams und somit auch die Analyse von Teammeetings, um Probleme frühzeitig zu erkennen.

Klünder et al. [38] analysierten Interaktionen in Meetings im Bereich der Softwareentwicklung. Dazu diente das von act4teams abgeleitete Kodierungsschema act4teams-SHORT, um den Analyseprozess zu vereinfachen. Es konnten Probleme der Teams wie fehlende Projektstruktur und Zielsetzung identifiziert werden. Außerdem stellten sie in einigen Meetings destruktives Verhalten fest, welches das Vertrauen der Mitglieder untereinander beeinflussen kann.

Die bisher genannten Arbeiten beschäftigen sich hauptsächlich mit den grundlegenden Konzepten und Auswirkungen von unterschiedlichen Verhaltensweisen in Teammeetings. Zusätzlich konnten Modelle entwickelt werden, welche die Analyse der Meetings vereinfachen. Die folgenden Arbeiten gehen auf Konzepte und Umsetzungen automatischer Analyseverfahren ein. Im Vordergrund steht die Erkennung und Klassifizierung von Äußerungen, Emotionen und Sentimenten anhand von Audiodaten.

Ein von Herrmann [26] entwickeltes Programm ist dazu in der Lage, Mikrofonaufnahmen aus Meetings zu transkribieren und automatisch, mithilfe weiterer Analyseverfahren von Horstmann [28] und Meyer [48], in eine von drei Polaritätsklassen (*negativ, neutral, positiv*) einzuordnen.
Obeidi [49] verfolgte ebenfalls das Ziel, Aussagen automatisch zu klassifizieren. Sie entwickelte ein Konzept, welches transkribierte Aussagen in destruktive und nicht-destruktive Äußerungen unter Verwendung der act4teams-SHORT Kategorien einordnet.
In dieser Arbeit wird zwischen *destruktiven* und *nicht-destruktiven* Interaktionen klassifiziert. Die Definition unterscheidet sich von der Definition destruktiver Aussagen von Obeidi [49]. Die hier genutzte Definition beruht sowohl auf dem act4teams als auch act4teams-SHORT Schema und umfasst zehn act4teams Codes.

Hung und Gatica-Perez [29] untersuchten unterschiedliche Features im Zusammenhang auf den Zusammenhalt von Teams. Aus Audio- und Videomaterial von Meetings wurden verschiedene Features extrahiert und teilweise kombiniert. Anschließend erfolgte ein Training von Support Vector Machines mit jeweils einem Feature, sodass die Features separat evaluiert werden konnten. Besonders mit den Merkmalen, welche sich auf den Redefluss beziehen, konnte der Zusammenhalt von Teams erfolgreich identifiziert werden.

In der Arbeit von Pan et al. [50] werden unterschiedliche Kombinationen von Audiomerkmalen für die Emotionsalayse getestet. Für die Klassifizierung der Emotionen *fröhlich, traurig* und *neutral* werden Support Vector Machines genutzt. Die Kombination der Features MFCC, MEDC und Energy führten zu einer Erkennugsrate von bis zu 95,1%. Aufbauend auf diesen Ergebnissen versuchten Ghai et al. [22] weitere Emotionen zu erkennen. Sie nutzen ebenfalls MFCC und Energie als Audiofeatures für die Erkennung von sechs Emotionen. Als Klassifizierer kamen SVMs, Gradient Boosting und Random Forests zu Einsatz. Random Forests erreichten mit 81,05% die höchste Genauigkeit.

Ramadhan et al. [56] nutzten textbasierte Merkmale, um eine Sentimentanalyse verschiedener Aussagen auf Social Media Plattformen durchzuführen. Dieser Ansatz nutzt den Logistic Regression Klassifikator, um zwischen den Kategorien *positiv* und *negativ* zu unterschieden. Die Genauigkeit belief sich auch 74%.

Eine Sentimentanalyse mithilfe von rekurrenten neuronalen Netzen führten Poria et al. [53] erfolgreich durch. Es wurden Merkmale aus Text-, Video- und Audiodaten kombiniert, um eine Klassifizierung zwischen den Sentimenten *negativ* und *positiv* umzusetzen. Audiofeatures wie MFCC, Intensität und Tonhöhe wurden genutzt. Die Text- und Videomerkmale konnten mithilfe von CNNs (Convolutional Neural Networks) extrahiert werden. Anschließend führte eine LSTM Architektur die eigentliche Sentimenterkennung durch. Kim und Lee [35] entwickelten diesen Ansatz weiter und nutzten zusätzliche Attention Layer in den neuronalen Netzen, welche kontextbasiert Merkmale verstärken beziehungsweise abschwächen können. Anstatt einfacher LSTM Schichten kamen bidirektionale GRU Schichten zum Einsatz.

Die hier vorgestellten Arbeiten beziehen sich auf ähnliche Themen oder Ansätze wie diese Arbeit. Allerdings können entscheidende Unterschiede zu den anderen Arbeiten festgestellt werden. Diese Arbeit definiert *destruktive* Interaktionen anhand der act4teams und act4teams-SHORT Schemata. Der genutzte Datensatz, wurde bisher noch nicht in dieser Form für eine Analyse eingesetzt. Die verschiedenen Features werden direkt aus den Audiosignalen extrahiert. Es erfolgt keine Transkribierung der Audiodateien, da die Analyse auf qualitative Verfahren setzt. Die genutzten Features werden auch in anderen Arbeiten mit Bezug auf Sentiment und Emotionsanalyse eingesetzt, jedoch nicht im Zusammenhang mit den act4teams Kategorien. Die genutzten Klassifikatoren finden in ähnlichen Arbeiten Anwendung, allerdings wird dadurch die Auswahl und Eignung der Klassifikatoren für diese Arbeit zusätzlich unterstützt. Es ist anzumerken, dass die Arbeit mitunter auf den Erkenntnissen der hier genannten Arbeiten beruht und ohne diese nicht umzusetzen wäre.

Kapitel 4

Preprocessing

Im Laufe dieser Arbeit sollen Klassifikatoren anhand von Audiodaten und Annotationen trainiert und ausgewertet werden.
Der für die Audiodaten zur Verfügung stehende Datensatz besteht aus Video- und Audioaufzeichnungen von 38 Teammeetings innerhalb der Veranstaltung „SWP“ von 2012 bis 2016 der Leibniz Universität Hannover. Die Teammitglieder waren hauptsächlich Studierende der Informatik, welche innerhalb der Veranstaltung ein Softwareprojekt umsetzen sollten. Inhalt der Videos ist die Anforderungsanalyse der jeweils zu erstellenden Software. Es wurde ein Meeting pro Team aufgezeichnet, wobei die Teamgröße zwischen fünf und neun Mitgliedern lag. Die Aufzeichnungen besitzen eine Länge von 15 bis 75 Minuten und wurden bereits von Psychologen nach dem act4teams Kodierungsschema analysiert.

4.1 Visualisierung der Annotationsdaten

Für jedes Video existiert eine CSV-Datei, welche von ausgebildeten Psychologen erstellt wurde. Diese enthält Annotationen, welche jede Interaktion während eines Meetings beschreiben. Insbesondere Start- und Endzeitpunkt sowie der act4teams Code jeder erkannten Interaktion sind vermerkt. Die Daten werden genutzt, um die Audioausschnitte aus den Meetings zu extrahieren und ihnen einen act4teams Code zuzuordnen. In den Kapiteln 5 und 6 können die Merkmale (Features) aus diesen Ausschnitten extrahiert werden. Ohne die Annotationen gebe es keine Daten, um die späteren Lernmethoden validieren zu können.
Eine Sichtung der Ausgangsdaten ist wichtig, um möglichst früh Vorkenntnisse über die Verteilung der unterschiedlichen Klassen (act4teams Codes) zu erhalten. Diese Beobachtungen helfen auch entsprechende Lernmethoden auszuwählen. Zum einen ist das Verhältnis zwischen *destruktiven* und *nicht-destruktiven* Interaktionen interessant, des Weiteren auch das Verhältis zwischen den verschiedenen *destruktiven* Interaktionsklassen. Das Wissen

über diese Verhältnisse kann die Auswahl der Klassifikatoren und ihre Parameter beeinflussen.

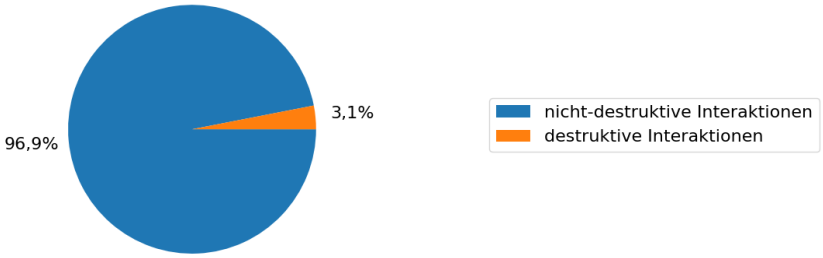


Abbildung 4.1: 3:97 Verhältnis (gerundet) zwischen *destruktiven* und *nicht-destruktiven* Interaktionen im Annotationsdatensatz.

Aus Abbildung 4.1 wird sichtbar, dass die *nicht-destruktiven* Interaktionen signifikant überwiegen. Insgesamt existieren 27807 Interaktion. Der Datensatz kann als deutlich unausgewogen bezeichnet werden, da nur ungefähr 3% (854) der Interaktionen *destruktiv* sind. Das Verhältnis in den einzelnen Meetings verhält sich ähnlich. Diese Verteilung kann mit ähnlichen Beobachtungen von Klünder et al. [38] bestätigt werden.

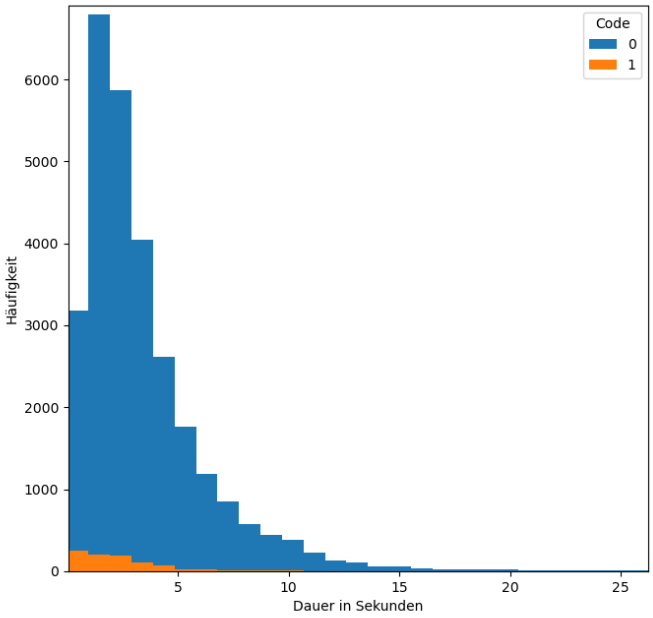


Abbildung 4.2: Histogram der Länge der *destruktiven* (orange) und *nicht-destruktiven* (blau) Interaktionen bis 25 Sekunden.

Abbildung 4.2 stellt ein Histogramm der Länge der Interaktionen dar. Die Abbildung wird nur bis Sekunde 25 dargestellt, weil danach nur noch sehr vereinzelt Interaktionen auftreten. 0,2% aller Interkationen sind länger als 25 Sekunden. Die mittlere Dauer der Interaktionen liegt bei 3.05 Sekunden. 97% aller Interaktionen befinden sich im Intervall zwischen null und zehn Sekunden. Diese Daten werden für die Feature Extraktion in Kapitel 6 benötigt.

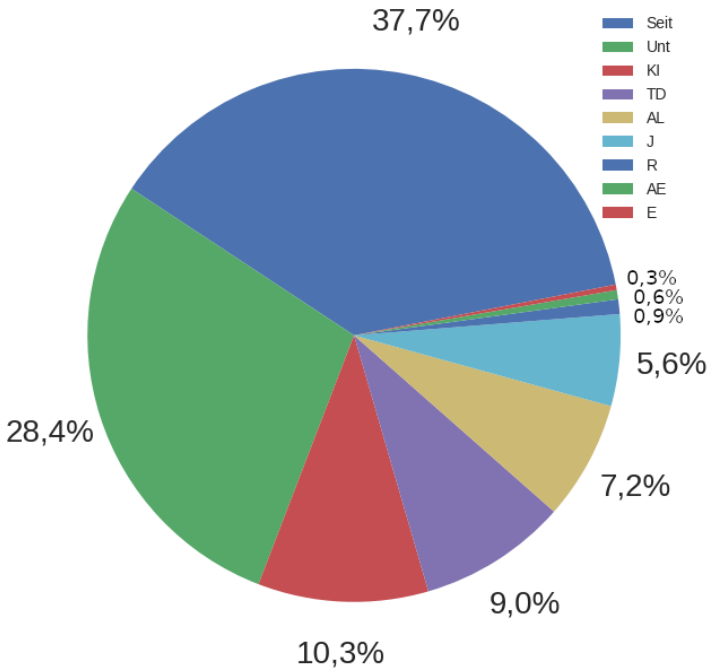


Abbildung 4.3: Zusammensetzung der *destruktiven* Interaktionen aus act4teams Codes.

Abbildung 4.3 stellt den Anteil aller *destruktiven* Interaktionen nach act4teams Codes dar. Es wird ersichtlich, dass Seitengespräche und Unterbrechungen fast 70% der *destruktiven* Interaktionen ausmachen. Danach folgen *KI, TD, AL* und *J*. Der Anteil der Klassen *R, AE und E* liegt unter 1%. Diese Interkationen kommen weniger als zehn Mal im Datensatz vor. Interaktionen der Klasse *S* existieren nicht im Datensatz.
Eine Klassifizierung der vier Klassen *R, AE, E* und *S* kann nicht erfolgreich sein, da zu wenige Interaktionen für Training- und Testdatensatz zur Verfügung stehen. Aus diesem Grund werden diese Klassen fortan in allen weiteren Berechnungen ignoriert. Für den Datensatz können nur sechs von zehn Klassen als *destruktive* Interaktionen gewertet werden. Die genannten Ergebnisse sind in Anhang A.1 zusammengefasst.

4.2 Aufbereitung der Audiodaten

Jedes der 38 Meetings wurde in Videos dokumentiert. Teilweise existieren mehrere Videodateien pro Meeting, wenn die maximale Aufnahmelänge der verwendeten Kamera überschritten wurde. Die Tonspur in den Videos wurde dabei vom internen Kameramikrofon aufgezeichnet, weshalb die Qualität der Aufnahmen je nach Position der Kamera stark variiert. Viele der Audiospuren enthalten Neben- und Störgeräusche sowie starkes Rauschen. Dies ist darauf zurückzuführen, dass die meisten Videokameras kein besonders gutes Mikrofon besitzen und aus einigen Videos ersichtlich wird, dass die Kamera mit einem Abstand von mehreren Metern zu den Teilnehmern aufgestellt wurde. Dadurch werden sowohl Nebengeräusche als auch Rauschen stärker aufgezeichnet. Die Stimmen hingegen geraten in den Hintergrund. Des Weiteren gibt es deutliche Qualitätsunterschiede in der Aufnahme der einzelnen Beteiligten, da nicht jeder in gleichem Abstand und gleicher Richtung zum Mikrofon sitzt.
Für einige Videos existieren zusätzliche Audiodateien, welche parallel mithilfe eines externen Voice Recorders aufgenommen wurden. Der Voice Recorder befand sich dabei immer in unmittelbarer Nähe zu den Teammitglieder, sodass diese Aufnahmen eine deutlich höhere Qualität aufweisen. Jedoch wurde als Audiodatenquelle primär die Audiospur der Videos gewählt. Es ist wahrscheinlich, dass in der Realität in Entwickler-Meetings keine Zeit besteht, eine professionelle Audioaufnahme zu machen. Aus diesem Grund spiegeln die Aufnahmen der Kamera eine realistischere und alltagsnähere Meetingsituation wieder als die Voice Recorder Aufnahmen. In sechs Fällen sind die Audioaufzeichnungen der Kamera so unverständlich und leise, dass auf die Voice Recorder Aufzeichnungen zurückgegriffen werden musste.
Mithilfe der Software ffmpeg [66] werden die Audiospuren aus den Videodateien extrahiert und alle Kanäle zu einem Monokanal zusammengefasst. Die Zusammenfassung ist sinnvoll, da beim späteren maschinellen Lernen nicht die Position und Richtung der Audioquellen einbezogen werden soll. Die Tonspuren werden im verlustfreien WAV-Format abgespeichert, um keine weiteren Qualitätsverluste zu erleiden.

Die Audiodateien der Entwicklermeetings weisen Qualitätsunterschiede auf. Häufig unterscheidet sich die Lautstärke der Stimmen zwischen den einzelnen Audiodateien erheblich. Auch innerhalb einer Aufnahme kommt es zu Qualitätsunterschieden. Die Gründe dafür werden in den folgenden Punkten zusammengefasst.

1. Stör-/Nebengeräusche und Übersteuerungen,

2. Unterschiedliche Lautstärken der einzelnen Stimmen (teilweise, weil Personen nicht in Richtung Mikrofon sprechen),

3. Erhöhtes Rauschen (weil das Mikrofon zu weit von den Teilnehmern entfernt ist)

Für die spätere Verwendung der Audiodaten im Kontext des maschinellen Lernens ist es wichtig, dass die Qualität der Daten so wenig wie möglich variiert, um die Klassifizierung nicht unnötig zu beeinflussen. Aus diesem Grund werden die Audiodateien vor der weiteren Nutzung vorverarbeitet, um die oben genannten Probleme 1.-3. zu reduzieren. Die Audiosoftware Audacity bietet spezielle Filter an, welche eine Verbesserung der Audioqualität erziehlen. Auch Herrmann [26] nutzte für die Aufbereitung von Meetingaufzeichnungen ähnliche Filter.

4.2.1 Compressor

Ein Kompressor verringert die Lautstärke der lauten Töne, sodass Peaks durch Übersteuerungen, aber auch sehr laute Stimmen reduziert werden [3, 51, 54]. Durch diese Anpassung wird erreicht, dass zum Mikrofon abgewandte oder sehr leise sprechende Personen nicht so stark durch andere Personen übertönt werden. Der Kompressor wird mit folgenden Parametern betrieben:

- Threshold: -14 dB; der Threshold gibt an, ab welchem Pegel der Kompressor die Lautstärke verringern soll. Bei leisen Stimmen liegt der Lautstärkepegel in den vorliegenden Audiodateien meistens unter -14 dB. Laute Stimmen hingegen reichen bis zu -6 dB. Der gewählte Threshold sorgt dafür, dass der Kompressor alle Pegel über -14 dB verringert.

- Noise Floor: -40 dB; dieser Wert kann ignoriert werden, da keine Gain-Verstärkung vorgenommen wird.

- Ratio: 2.8:1; gibt an, wie stark die Kompression erfolgt. Damit laute Stimmen im Nachhinein nicht unnatürlich klingen, wird eine eher mittlere Abschwächung von 2.8:1 gewählt [51, 54].

- Attack Time: 0.20 secs; die Attack Time gibt an, wie schnell der Kompressor eingreifen soll, wenn sich der Pegel über den Threshold bewegt. Es wird Audacitys Standardwert genutzt [3].

- Release Time: 1 sec; die Zeit nachdem der Kompressor nach einer lauten Passage wieder deaktiviert wird. Eine Release Time von 1 Sekunde ist ein Standardwert [3] und sorgt für einen fließenden Übergang.

- Make-up gain for 0 dB after compressing: Off; nach der Kompression soll keine Gain-Anhebung auf 0 dB erfolgen.

- Compress based on Peaks: OFF; bedeutet, dass Audiosignale über dem Threshold angeschwächt werden. Bei Aktivierung würden alle Pegel unter dem Threshold verstärkt werden.

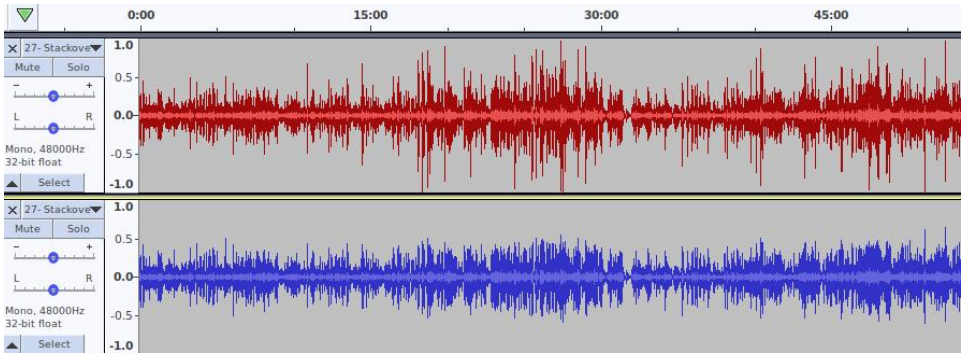


Abbildung 4.4: Originale Audiospur (oben) und Audiospur nach Anwendung des Kompressors (unten)

4.2.2 Normalizer

Die Audiodateien sind nach Einsatz des Kompressors teilweise in ihrer Gesamtlautstärke verringert worden. Es lässt sich aber nicht genau sagen, ob alle Dateien eine ähnliche Lautstärke besitzen. Im Anschluss sorgt der Normalizer für eine Anpassung der maximalen Amplitude [3]. Diese wird für alle Audiospuren auf den gleichen Wert gesetzt. Oftmals bewirkt dies eine Verstärkung des Audiosignals, welche sinnvoll erscheint, da somit der Dynamikumfang erhöht wird. Ein lauteres Signal grenzt sich eher vom Grundrauschen ab und ist später leichter zu verarbeiten. Für den Normalizer wurden folgende Paramter gewählt:

- Remove DC offset: ON; Audiospuren können einen DC Offset besitzen, also einen Gleichanteil innerhalb des Audiosignals. Dieser Gleichanteil wird auf die 0.0 Amplitude korrigiert, damit es nicht zu Verzerrungen oder Übersteuerung kommt.
- Normalize Maximum Amplitude to: -1 dB; gibt den maximalen Pegel nach Verstärkung an. -1 dB ist ein Standardwert [3]. Es verbleibt noch 1 dB nach oben und es kann durch die Normalisierung nicht zu Clipping (> 0 dB) kommen.
- Normalize stereo channels independently: OFF; wird nicht genutzt, da es sich um Monospuren handelt.

Eine Analyse der durchschnittlichen Lautstärke ergab, dass zwei der sechs genutzten Audiospuren des Voice Recorders signifikant lauter sind als alle

anderen Aufnahmen. Die Lautstärke dieser Dateien wird zusätzlich um -6 dB also ungefähr die Hälfte reduziert, um eine geringere Lautstärkeabweichung zwischen den einzelnen Dateien zu erzielen.

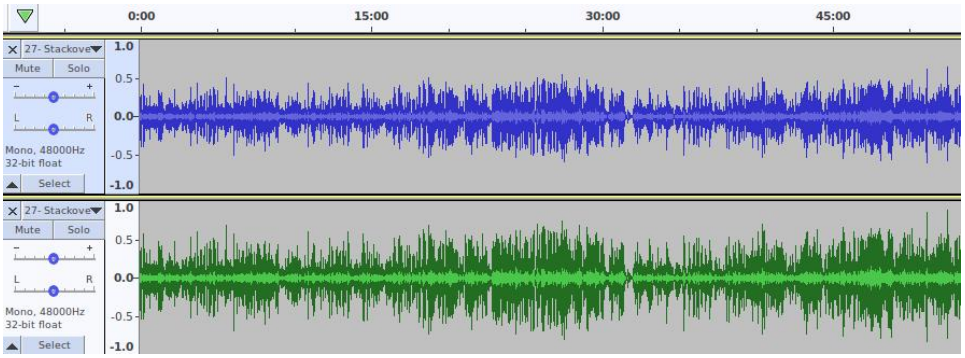


Abbildung 4.5: Komprimierte Audiospur (oben) und normalisierte Audiospur (unten)

4.2.3 Noise Reduction

Die Kompression verringert die durchschnittliche Lautstärke der Audiodateien. Durch die anschließende Normalisierung wird die Lautstärke wieder angehoben. Allerdings werden dabei auch ungewollte Geräusche verstärkt wie das Hintergrundrauschen. In diesem Schritt wird durch Rauschreduzierung eine Abschwächung des Rauschens erziehlt. Der Effekt Noise Reduction wird auf alle Audiodateien angewendet. Die folgenden Parameter werden laut Audacity [3] für die Rauschreduzierung in Gesprächen empfohlen und genutzt:

- Noise Reduction: 6 dB; das erkannte Rauschen wird um 6 dB reduziert. Dies entspricht ungefähr der Hälfte der Originallautstärke.
- Sensitivity: 6; je höher die Sensitivity, desto mehr Rauschanteil wird entfernt. Dieser Wert ist ein ebenfalls ein Standardwert.
- Frequency Smoothing: 6 Bands; sorgt für eine Reduktion möglicher Artifakte, welche durch die Rauschreduzierung entstehen können.
- Noise: Reduce; das detektierte Rauschen wird aus der Audiospur gefiltert.

4.3 Auswahl der Audiofeatures

Maschinelle Lernmethoden lernen und klassifizieren Datenpunkte (Samples) anhand von Features (Merkmalen). Die WAV-Dateien enthalten Informatio-

nen über die zeit- und wertdiskrete Frequenz und Amplitude der Schallwellen. Diese Informationen sind höchstwahrscheinlich nicht ausreichend, um ein Modell zu trainieren. Aus diesem Grund müssen aussagekräftigere Features aus den Audiospuren abgeleitet werden. Bei der Featureauswahl wird verstärkt Fokus auf die sprechenden Personen gelegt, da diese den einzigen Ausgangspunkt für die spätere Erkennung von *destruktiven* Interaktionen darstellen. Dabei ist es wichtig, dass die Personen möglichst nicht anhand der Features identifiziert werden können. Theoretisch kann jede Person destruktive Aussagen tätigen, deshalb sollen die Features möglichst nicht personengebunden sein. Eine Frequenzanalyse der Sprecherstimmen wäre zum Beispiel hinderlich, da dies eher zur Identifikation der Sprecher dient und weniger zur Identifikation der getätigten Aussagen. Die folgenden Features werden in den Kapiteln 5 und 6 aus den Audiospuren extrahiert. Die dafür verwendeten Tools sind die Programmiersprache **Python** [67], die Python Bibliothek **LibROSA** [47] und die Audiosoftware **Praat** [6]. Zusätzlich wurden vorhandene Skripte [15] für Praat angepasst, welche einen Teil der Extraktion vereinfachen.

4.3.1 Sprachfeatures

Sprachfeatures beschreiben in dieser Arbeit Eigenschaften der Stimme und des Sprechens. Besonders diese Features sollen dabei helfen, die Art der Interaktion zwischen Menschen zu erkennen.

F0

F0 ist die Grundfrequenz eines Tons. Diese Frequenz wird beim Sprechen durch die Stimmlippen erzeugt und ist die tiefste Frequenz des entstehenden Tons. Durch Rachen, Lippenraum, Mundraum und Koronalraum werden weitere Frequenzen auf die Grundfrequenz modelliert, sodass unterschiedlichste Klänge beim Sprechen erzeugt werden können [59]. Die Grundfrequenz ist bei jedem Menschen unterschiedlich, weshalb diese eher ungeeignet für eine personenunabhängige Analyse erscheint. Jedoch gibt es kontinuierliche Veränderungen des Grundtons, welche zum Beispiel mit unterschiedlichen Emotionen zusammenhängen [33, 61, 64]. Aus diesem Grund wird F0 und die Standardabweichung als Feature für die Erkennung von *destruktiven* Interaktionen eingesetzt.

Jitter

Jitter beschreibt kleinste Variationen in der Grundfrequenz der Stimme. Johnstone und Scherer [33] entdeckten Korrelationen zwischen verschiedenen Emotionen und dem Jitter der Stimme. Auch bei Stress konnten Veränderungen festgestellt werden. Die Erkennung von Emotionen anhand des Jitters

war bereits erfolgreich [64]. Aus diesem Grund kann auch Jitter zur Analyse beitragen und wird genutzt.

Shimmer

Shimmer ist die durschschnittliche Differenz aufeinanderfolgender Amplituden und wurde wie Jitter bereits zur Emotionserkennung genutzt [64]. Shimmer beschreibt allerdings Änderungen der Amplitude, wohingegen Jitter Frequenzänderungen beschreibt.

MFCC

Die Mel-Frequenz-Cepstrum-Koeffizienten (MFCC) werden häufig im Gebiet der automatischen Spracherkennung genutzt. Ghai et al. [22] konnten mithilfe von MFCC zwischen sechs Emotionen von Sprechern unterscheiden. Weitere Arbeiten nutzen ebenfalls primär MFCC für die Sentiment- und Emotionserkennung [13, 14, 36, 50]. Die MFCC beschreiben die Modulation des durch die Stimmbänder erzeugten Grundtons. Dies geschieht durch die Analyse von periodischen Vorgängen im Frequenzsprektrum. Das daraus entstehende Spektrum des Frequenzspektrums wird als Cepstrum bezeichnet. In nächsten Schritt werden entsprechende Koeffizienten abgeleitet. Bei der MFCC Analyse werden personenabhängige Eigenschaften der Stimme wie die Grundfrequenz entfernt. Die Koeffizienten beschreiben die Änderungen während des Sprechens, welche abhängig vom gesagten Inhalt und der Betonung sind. Jeder extrahierte MFCC Frame enthält 12 MFC-Koeffizienten. Dan und Monica [14] beobachteten, dass der erste Koeffizient nicht zu einer besseren Erkennung beiträgt, da er die Summe der Kanalenergien repräsentiert. Deshalb wird auch in dieser Arbeit der erste Koeffizient nicht genutzt.

Sprechgeschwindigkeit

Die Sprechgeschwindigkeit ist definiert als Anzahl der Silben pro Zeiteinheit. Dabei werden auch Pausezeiten hinzugenommen in denen niemand spricht. Die Sprechgeschwindigkeit spielt besonders bei der Kommunikation mit Anderen eine wichtige Rolle. Hung und Gatica-Perez [29] nutzten unter anderem dieses Feature um den Zusammenhalt von Teams zu untersuchen.

Artikulationsrate und Silbendauer

Die Artikulationsrate beschreibt die Anzahl der Silben pro Sprecheinheit. Anders als bei der Sprechgeschwindigkeit wird hier nur die Zeit gemessen, welche während des Sprechens vergeht. Pausezeiten werden nicht mit einbezogen.

Ein weiteres Feature ist die durchschnittliche Silbendauer. Diese ist definiert als die Zeit, die benötigt wird, um eine Silbe auszusprechen. Mathematisch betrachtet ist die Silbendauer der Kehrwert der Artikulationsrate.

4.3.2 Weitere Audiofeatures

Dieser Abschnitt fasst alle weiteren genutzten Features zusammen, welche aus Schallinformationen gewonnen werden können [24]. Diese Features lassen sich aus jeder Audioaufnahme extrahieren und stehen nicht im direkten Zusammenhang mit der menschlichen Stimme.

Amplitude und Lautstärke

Die Amplitude ist die maximale Auslenkung einer Schwingung. Die Amplitude einer Schallwelle ist maßgeblich für die Lautstärke des empfangenden Tons verantwortlich.

Energie

Die Schallenergie entsteht durch die Bewegungen der Teilchen innerhalb des Schallmediums. Auf diese Weise wird die Schallwelle innerhalb des Mediums transportiert. Je stärker die Teilchenbewegungen, desto größer die Energie.

Leistung

Die Schallleistung ist die pro Zeiteinheit abgegebene Energie einer Schallquelle.

Intensität

Die Schallintensität ist definiert durch die Schallleistung pro Flächeneinheit.

Harmonizität

Die Harmonizität beschreibt die „Lebendigkeit“ eines akustischen Tons.

Kapitel 5

Klassische Lernmethoden

Dieses Kapitel behandelt die Feature Extraktion sowie die Auswahl der Klassifikatoren im Bereich der klassischen Lernmethoden. Klassische Lernmethoden beinhalten in dieser Arbeit alle Algorithmen, welche nicht im Zusammenhang mit künstlichen neuronalen Netzen stehen, da diese erst in Kapitel 6 behandelt werden. Der Grund für die Aufteilung der Lernmethoden hängt mit der unterschiedlichen Strukturierung der Features zusammen, welche die Klassifikatoren als Eingabe benötigen.
Dieses Kapitel ist in drei Teilkapitel aufgeteilt. Zuerst wird auf die Feature Extraktion und die Struktur der Features und Label eingegangen. Anschließend wird der extrahierte Datensatz visualisiert, um im nächsten Schritt geeignete Klassifikatoren auszuwählen und zu trainieren. Die Evaluation findet in Kapitel 7 statt.

5.1 Feature Extraktion

Die Ausgangsdaten bestehen aus den in Kapitel 4.2 aufbereiteten Audiodaten der Teammeetings. Zudem ist für jedes Meeting eine CSV-Datei mit den zugehörigen Annoationen vorhanden. Eine Annotation besteht unter anderem aus Start- und Endzeitpunkt, sowie dem act4teams Code jeder analysierten Interaktion. Die Annotationen der Psychologen werden für jedes Meeting ausgelesen. Mithilfe der Start- und Endzeitpunkte der Annotationen kann die Audiodatei des dazugehörigen Meetings zerteilt werden, um für jede Annotation ein Audiosegment zu erhalten. Anschließend können die in Kapitel 4.3 ausgewählten Features für jedes Segment extrahiert werden.
Die Sprachfeatures F0, Jitter, Shimmer, Sprechgeschwindigkeit, Artikulationsrate und Silbendauer werden für jedes Audiosegment extrahiert. Für alle weiteren Features wie MFCC, Energie, Leistung, Intensität, Amplitude und Lautstärke wird jedes Segment in drei gleich große Teile zerteilt. Daraufhin werden diese Features für jedes der drei Parts einzeln extrahiert, um detailliertere Werte für das Segment zu erhalten anstatt eines Druchschnittwertes.

Dieses Verfahren ist mit den oben genannten Sprachfeatures nicht möglich, da die 3 einzelnen Audioteile oftmals zu kurz für die Extraktion sind. Um beispielsweise eine Analyse der Artikulationsrate durchzuführen, müssen mehrere Peaks in einer Audiospur gezählt und zusammengefasst werden. Ist eine Audiosequenz kürzer als 0,5 Sekunden, kommt es vermehrt vor, dass die Analyse nicht durchgeführt werden kann, da zu wenige Peaks erkannt werden. Da viele der Interaktionen eine Sekunde oder kürzer sind (siehe Abbildung 4.2) werden die Sprachfeatures nur einmal für das gesamte Segment extrahiert.

Teilweise kommt es vor, dass die Feature Extraktion der Teilsegmente fehlschlägt, da diese auch zu wenig Informationen enthalten können. Die Features dieser Audiospuren sind dann nicht vollständig, wodurch die ganze Annotation nicht genutzt werden kann und nicht im Featuredatensatz auftaucht. Aus diesem Grund verkleinert sich der Datensatz der klassischen Methoden um 19% im Vergleich zu den Ausgangsdaten. Das Klassenvehältnis von 3:97 (gerundet) bleibt jedoch gleich. Auch in der Feature Extraktion der rekurrenten neuronalen Netze in Kapitel 6 kommt es zu einer Verkleinerung des Featuresdatensatzes, jedoch im geringeren Ausmaß, sodass sich die Datensätze der beiden Ansätze in ihrer Größe unterscheiden.

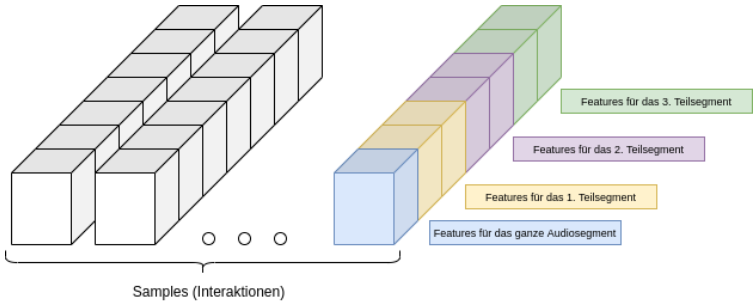


Abbildung 5.1: Vereinfachte Struktur des Featuredatensatzes der klassischen Methoden.

Die jeweils für eine Annotation extrahierten Features werden in einem Vektor gespeichert. Dieser enthält insgesamt 105 Features. Zusätzlich wird der act4teams Code der jeweiligen Annotation vermerkt. Aus der Feature Extraktion aller Annotationen resultiert ein Featuredatensatz mit 22550 Featurevektoren mit 105 Dimensionen und ihren dazugehörigen act4teams Codes. Abbildung 5.1 zeigt eine vereinfachte und symbolische Darstellung des Featuredatensatzes ohne zusätzliche act4teams Codes. Die Speicherung und weitere Verarbeitung der Datensätze erfolgt mithilfe der Python Bibliotheken *numpy* [25] und *pandas* [68].

Der Featuredatensatz bildet die Grundlage für zwei Klassifizierungsarten, die in dieser Arbeit genutzt werden: Binäre- und Multi-Class Klassifizierung.

Binäre Klassifizierung

Die binäre Klassifizierung ist der erste Schritt der Analyse, wobei nur zwischen *destruktiven* und *nicht-destruktiven* Interaktionen unterschieden wird. Die hierbei als *destruktiv* klassifizierten Interaktionen können in Schritt 5.1 genauer klassifiziert werden. Diese Vorauswahl der *destruktiven* Interaktionen spielt eine wichtige Rolle, da nur 3% aller Interaktionen in diese Teilmenge fallen. Eine Klassifizierung zwischen den sechs *destruktiven* und den restlichen *nicht-destruktiven* Kategorien würde die Komplexität des Problems deutlich erhöhen, da der Datensatz mit einem Verhältnis von 3:97 ohnehin schon unausgewogen ist. Es ist höchst unwahrscheinlich, dass eine Klassifizierung zwischen mehr als zwei Klassen mit dem gesamten Datensatz erfolgreich ist, weil der Anteil der *destruktiven* Interaktionen bereits sehr klein ausfällt. Um dem unausgewogenen Verhältnis der Klassen entgegenzuwirken, werden für das Training der Klassifikatoren in Abschnitt 5.3 Methoden wie Undersampling und Klassengewichte genutzt.
Aus dem ursprünglichen Datensatz wird ein separater Datensatz für die binäre Klassifizierung gebildet. Der Unterschied besteht darin, dass alle *nicht-destruktiven* act4teams Codes durch Nullen und die *destruktiven* Codes durch Einsen kodiert werden. Dieses Vorgehen reduziert die 44 act4teams Codes auf nur zwei Klassen, welche für die binäre Klassifizierung von Bedeutung sind. Diese Klassen dienen als Label für das überwachte Lernen und alle späteren Tests. Durch die Kombination aus Featurevektoren und Labeln (den wahren Klassen) wird beim überwachten Lernen eine Klassifizierung erlernt.

Multi-Class Klassifizierung

In diesem Schritt sollen *destruktive* Interaktionen genauer klassifiziert werden. Um diese Interaktionen ihren genauen act4teams Codes zuordnen zu können, wird eine Multi-Class Klassifizierung genutzt. Hierfür wird ein Datensatz erstellt, welcher nur Samples von *destruktiven* Interaktionen des ursprünglichen Datensatzes enthält. 97% der Daten fallen weg, wodurch das Klassifizierungsproblem nur auf die *destruktiven* Interaktionen reduziert wird.

5.2 Visualisierung der Feature-Datensätze

Aus Unterteilung der Klassifizierung resultieren zwei Datensätze, welche in diesem Unterkapitel gesichtet und beschrieben werden, um im nächsten Schritt passende Klassifikatoren auszuwählen. Anhand der Verteilung der Daten können außerdem erste Prognosen und mögliche Probleme identifiziert werden. Der Datensatz für die binäre Klassifizierung ist durch den Überfluss von *nicht-destruktiven* Interaktionen sehr unausgewogen. Das Verhältnis von

3:97 wurde bereits im Kapitel 4.1 analysiert. Der zweite Datensatz für die Multi-Class Klassifizierung enthält nur *destruktive* Interaktionen. Das Verhältnis der Klassen ist ebenfalls in Kapitel 4.1 zu finden. Dieser Datensatz ist deutlich ausgewogener, enthält allerdings auch mehr Klassen, wodurch die Klassifizierung schwieriger wird. Um einen Überblick über die Komplexität des Problem zu erhalten, ist es sinnvoll, die Features gegeneinander zu visualisieren. Dafür gibt es zwei weit genutzte Möglichkeiten: Scatterplot Matrix und PCA [11, 23, 41].

5.2.1 Scatterplot Matrix

Die Scatterplot Matrix stellt Features anderen Features gegenüber. Es wird eine Matrix erzeugt, wobei jedes Feld eine Punktwolke (Scatterplot) von zwei Features darstellt. Dadruch kann jeweils ein Feature in Abhängigkeit eines anderen Features betrachtet werden [23, 41]. Die Punkte spiegeln alle Datenpunkte im Datensatz wieder und sind entsprechend ihrer Klasse markiert. Bei einfachen Klassifizierungsproblemen ist es hier bereits möglich, eine räumliche Trennung der einzelnen Klassen zu erkennen.

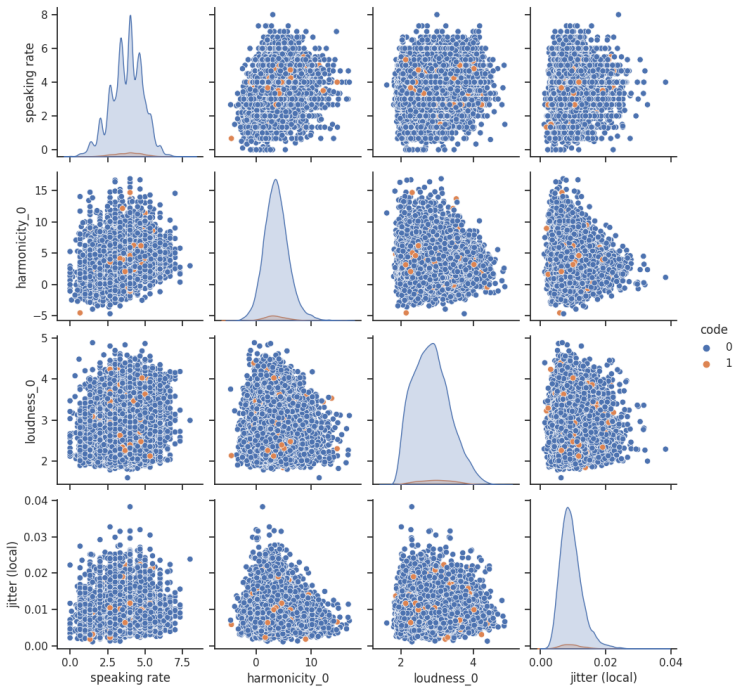


Abbildung 5.2: Scatterplot Matrix für vier Features des binären 3:97 Datensatzes.

Die erste Abbildung zeigt die 4x4 Scatterplot Matrix für den ersten Featuredatensatz also für die binäre Klassifizierung zwischen *destruktiven*

und *nicht-destruktiven* Interaktionen. Es fällt auf, dass in den Graphen keine klare Trennung der beiden Klassen mit dem bloßen Auge möglich erscheint. An der Häufigkeitsverteilung der Features wird auch erkenntlich, dass diese annähernd normalverteilt sind und oftmals beide Klassen einen ähnlichen Erwartungswert besitzen. Diese Beobachtungen deuten darauf hin, dass eine Klassifikation anhand der Features einen erhöhten Schwierigkeitsgrad aufweist, da eine Gegenüberstellung zweier Features nicht ausreicht, um eine räumliche Trennung der beiden Klassen zu erkennen.
Für den zweiten Datensatz, welcher verschiedene Klassen von *destruktiven* Interaktionen enthält, sehen die Punktwolken ebenfalls nicht voneinander trennbar aus.

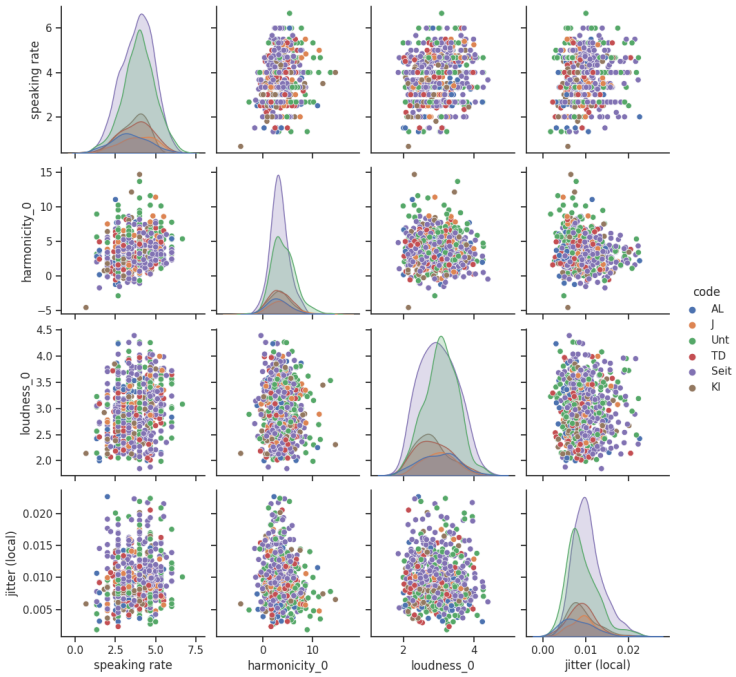


Abbildung 5.3: Scatterplot Matrix für vier Features des *destruktiven* Multi-Klassen Datensatzes.

Die Graphen des zweiten Datensatzes enthalten nur 3% der Datenpunkte des ersten Datensatzes, weshalb die Punktwolken weniger dicht erscheinen. Teilweise heben sich einige Punkte einer Klasse von anderen ab, jedoch ist dies nur die Ausnahme. Die meisten Datenpunkte liegen als Wolke um einen Mittelwert verteilt und überschneiden sich.

5.2.2 Hauptkomponentenanalyse

Die vorherige Visualisierungsmethode ist limitiert, weil nur der Zusammenhang zwischen jeweils 2 Merkmalen dargestellt werden kann [41].

Interessanter ist es, einen Zusammenhang zwischen allen Merkmalen zu visualisieren. Die Featurevektoren enthalten allerdings 105 Dimensionen (Features), sodass eine Darstellung in einem zwei- oder dreidimensionalen Koordinatensystem nicht möglich ist.

Die Hauptkomponentenanalyse (PCA) ist ein Verfahren, welches für die Reduktion von Dimensionen des Datensatzes und damit auch die Visualisierung genutzt werden kann. PCA versucht die Vielzahl von Merkmalen durch geeignete Linearkombinationen darzustellen, die sogenannten Hauptkomponenten. Mithilfe dieses Verfahrens, ist es möglich die Dimensionen der Featurevektoren so weit zu reduzieren, bis der Datensatz dargestellt werden kann [11, 23].

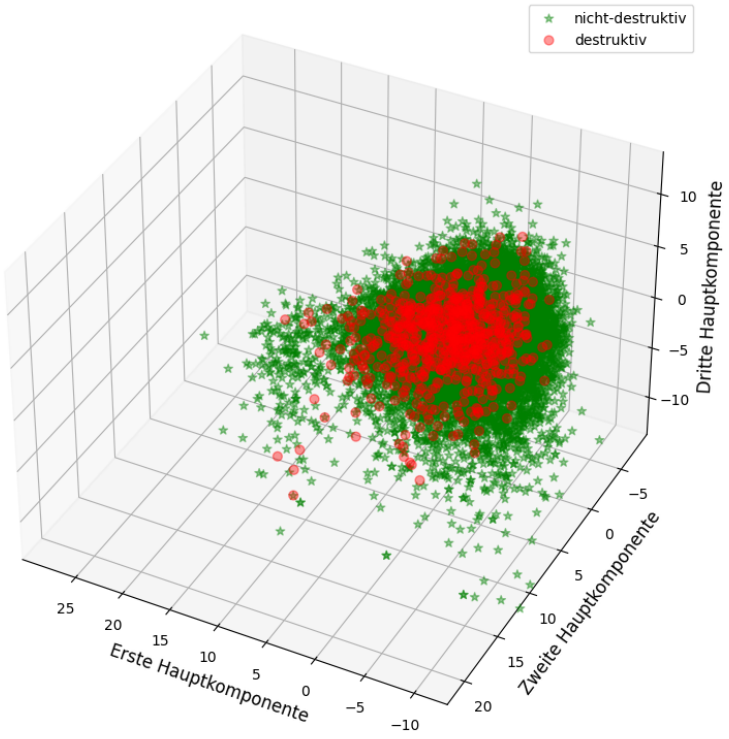


Abbildung 5.4: 3D PCA Visualisierung des binären Datensatzes der klassischen Methoden.

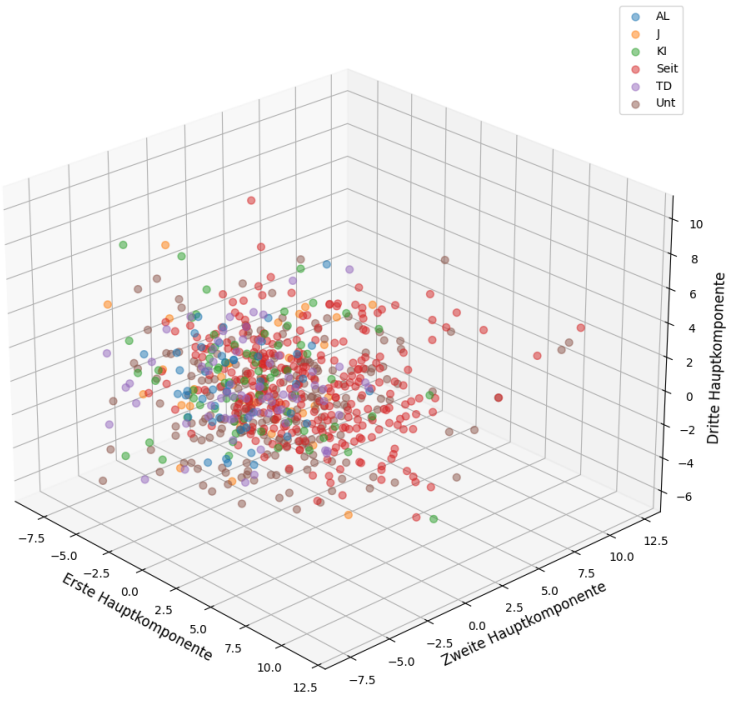


Abbildung 5.5: 3D PCA Visualisierung des Multi-Class Datensatzes der klassischen Methoden.

Die Visualisierung der Datensätze mit drei Hauptkomponenten in Abbildung 5.4 und 5.5 bestätigt die Vermutungen der Scatterplot Beobachtungen. Es ist keine räumliche Separierung der Klassen erkennbar. Die Punktwolken liegen ineinander. In Abbildung 5.4 ist erkennbar, dass die *destruktiven* Datenpunkte hauptsächlich im Zentrum der *nicht-destruktiven* Punktwolke liegen. In Anhang B sind zusätzlich die PCA Visualisierungen mit jeweils zwei Hauptkomponenten zu sehen.
Zusammengefasst lässt sich beobachten, dass in beiden Feature-Datensätzen keine triviale Trennung zwischen den verschiedenen Klassen sichtbar wird. Für die folgende Klassifizierung mithilfe der Featuredatensätze bedeutet dies eine hohe Komplexität des Ausgangsproblems.

5.3 Training der Klassifikatoren

Innerhalb der klassischen Lernmethoden gibt es eine Vielzahl von Algorithmen, welche je nach Problem ausgewählt und trainiert werden müssen. Die hier verwendeten Klassifikatoren gehören zum Teilgebiet des überwachten Lernens, da in den Datensätzen bereits hinterlegt ist, in welche Klasse ein Sample (Featurevektor) einzuordnen ist.

Für das Training eines Klassifikators muss der Featuredatensatz jeweils in zwei Datensätze aufgeteilt werden. Einer der Teildatensätze dient als Trainingsdatensatz, welcher dem Klassifizierer bekannt ist und an dem gelernt werden soll. Der andere Datensatz dient als Testdatensatz und ist dem Klassifizierer unbekannt. Dieser Datensatz wird zum späteren Testen genutzt, um zu erkennen, ob neue Daten ebenfalls korrekt klassifiziert werden, oder der Klassifizierer nur die Trainingsdaten auswendig gelernt hat (Overfitting).
Die binären Klassifikatoren werden mit zwei unterschiedlichen Methoden trainiert. Die erste Methode nutzt einen Trainingsdatensatz, welcher ein ausgewogenes Verhältnis von 50:50 beziehungsweise 1:1 zwischen *destruktiven* und *nicht-destruktiven* Klassen besitzt. Da im originalen Datensatz das Verhältnis nicht ausgewogen ist wird der Datensatz mithilfe von Undersampling angepasst. Dafür wird der *Random Undersampler* der imbalanced-learn Bibliothek [44] genutzt. Die zweite Trainingsmethode nutzt das Klassenverhältnis des originalen Datensatzes. Dieses liegt ungefähr bei 3:97, wobei die *nicht-destruktiven* Interaktionen überwiegen. Für die anschließende Multi-Class Klassifizierung wird aufgrund der geringen Anzahl an Daten nur das ursprüngliche Klassenverhältnis verwendet. Zudem erfolgt eine Skalierung der Trainingsdatensätze. Es wird der *Standardscaler* genutzt, welcher dafür sorgt, dass die Features einen Mittelwert von 0 und eine Standardabweichung von 1 besitzen. Das ist wichtig, weil einige Klassifikatoren wie die SVM diese Skalierung erwarten [52].

Trainingsdatensatz	relatives Verhältnis	Sampling Methode
50:50 Datensatz	1:1	Undersampling
3:97 Datensatz	3:97	-
Multi-Class Datensatz	siehe Abb. 4.3	-

Tabelle 5.1: Relative Verhältnisse der Datensätze für das Training aller Klassifizierer. In Kapitel 6 werden andere Datensätze als in diesem Kapitel genutzt, jedoch stimmen die gerundeten Verhältnisse überein.

Die klassischen Lernmethoden besitzen Hyperparameter, welche während des Trainings angepasst werden sollten, um bessere Ergebnisse zu erzielen. Dies muss für jeden der Trainingsdatensätze separat erfolgen, da dass Klassenverhältnis ebenfalls einen Einfluss auf die Hyperparamter hat. Diese Anpassung geschieht über GridSearchCV, wobei der Balanced-Accuracy Score als Gütekriterium genutzt wird. Nicht alle Klassifikatoren eigenen sich für jede Art von Problem, weshalb die hier getroffene Auswahl an Methoden jeweils begründet wird. Beispielsweise scheitert ein Algorithmus wie K-Nearest-Neighbors bei der Erkennung *destruktiver* Interkationen, da

dieser einen n-dimensionalen Datenpunkt mit den umliegenden Punkten abgleicht und ihm die Klasse zuordnet, welche die meisten seiner Nachbarn besitzen. Im ersten Datensatz konnte man sehen, dass keine klare Trennung der Klassen vorliegt und meistens viele *nicht-destruktive* Nachbarn um einen *destruktiven* Datenpunkt herumliegen. Daraus resultiert eine falsche Einordnung von *destruktiven* Interkationen als *nicht-destruktiv*. Ein solcher Algorithmus eignet sich also nicht für die Problemstellung.
Die folgenden Klassifizierer stammen aus der Python Bibliothek *scikit-learn* [52], welche auch für das Training und die Hyperparameterauswahl geeignete Methoden bereitstellt.

5.3.1 Support Vector Machines

Support Vector Machines (SVMs) eignen sich für eine Vielzahl von Klassifizierungs- und Regressionsproblemen. Sie wurden schon oft für die Erkennung von Stimmungen und Emotionen anhand von Audiofeatures genutzt [13, 22, 29, 50, 61]. Eine SVM berechnet eine Hyperebene in einem mehrdimensionalen Raum, welcher die Featurevektoren enthält. Ziel ist es die unterschiedlichen Label durch diese Ebene zu trennen. Optimalerweise können so zwei Klassen voneinander unterschieden werden. Die einfachste Trennungsmethode ist wie eben beschrieben durch eine Hyperebene. Viele Probleme sind allerdings nicht-linear trennbar, woraufhin SVM mit verschiedenen Kernel Funktionen angepasst wurden. Ein Kernel transformiert die Featurevektoren in einen höherdimensionalen Raum, mit dem Ziel in diesem Hyperraum eine lineare Separierung der Klassen zu erhalten, welche im normalen Raum nicht möglich wäre [22].
Aus Abschnitt 5.2 wird erkenntlich, dass eine lineare SVM wahrscheinlich nicht für das vorliegende Problem geeignet ist. Weitere bekannte Kernel, die auch in dieser Arbeit genutzt werden, sind der polynomielle- und der radial-basis-function (RBF) Kernel. Diese Kernel sind nicht-linear und weisen das im vorherigen Absatz beschriebene Verhalten auf. Arbeiten wie die von Dahake et al. [13] zeigen, dass beide Kernel bessere Ergebnisse im Bereich der Emotionserkennung mit Audiosignalen liefern können als eine lineare SVM. Der 3:97 Datensatz ist stark unausgewogen, sodass die Möglichkeit besteht, dass alle Samples als *nicht-destruktiv* klassifiziert werden. Um diesem Problem entgegenzuwirken, müssen die Hyperparameter der SVM verändert werden. Die Hyperparameter *kernel*, *gamma*, C und *classweight* werden für die drei verschiedenen Datensätze experimentell mit der Methode GridSearchCV ermittelt [52]. Die Ergebnisse sind in Tabelle 5.2 zu sehen. Der *classweight* Parameter gibt die Gewichtug der einzelnen Klassen an. Wie erwartet liefert GridSearchCV bei den unausgewogenen Datensätzen den Wert „balanced“, also eine Gewichtung, welche dem Klassenungleichgewicht entgegenwirkt, die besten Ergebnisse. Der *RBF Kernel* lieferte in allen Fällen etwas bessere Ergebnisse als der *Polynomialkernel*. C ist

der Regulierungsparameter, welcher Einfluss auf den Abstand zwischen Hyperebene und Datenpunkte hat. Der Hyperparamter wird eingesetzt, um Overfitting, also ein Auswendiglernen der Trainingsdaten, zu vermeiden. C behält den Standardwert von 1 für die besten Validierungsergebnisse. *Gamma* ist der Kernel Koeffizient, welcher ebenfalls Over- beziehungsweise Underfitting regulieren kann. Für die binäre Klassifizierung wird *gamma* auf den Standardwert „scale“ gesetzt [19]. In der Multi-Class Klassifizierung wurden mit einem *gamma* Wert 0,01 die besten Ergebnisse erzielt.

Trainingsdatensatz	kernel	gamma	C	classweight
50:50	rbf	scale	1	None
3:97	rbf	scale	1	balanced
Multi-Class	rbf	0,01	1	balanced

Tabelle 5.2: Vollständige Werte der SVM Hyperparamter, welche durch GridSearchCV ermittelt wurden.

5.3.2 Random Forests

Ghai et al. [22] nutzten neben Support Vektor Machines auch Random Forest für die Klassifizierung von Emotionen anhand von MFCC und weiteren Features. Grundlage für einen Random Forest sind Entscheidungsbäume. Ein Entscheidungsbaum Algorithmus versucht im Training Features anhand ihrer Werte aufzuteilen, sodass nach mehreren Unterscheidungen eine Trennung der Klassen erfolgt. Der Entscheidungsprozess kann dabei als Baum modelliert werden. Beim Klassifizieren eines Samples wird an der Wurzel des Baum angefangen und je nach den Werten der einzelnen Features ein bestimmter Pfad abgelaufen. Der letzte Knoten des Pfades ist ein Blatt, welches die Klassenvorhersage des Samples bestimmt [17].
Ein Random Forest besteht aus einer beliebigen Anzahl an Entscheidungsbäumen, wobei die Bäume durch unterschiedliche Datensätze und Features trainiert werden. Diese Methode wird als Bagging bezeichnet. Die Klasse, welche am häufigsten unter den Bäumen vorhergesagt wird, bildet die finale Ausgabe das Waldes [17].
Die Hyperparamter *nestimators, maxdepth, criterion* und *classweight* des Random Forests werden mit GridSearchCV angepasst. Eine geringe Baumanzahl (*nestimators*) führt dazu, dass die Klassifizierung ungenau und weniger aussagekräftig ist. Die Tiefe der Bäume (*maxdepth*) wirkt sich auf mögliches Under- beziehungsweise Overfitting aus. Außerdem gibt es verschiedene Methoden, wie die Qualität einer Aufteilung (*criterion*) innerhalb eines Baumes bewertet werden soll. Die Gewichtung der Samples je nach Klasse (*classweight*) wird ebenfalls bestimmt. Die Ergebnisse finden sich in Tabelle 5.3 wieder.

Trainingsdatensatz	nestimators	maxdepth	criterion	classweight
50:50	100	20	entropy	None
3:97	5	5	gini	balanced
Multi-Class	100	5	gini	balanced

Tabelle 5.3: Vollständige Werte der Random Forest Hyperparamter, die durch GridSearchCV ermittelt wurden.

5.3.3 Logistic Regression

Logistic Regression wird verwendet, um Klassifizierungsprobleme durch Regressionen zu lösen. Ramadhan et al. [56] nutzten diese zur Sentiment Analyse von Social Media Daten. Die Logistic Regression berechnet eine Regressionsfunktion, die beispielsweise versucht den *nicht-destruktiven* Interaktionen eine 0 und den *destruktiven* Interaktionen eine 1 zuzuordnen. Wird die Funktion mit einem neuen Datenpunkt getestet, bildet diese das Datum ebenfalls auf einen Wert zwischen 0 und 1 ab. Durch Runden des Ergebnisses kann das Datum schließlich einer der beiden Klassen zugeordnet werden. Logistic Regression eignet sich ebenfalls für die Multi-Class Klassifizierung [39, 52].
Die Hyperparameter für die Logistic Regression können automatisch über GridSearchCV angepasst werden. Zum einen wird der Regularisierungsparamter C getestet, des Weiteren werden verschiedene Regressionsalgorithmen über den *solver* Parameter überprüft. Das Klassengewicht wird ebenfalls wie bei den vorherigen Klassifikatoren in Betracht gezogen. Einige Parameter erhalten einen festen Wert, um Konflikte zwischen einzelnen Paramtern zu verhindern. Als *penalty* wird l2 gewählt. Die maximale Anzahl an Iterationen wird auf 10000 gesetzt, da es sonst vorkommen kann, dass Regressionsmodelle nicht konvergieren.

Trainingsdatensatz	C	solver	classweight
50:50	10	saga	None
3:97	10	saga	balanced
Multi-Class	100	newton-cg	None

Tabelle 5.4: Vollständige Werte der Logistic Regression Hyperparamter, die durch GridSearchCV ermittelt wurden.

5.3.4 Naive Bayes

Naive Bayes ist ein Klassifikator, der ein Sample anhand der statistischen Wahrscheinlichkeiten seiner Features einer Klasse zuordnet. Der Algorithmus

geht davon aus, dass alle Features voneinander unabhängig sind. In den meisten Klassifizierungsproblemen ist diese Voraussetzung allerdings nicht gegeben. Der Grad der Unabhängigkeit der Features beeinflusst deshalb die Performanz des Algorithmus [18, 39]. Obeidi [49] nutze den Klassifikator bereits in ihrer Arbeit für die Erkennung von *destruktiven* Äußerungen in Meetings anhand von transkribierten Audiospuren.
Auch in dieser Arbeit wird Naive Bayes getestet. Da es sich um einen sehr einfachen Klassifizierungsalgorithmus handelt, gibt es wenige Möglichkeiten, diesen durch Veränderungen von Hyperparamtern zu beeinflussen. Deshalb wird auf die automatische Auswahl von Hyperparametern verzichtet.

Kapitel 6

Rekurrente neuronale Netze

Dieses Kapitel bildet den zweiten und damit letzten Ansatz, geeignete Klassifikatoren für die Erkennung *destruktiver* Interaktionen auszuwählen. Neuronale Netze bieten den Vorteil, dass sie in der Lage sind, auch komplexe Muster und Zusammenhänge selbstständig zu erlernen. Dazu benötigen sie jedoch eine ausreichende Menge an Daten.

6.1 Feature Extraktion

Die Featurevektoren für rekurrente neuronale Netze müssen anders aufgebaut sein, als die der klassischen Methoden. Ein rekurrentes Netz lernt ein Muster mithilfe von Sequenzen, welche in mehreren Zeitschritte unterteilt sind. Das bedeutet, anstatt eines Featurevektors, welcher alle extrahierten Features einer Audiosequenz enthält, werden mehrere Vektoren benötigt, wobei die Vektoren aufeinander folgende Zeiträume innerhalb einer Audiosequenz darstellen. Für ein Sample wird also eine Matrix aus mehreren Featurevektoren benötigt, welche eine Sequenz abbilden. Dafür muss ein Annotationssegment in mehrere Teile geteilt werden. Der Unterschied zu den klassischen Methoden besteht darin, dass die Teilabschnitte immer eine konstante Länge besitzen. Für jeden Abschnitt wird ein separater Featurevektor erstellt. Als Features werden MFCC, Amplitude, Intensität, Lautstärke, Energie und Leistung verwendet. Die Länge der einzelnen Teile (Frames) wird durch den MFCC Extraktionsalgorithmus vorgegeben. Jeder Frames ist 93ms lang und wird per Sliding Window mit einer Hop-Länge von 23ms extrahiert. Da die Hop Länge kürzer ist als die Länge eines Frames, überlagern sich diese um jeweils $93ms - 23ms = 70ms$. Aus jedem Frame werden 12 MFC Koeffizienten extrahiert [13]. Dazu kommen die acht restlichen Features, sodass jeder Featurevektor 20 Features für einen Zeitschritt von 93ms enthält. Andere Features konnten nicht verwendet werden, da die Frame-Länge mit 93ms für andere Analysen zu kurz ist. Die Analyse der hier genannten Features ist nicht immer vollständig, falls

zu wenig Informationen in den Audiosegmenten vorhanden sind. Diese Interaktionen werden nicht im Datensatz aufgenommen. Der resultierende Datensatz ist nur um 0,02% kleiner als der Ausgangsdatensatz. Wie bereits in Kapitel 5.1 aufgezeigt wurde, unterscheiden sich die extrahierten Datensätze voneinander. Der Datensatz der rekurrenten neuronalen Netze enthält mehr Daten, allerdings ist das Verhältnis der Featuredatensätze jeweils bei 3:97 (gerundet).
Da Annotationen von unterschiedlicher Länge sind, entseht auch eine unterschiedliche Anzahl von Featurevektoren pro Annotation. Das ist ein Problem, weil neuronale Netze eine konstante Länge der Eingabedaten benötigen. In Kapitel 4.1 wurde festgestellt, dass 97% aller Interaktionen eine Länge von maximal zehn Sekunden aufweisen. Aus diesem Grund werden die Methoden Padding und Trimming zu verwenden, um alle Samples auf eine konstante Länge zu bringen. Aus einer zehnsekündigen Annotation können mit dem oben vorgestellten Verfahren 431 Vektoren mit jeweils 20 Features extrahiert werden. Ist eine Annotation kürzer als 10 Sekunden werden durch Padding die fehlenden Vektoren mit Nullvektoren aufgefüllt. Trimming kürzt Annotationen, welche 10 Sekunden überschreiten und führt damit zu einem Informationsverlust. Dieser Verlust ist allerdings vernachlässigbar, da nur 3% der Interaktionen betroffen sind und die Interaktionen dennoch bis zur zehnten Sekunde analysiert werden.

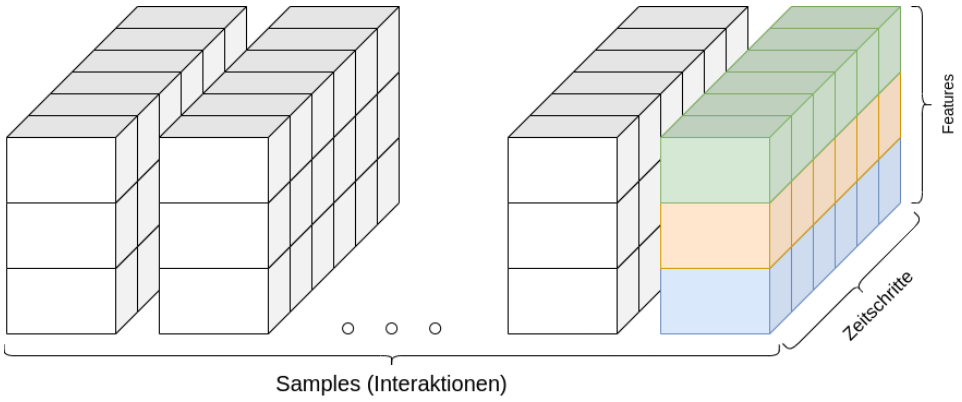


Abbildung 6.1: Symbolische Darstellung von Samples mit mehreren Zeitschritten und mehreren Features pro Zeitschritt. Die unterschiedlichen Farben im letzten Sample stehen für die verschiedenen Features.

Die Klassifizierung wird wieder in zwei Schritte unterteilt. Im ersten Schritt soll nur zwischen *destruktiven* und *nicht-destruktiven* Interaktionen unterschieden werden. Im zweiten Schritt werden nur die *destruktiven* Interaktionen genauer analysiert und einem act4teams Code zugeordnet. Daraus ergeben sich zweit Datensätze für das überwachte Training.

Binäre Klassifizierung

Der erste Datensatz enthält die Featurevektoren aller Interaktionen. Die Label werden wie in Kapitel 5 je nach *destruktiver* oder *nicht-destruktiver* Interaktion durch 1 beziehungsweise 0 kodiert und separat abgespeichert.

Multi-Class Klassifizierung

Für die Multi-Class Klassifizierung werden erneut nur *destruktive* Interaktionen ausgewählt. Die Label müssen jedoch anders kodiert werden, da die Ausgabe eines neuronalen Netzes kein konkreter act4teams Code wie KI, Seit, Unt usw. ist, sondern ein numerischer Vektor. Die sechs *destruktiven* Codes werden mithilfe des ONE-HOT-Encodings kodiert und können auch wieder in ihre ursprüngliche Form dekodiert werden. Das bedeutet, ein Label besteht aus einem Vektor mit sechs Feldern. Jeder Eintrag steht für einen act4teams Code, wovon nur das Feld mit dem jeweils zutreffenden Code eine Eins enthält. Alle anderen Felder enthalten eine Null. Die Reihenfolge der Felder entspricht der alphabetischen Reihenfolge der act4teams Codes. Tabelle 6.1 stellt die Kodierung eines Labels dar.

Code:	AL	J	KI	Seit	TD	Unt
Vektor:	0	1	0	0	0	0

Tabelle 6.1: Repräsentation des act4teams Codes “Jammern” als ONE-HOT Vektor.

Die Ausgabe des neuronalen Netzes wird ebenfalls ein Vektor sein. Dieser enthält sechs Wahrscheinlichkeiten für den jeweiligen Code. Das Feld mit der höchsten Wahrscheinlichkeit entscheidet über den vorhergesagten act4teams Code. Tabelle 6.2 zeigt einen beispielhaften Ausgabevektor.

Code:	AL	J	KI	Seit	TD	Unt
Vektor:	0,01	0,89	0,05	0,02	0,01	0,02

Tabelle 6.2: Beispiel eines Ausgabevektors eines neuronalen Netzes für die Vorhersage des Codes “Jammern”.

6.2 Training rekurrenter neuronaler Netze

In diesem Unterkapitel werden verschiedene Architekturen von rekurrenten neuronalen Netzen vorgestellt. Diese besitzen, wie in Kapitel 2.5 beschrieben,

auch Verbindungen innerhalb einer Schicht. Dadurch ist es möglich, Ergebnisse aus vorherigen Schritten in die aktuelle Berechnung einzubeziehen. Für die binäre Klassifizierung mit neuronalen Netzen werden, wie in Kapitel 5, zwei Trainingsansätze verfolgt. Für die erste Methode wird der originale Trainingsdatensatz durch Undersampling auf ein Klassenverhältnis von 50:50 ausgeglichen. Dafür wird ein Großteil der *nicht-destruktiven* Interaktionen entfernt, bis das Verhältnis zwischen *destruktiven* und *nicht-destruktiven* Samples ausgeglichen ist. Für die zweite Methode wird der originale Trainingsdatensatz beibehalten, stattdessen werden Klassengewichte berechnet, welche das Ungleichgewicht der Klassen ausgleichen soll. In der Multi-Class Klassifizierung wird ebenfalls das unveränderte Klassenverhältnis im Training genutzt. Die Datensätze unterschieden sich im Vergleich zu den Datensätzen für die klassischen Methoden in ihrer Kardinalität und dem Aufbau der Featurevektoren, jedoch sind die relativen Klassenverhältnisse und Sampling Methoden gleich. Eine Zusammenfassung der Trainingsdatensätze findet sich in der Tabelle 5.1. Für die Evaluierung aller Methoden werden Validierungs bzw. Testdatensätze erstellt. Eine genaue Beschreibung dieser Datensätze findet sich im folgenden Kapitel 7. Der Validierungsdatensatz dient auch zur Überwachung des Trainings. Dieses ist in Epochen unterteilt. Nach jeder Epoche wird das Netz anhand des Validierungsdatensatzes getestet, um Probleme wie Overfittung zu erkennen. Die Netze werden mithilfe der Python Machine-Learning Bibliothek **Keras** [10] implementiert. Das Training erfolgt mit dem Google **Tensorflow** Backend [1] für Keras. Verwaltet werden die installierten Bibliotheken mithilfe von **Miniconda**, eines minimalen Paketmanagers von Anaconda [2].

6.2.1 LSTM mit Dense Layer

Im ersten Ansatz soll geprüft werden, wie gut die Vorhersagen eines Netzes mit LSTM Schicht, kombiniert mit einem Feed-Forward Netz sind. Als Vorbild dient das Netz von Kumbhar und Bhandari [42], welches für die Emotionserkennung mithilfe von MFCC Features vorgesehen ist. Für die binäre Klassifizierung wird eine LSTM Schicht mit 128 Zellen gewählt. Auch 32, 64 und 256 Zellen wurden getestet, allerdings mit schlechteren Resultaten. Danach folgen mehrere Dense Layer [42]. Die Neuronen innerhalb dieser Schichten besitzen keine Verbindungen untereinander. Sie sind zu allen Neuronen in der nächsten Schicht verbunden. Diese Feed-Forward Architektur soll die Ausgaben der 128 LSTM Zellen sukzessiv reduzieren und zu einer Vorhersage führen. Für die binäre Klassifizierung wird eine Ausgabeschicht mit 1 Ausgang gewählt. Die *Sigmoid* Funktion dient als Aktivierungsfunktion, welche dafür sorgt, dass die Ausgabe zwischen 0 und 1 liegt [39, 62]. Werte unter 0,5 werden somit als *nicht-destruktive* Interaktionen gewertet und höhere Werte werden als *destruktiv*. Zusätzlich werden zwischen allen Schichten Dropout Layer eingefügt, um Overfitting zu

vermeiden. Zuerst lagen zwischen LSTM und Ausgabeschicht 5 Dense Layer, welche abwechselnd *Tanh* und *ReLU* als Aktivierungsfunktion nutzten. Diesen Ansatz verfolgten auch Kumbhar und Bhandari [42], allerdings konvergierte das Netz in dieser Arbeit nicht. Das bedeutet, die Verlustrate wurde über mehrere Epochen (Durchläufe) nicht geringer. Ein Grund für dieses Verhalten kann die Tiefe des Netzes in Kombination mit der *ReLU* Aktivierungsfunktion sein. Wahrscheinlich kommt es zu einem Problem namens "Dying ReLU", welches eine Verbesserung der *ReLU* Schichten während des Lernprozesses verhindert [46]. Aus diesem Grund wurden die *ReLU* Schichten entfernt, wodurch das Problem behoben werden konnte. Ein schematischer Aufbau dieses *LSTM+Dense* Netzes ist in Anhang D zu sehen.

Das binäre Netz wird sowohl mit originalem Klassenverhältnis, als auch mit dem 50:50 Klassenverhältnis trainiert. Die Fehlerfunktion beim Training mit dem 3:97 Datensatz wird mit dem umgekehrten Klassenverhältnis gewichtet, um *destruktive* Interaktionen stärker im Verlustwert zu berücksichtigen. Ohne diese Anpassung besteht die Gefahr, dass das Netz alle Interaktionen als *nicht-destruktiv* klassifiziert, um den Verlust möglichst gering zu halten.

Das Netz für die Multi-Class Klassifizierung ist ähnlich aufgebaut. Die besten Ergebnisse konnten jedoch mit 64 statt 128 LSTM Zellen erzielt werden. Ein weiterer Unterschied besteht darin, dass sechs Ausgaben in Form eines Vektors für die einzelnen Klassen gemacht werden müssen. Dazu dient eine Schicht mit der *Softmax* Aktivierungsfunktion, welche eine Wahrscheinlichkeit für jede Klasse ausgibt wie bereits in Tabelle 6.2 dargestellt [62].

Beide Netze verwenden den Adam Optimierungsalgorithmus mit einer Lernrate von 0,0001. Die binäre Klassifizierung nutzt die *binary cross-entropy* Verlustfunktion (Loss Function), welche standardmäßig für binäre Verfahren angewendet wird [4, 35]. Das Netz für die Multi-Class Klassifizierung verwendet die *categorical cross-entropy* Verlustfunktion, welche sich für die *Softmax* Ausgabeschicht eignet [53]. Nach jeder Epoche werden der Verlust (Loss) und die Metriken Balanced-Accuracy, Precision und Recall für den Trainings- und Validierungsdatensatz aufgezeichnet. Das Training läuft für maximal 1000 Epochen. Es wird allerdings schon vorher abgebrochen, falls sich der Loss des Validierungsdatensatzes über 200 Epochen nicht verbessert. Das beste Netz bezogen auf den niedrigsten Validierungsverlust wird jeweils gespeichert. Dies geschieht um Overfitting frühzeitig zu erkennen und abzubrechen. Overfitting kann durch eine fortsetzende Verbesserung der Trainingsergebnisse und eine gleichzeitige Verschlechterung der Validierungsergebnisse erkannt werden. Das Netz tendiert in diesem Fall dazu, die Trainingsdaten auswendig zu lernen und klassifiziert unbekannte Daten immer schlechter.

6.2.2 BiGRU mit Attention Layer

Die zweite Netzarchitektur basiert hauptsächlich auf der Arbeit von Kim und Lee [35], deren Netz schon erfolgreich für die Sentimentanalyse anhand von Video- und Audio- und Textfeatures genutzt wurde. Das Netz verwendet einen Bidirectional GRU Layer. Dieser Layer verwendet zwei entgegengesetzte Verkettungen von GRU Zellen. Eine GRU Schicht kann somit auf vorherige Zeitschritte und die Andere auf zukünftige Schritte einer Sequenz zugreifen. Es wurden auch verschiedene Schichtbreiten der GRU Layer getestet, wobei 32 und 128 Zellen pro Schicht die besten Ergebnisse für die binäre, beziehungsweise Multi-Class Klassifizierung, ergaben. Die Ausgaben der beiden Schichten werden konkateniert und an einen Attention Layer weitergegeben. Dieser Layer sorgt wie ein Filter dafür, dass einige Daten stärker und andere schwächer gewertet werden. Die Ausgabedaten des Layers werden mit den ursprünglichen Ausgabedaten der BiGRU Schicht addiert [35]. Anstatt die Ausgaben direkt über eine *Tanh* Aktivierungsfunktion weiterzugeben, wird in dieser Arbeit aus praktischen Gründen ein Dense Layer genutzt, welcher die *Tanh* Aktivierungsfunktion nutzt. Die Ausgaben für die binäre und Multi-Class Klassifizierung erfolgen wie in Abschnitt 6.2.1 beschrieben, über die *Sigmoid* beziehungsweise *Softmax* Aktivierungsfunktion. Dropout Schichten werden zwischen allen Schichten eingesetzt um Overfitting zu verringern [35, 4]. Die sonstigen Trainingsparamter sind mit denen des *LSTM+Dense* Netzes aus Abschnitt 6.2.1 identisch. Der schematische Aufbau des *BiGRU* Netzes ist in Anhang D zu sehen.

Kapitel 7

Evaluation

In diesem Kapitel werden die in Kapitel 5 und 6 trainierten Modelle mithilfe von Testdaten evaluiert. Die Auswertung der Ergebnisse und der Vergleich unterschiedlicher Klassifikatoren wird anhand der in Kapitel 2.6 erklärten Metriken durchgeführt. Die Auswahl der Metriken erfolge durch die Definition mehrerer Ziele, welche ein Klassifizierer optimalerweise erfüllen sollte. Zum einen sollen möglichst viele *destruktive* Interaktionen auch als *destruktiv* klassifiziert werden. Das Verhältnis zwischen den True-Positives und allen *destruktiven* Interaktionen wird durch den Recall gemessen. Ein weiteres Ziel ist, dass möglich wenig *nicht-destruktive* Interaktionen als *destruktiv* also False-Positives klassifiziert werden. Aus diesem Grund wird neben dem Recall auch die False-Positive Rate betrachtet. Precision bestimmt das Verhältnis zwischen True-Positives und allen *destruktiv* klassifizierten Daten (TP+FP). F1 wird aus Vollständigkeitsgründen ebenfalls erhoben. Jedoch können Precision und somit auch F1 durch die starke Imbalance des Klassenverhältnisses beeinflusst werden. Um eine Gesamtaussage über die Performanz der Modelle zu erhalten, wird neben den genannten Metriken auch Balanced Accuracy genutzt. Diese Metrik lässt sich vor allem auf unausgewogene Datensätze anwenden und liefert Informationen über die Erkennungrate der positiven und negativen Klasse. Sie hilft bei der Bestimmung des besten Klassifizierers und der Einschätzung der Gesamtqualität. Sowohl auf die binären als auch Multi-Class Klassifizierer werden die genannten Ziele und Metriken angewendet.

Im ersten Abschnitt werden die klassischen Methoden evaluiert, danach folgen die rekurrenten neuronalen Netze. Im Anschluss erfolgt ein Vergleich zwischen den besten Klassifizierern aus beiden Methoden. Binäre und Multi-Class Klassifizierer werden jeweils separat evaluiert. Im letzten Abschnitt erfolgt eine Beschreibung des erstellten Software Prototyps.

7.1 Evaluation der klassischen Methoden

Binäre Klassifizierung

Innerhalb der binären Klassifizierung werden zwei Ansätze verfolgt, um mit dem unausgeglichenen Klassenverhältnis von 3:97 umzugehen. Einerseits erfolgte das Training mit einem Datensatz mit ausgeglichenem Klassenverhältnis, welcher durch Undersampling der überlegenden Klasse erzeugt wurde. Zudem existiert ein Validierungsdatensatz mit einem 50:50 Verhältnis, um vergleichen zu können, wie gut die trainierten Modelle auf das Trainingsverhältnis (Validierungsdatensatz) und auf das ursprüngliche Verhältnis (Testdatensatz) anwendbar sind.
Andererseits erfolgte in Methode 2 ein Training mit dem ursprünglichen Klassenverhältnis und anderen Hyperparamtern, um dem Ungleichgewicht der Klassen entgegenzuwirken. Vorteil dieser Methode ist, dass mehr *nicht-destruktive* Daten zum Lernen zur Verfügung stehen. Dies ist gleichzeitig auch der Nachteil dieser Methode, da das Klassenverhältnis die Klassifizierer beeinflusst. Die beiden Methoden werden in den folgenden Unterabschnitten genauer erklärt. Zudem ist ein Schema zur Erstellung der verwendeten Datensätze in Anhang B zu finden.

Methode 1: Ergebnisse mit ausgeglichenem Trainingsdatensatz

Für das ausgeglichene Training wird der ursprüngliche Datensatz mit einem Teilungsverhältnis von 80:20 in Trainings- und Testdatensatz geteilt. Danach erfolgt Undersampling auf dem Trainingsdatensatz. Anschließend wird dieser in Validierungs- und neuen Trainingsdatensatz unterteilt. Beide besitzen das gleiche Klassenverhältnis von 50:50. Alle Datensätze werden anhand der Trainingsdaten skaliert. Das Training wird einmal durchgeführt. Danach folgen Validierung und Test mit den entsprechenden Datensätzen. Unter *Methode 1* in Anhang B ist dieser Prozess dargestellt.

	F1	Precision	Recall	FP Rate	Balanced Acc.
Naive Bayes	0,49	**0,69**	0,39	**0,18**	0,60
SVM	0,64	0,64	**0,65**	0,37	0,64
Rand Forest	**0,65**	0,67	0,63	0,31	**0,66**
Log Reg	0,63	0,64	0,62	0,36	0,63

Tabelle 7.1: Ergebnisse 50:50 Klassifikator mit Validierungsdatensatz

Tabelle 7.1 zeigt die Validierungsergebnisse der vier klassischen Klassifikatoren. SVM, Random Forest und Logistic Regression weisen bessere Werte beim F1-Score, Recall und Balanced Accuracy Wert vor als Naive

Bayes. Naive Bayers besitzt die nierdigste False-Positive Rate von allen Klassifikatoren mit 18%, allerdings auch den niedrigsten Recall mit 39%. Das bedeutet, im Vergleich mit den anderen Klassifizierern werden weniger False-Positives klassifiziert, aber auch generell weniger *destruktive* Interaktionen erkannt. Daraus folgt die höchste Genauigkeit (Precision) von 69%, jedoch auch die niedrigste Balanced Accuracy von 60% unter allen Klassifikatoren. Die SVM hat den höchsten Recall mit 65%, allerdings auch die höchste False-Positive Rate (37%). Die Logistic Regression weist ähnliche Ergebnisse auf. Die Ergebnisse des Random Forests unterscheiden sich nur geringfügig von denen der SVM, allerdings wirkt sich die um 6% niedrigere False-Positive Rate positiv auf die Balanced Accuracy aus, welche mit 66% die Höchste unter allen Klassifkatoren ist.

	F1	Precision	Recall	FP Rate	Balanced Acc.
Naive Bayes	0,10	**0,06**	0,35	**0,19**	0,58
SVM	**0,11**	**0,06**	**0,71**	0,39	**0,66**
Rand Forest	**0,11**	**0,06**	0,64	0,35	0,64
Log Reg	0,10	0,05	0,61	0,37	0,63

Tabelle 7.2: Ergebnisse 50:50 Klassifikator mit Testdatensatz

Die Tabelle 7.2 zeigt die Ergebnisse der Klassifikatoren mit dem Testdatensatz. Dieser besitzt das originale Klassenverhältnis von 3:97 und beinhaltet somit viel weniger *destruktive* als *nicht-destruktive* Interaktionen. Als zusammengefasste Erkennungsrate wird wieder Balanced Accuracy genutzt, welche im Gegensatz zu Tabelle 7.1 nicht der normalen Accuracy entspricht, da kein 1:1 Klassenverhältnis vorliegt. Der Balanced Accuracy Score aller Klassifikatoren unterschiedet sich nicht maßgeblich von den Validierungsergebnissen aus Tabelle 7.1. Bei Naive Bayes und SVM liegt dieser sogar um 2% höher. Auch Recall und False-Positive Rate der Testergebnisse verhalten sich ähnlich zu den Validierungsergebnissen. Diese Metriken werden mithilfe mit Daten innerhalb einer Klasse erhoben, sodass sich dort nur minimale Unterschiede zum ausgeglichenen Validierungsdatensatz ergeben. Große Abweichungen entstehen bei Metriken, welche Daten aus beiden Klassen (*destruktiv* und *nicht-destruktiv*) miteinander vergleichen. Somit ist der Precision Wert um 40-50% niedriger als vorher. Die False-Positive Rate hat sich im Vergleich mit den Validierungsergebnissen kaum verändert, allerdings werden dadurch absolut betrachtet mehr False-Positives klassifiziert als True-Positives, da 97% der Daten *nicht-destruktiv* sind. Daraus folgt bei ähnlichem Recall, dass die Genauigkeit (Precision) sehr gering ausfällt, weil die False-Positives überwiegen.
Es lässt sich feststellen, dass die Klassifikatoren auch mit unbekannten Daten umgehen können und auch eine unterschiedliche Klassengewichtung wenig

Einfluss auf Recall und Balanced Accuracy hat, jedoch wird die Precision ungenauer. Gemessen an der Balanced Accuracy, liefern SVM und Random Forest die besten Ergebnisse.

Methode 2: Ergebnisse mit unausgeglichenem Trainingsdatensatz

Für das unausgewogene Training wird, wie in Methode 1 aus Abschnitt 7.1, ein Trainings- zu Testdaten Verhältnis von 80:20 genutzt. Die Testdatensätze der beiden Methoden sind dabei identisch. Eine zusätzliche Unterteilung in Trainings- und Validierungsdatensatz wird nicht genutzt, da das Klassenverhältnis im Training und beim Testen das Gleiche ist. Unter *Methode 2* in Anhang B ist auch dieser Prozess dargestellt.

	F1	Precision	Recall	FP Rate	Balanced Acc.
Naive Bayes	0,09	0,06	0,20	0,10	0,55
SVM	**0,14**	**0,11**	0,19	**0,05**	0,57
Rand Forest	0,11	0,06	0,37	0,19	0,59
Log Reg	0,11	0,06	**0,58**	0,31	**0,63**

Tabelle 7.3: Ergebnisse 3:97 Klassifikator mit Testdatensatz

Tabelle 7.3 zeigt die Ergebnisse der Klassifizierung der Testdaten. Die SVM hat mit 0,14 den höchsten F1 Wert. Alle anderen Modelle erreichen einen F1 Wert von 0,11, bis auf Naive Bayes mit 0,09. Precision ist bei der SVM ebenfalls am höchsten mit 0,11. Der Recall ist allgemein schlechter als bei Methode 1 mit einem Minimum von 19% (SVM) und einem Maximum von 58% (Logistic Regression). Dafür sind jedoch auch die False-Positive Raten deutlich geringer als im vorherigen Ansatz. Es lässt sich feststellen, dass die Klassifizierer durch das Training mit dem unausgewogenen Datensatz trotz Hyperparamteranpassung die *nicht-destruktive* Klasse in der Klassifizierung bevorzugen. Dafür sprechen Recall und False-Positive Rate, welche beide geringer sind als im Test von Methode 1. Die Ausnahme stellt die Logistic Regression da, welche ähnlich wie zuvor abschneidet. Alle anderen Klassifizierer weisen im Vergleich mit der ersten Methode schlechtere Ergebnisse auf. Die Ergebnisse von Recall und False-Positive Rate weisen die Tendenz auf, dass häufiger *nicht-destruktive* Vorhersagen getroffen werden.

Multi-Class Klassifizierung

Für das Training und Testen der Multi-Class Klassifikatoren wird das originale Verhältnis der Klassenverteilung genutzt. Der Datensatz besteht aus nur 740 Interaktionen, deshalb wurde sich gegen die Undersampling Methode entschieden, wodurch theoretisch nur noch 240 Interaktionen für

Training und Test zur Verfügung stehen würden. Für die Evaluation kommt 5-Fold-Cross-Validation zum Einsatz, somit beinhaltet der Testdatensatz in jedem der fünf Durchläufe 20% der Daten. Dadurch können alle Daten einmal als Testdaten genutzt werden, um ein präzises Ergebnis zu erhalten.

	F1 micro	FP micro	Balanced Accuracy
Multi Naive Bayes	0,36	0,13	0,31
Multi SVM	**0,49**	**0,10**	**0,36**
Multi Rand Forest	0,45	0,11	0,33
Multi Log Reg	0,42	0,12	0,29

Tabelle 7.4: Ergebnisse Multi-Class Klassifikator mit Testdatensatz

In Tabelle 7.4 sind die Ergebnisse der Kreuzvalidierung für die Multi-Class Klassifizierung zu sehen. Da das Klassenverhältnis nicht ausgeglichen ist, wird für F1, Recall und Precision jeweils der Micro-Average Wert anstatt des Macro Average Werts genutzt. Dieser Wert ist für alle drei Metriken identisch. Zudem wird die Erkennungsrate der Modelle über den Balanced Accuracy Score gemessen. Die SVM hat den höchsten F1-/Recall-/Precision-Score mit 0,49, die niedrigste False-Positive Rate mit 0,10 und die beste Balanced Accuracy von 36%.

Die Konfusionsmatrix C.2 zeigt, dass besonders die Samples mit den Klassen *Seit* und *Unt* öfter korrekt klassifiziert werden als andere. Das ist vor allem damit zu erklären, dass fast 70% der Samples diesen Klassen angehören.

7.2 Evaluation rekurrenter neuronalen Netze

Binäre Klassifizierung

In Abschnitt 7.1 wurden bereits die zwei Trainingsmethoden der binären Klassifizierung vorgestellt. Diese Ansätze werden auch für die rekurrenten neuronalen Netze evaluiert. In der ersten Methode erfolgt das Training mit dem 50:50 Undersampling Datensatz. Die zweite Methode nutzt den 3:97 Trainingsdatensatz. Getestet werden die Netze jeweils mit dem gleichen 3:97 Testdatensatz. Die erste Methode nutzt außerdem einen Validierungsdatensatz (50:50), um zu überprüfen, wie sich Validierungs- und Testergebnisse durch das unterschiedliche Klassenverhältnis unterscheiden. Die Datensatzerstellung erfolgt mit einem anderen Ausgangsdatensatz als bei den klassischen Methoden. Jedoch ist das Verhältnis der Datensätze und das weitere Vorgehen gleich und kann in Anhang B eingesehen werden.

Methode 1: Ergebnisse mit ausgeglichenem Trainingsdatensatz

	F1	Precision	Recall	FP Rate	Balanced Acc.
LSTM+Dense	0,04	0,50	0,02	**0,02**	0,50
BiGRU	**0,56**	**0,56**	**0,55**	0,44	**0,56**

Tabelle 7.5: Ergebnisse Methode 1: RNNs mit 50:50 Validierungsdatensatz

Die Tabelle 7.5 zeigt die Ergebnisse der trainierten rekurrenten Netze mit dem 50:50 Validierungsdatensatz. Im Training wurde festgestellt, dass die anfängliche Fehlerrate des *LSTM+Dense* Netzes (rot markiert) kaum über die Epochen verbessert werden konnte. Daraus folgt, dass fast alle Samples der nicht-destuktiven Klasse zugeordnet werden. Das Netz kann nicht dazulernen und konvergiert somit nicht. Dies ist ebenfalls am Verlust-Epochen Diagramm D.2 zu erkennen. Die Verlustfunktion des Trainingsdatensatzes verbessert sich innerhalb von 200 Epochen minimal um 0,02, während der Validierungsverlust ansteigt. In der Diskussion in Kapitel 8 wird auf mögliche Ursachen dieses Problems eingegangen. 96% der Validierungsdaten werden als *nicht-destruktiv* klassifiziert, somit fallen Recall und False-Positive Rate mit 0,02 sehr gering aus. Precision liegt bei 50%, Recall und False-Positive Rate sowie das Klassenverhältnis sind gleich. F1 liegt bei 0,04 und die Balanced Accuracy bei genau 50%, weil die Hälfte der Daten richtig klassifiziert werden. Das Netz schneidet ähnlich ab, wie eine zufällige Klassifizierung (Raten). Aufgrund des fehlgeschlagenen Trainings wird das *LSTM+Dense* Netz in Methode 1 nicht weiter bewertet.
Das *BiGRU* Netz lies sich erfolgreich trainineren und klassifiziert die Validierungsdaten mit einer False-Positive Rate von 0,44 und einem Recall von 0,55. F1, Precision und Balanced Accuracy weisen jeweils einen Wert von 0,56 auf. Da Recall und False-Positive Rate nah an 50% liegen, kann daraus geschlossen werden, dass das Modell Schwierigkeiten hat, zwischen *destruktiven* und *nicht-destruktiven* Interaktionen zu unterschieden.

	F1	Precision	Recall	FP Rate	Balanced Acc.
BiGRU	**0,07**	**0,04**	**0,56**	**0,44**	**0,56**

Tabelle 7.6: Ergebnisse Methode 1: RNN mit 3:97 Testdatensatz

Tabelle 7.6 stellt die Ergebnisse der rekurrenten neuronalen Netze auf dem 3:97 Testdatensatz dar, welche zuvor mit einem 50:50 Datensatz trainiert wurden. Das *LSTM+Dense* Netz wird nicht auf den Testdaten evaluiert, da es im Training nicht konvergierte.

Die Testergebnisse des *BiGRU* Netzes sind ähnlich der Validierungsergebnisse. Der Recall liegt einen Prozent höher und die False-Positive Rate ist gleich. Das Klassenungleichgewicht sorgt hier für einen deutlichen Abfall der Precision um 0,52. Daraus folgt auch eine Verschlechterung des F1 Wertes auf 0,07. Die Balanced Accuracy liegt wie zuvor bei 56%.

Methode 2: Ergebnisse mit unausgeglichenem Trainingsdatensatz

	F1	Precision	Recall	FP Rate	Balanced Acc.
LSTM+Dense	0,08	**0,07**	0,10	**0,04**	**0,53**
BiGRU	**0,09**	**0,07**	**0,11**	**0,04**	**0,53**

Tabelle 7.7: Ergebnisse 3:97 Klassifikator mit Testdatensatz

Das Training beider Netze mit dem unausgeglichenen Datensatz verlief erfolgreich, sodass beide Netze konvergierten. Das Training erfolgte mithilfe einer gewichteten Verlustfunktion. Die Testergebnisse werden in Tabelle 7.7 dargestellt. Beide Netze erzielen ähnliche Ergebnisse. Die False-Positive Rate liegt jeweils bei 0,04. Dies bedeutet, dass sehr wenig False-Positives klassifiziert werden. Jedoch ist der Recall der Netze bei 0,1 und 0,11. Es werden somit nur 10% beziehungsweise 11% aller *destruktiven* Interaktionen korrekt klassifiziert. Die Precision beider Netze liegt bei 7%. F1 ist ebenfalls mit 0,08 beziehungsweise 0,09 sehr ähnlich. Die Ergebnisse zeigen, dass die *nicht-destruktive* Klasse einen starken Einfluss auf die Klassifizierung der Daten hat. Die Gewichtung der Verlustfunktion beider Netze hat einen eher geringen Einfluss auf die Klassifizierung. Der Balanced-Accuracy Wert von jeweils 53% zeigt, dass die Klassifizierung der Testdaten sehr unpräzise ist. Außerdem kann kein deutlicher Unterschied zwischen den Ergebnissen der beiden Architekturen festgestellt werden. Die Verlust-Epochen Diagramme D.4 und D.5 weisen ebenfalls darauf hin und zeigen, dass beide Netze durch die sinkenden Verlusteraten lernfähig erscheinen. In Kapitel 8 wird darauf genauer eingegangen.

Multi-Class Klassifizierung

Die Auswertung der Multi-Class Klassifizierung basiert auf einer Trainings- zu Testdaten Aufteilung von 80:20. Es wird wie in Abschnitt 7.1 der klassischen Methoden kein Undersampling für das Training genutzt. Eine Kreuzvalidierung wird nicht genutzt, da das Training der Netze erheblich länger dauert als das Training der klassischen Methoden.

	F1 micro	FP micro	Balanced Accuracy
Multi LSTM+Dense	0,39	**0,11**	0,17
Multi BiGRU	**0,44**	0,12	**0,28**

Tabelle 7.8: Ergebnisse Multi-Class Klassifikator mit Testdatensatz. F1 und False-Positive Werte sind jeweils Micro-Average Ergebnisse.

Die Tabelle 7.8 zeigt die Testergebnisse der trainierten Modelle. Es handelt sich beim F1 und False-Positive Wert um einen Micro-Average. Somit sind F1, Recall und Precision Wert identisch. Die False Positive Rate beider Klassifizierer liegt für dieses Multi-Class Problem bei 0,11 beziehungsweise 0,12. F1, Precision und Recall sind beim *Multi BiGRU* Netz mit 0,44 um 0,05 höher als beim *Multi LSTM+Dense* Netz. Die *Multi BiGRU* Balanced Accuracy unterschiedet sich noch deutlicher mit 28% zu 17%. Die Konfusionsmatrix D.7 des *Multi LSTM+Dense* Netzes zeigt, dass alle Testdaten einer Klasse zugeordnet wurden. Dadruch erklärt sich der Balanced Accuracy Wert von 17% (ungefähr $100/6 Klassen = 16,\overline{6}$). Die Verluste pro Epoche in Diagramm D.6 zeigen, dass der Validierungsverlust nach ungefähr 100 Epochen wieder zunimmt und auch der Trainingsverlust kaum abnimmt. Die Konfusionsmatrix D.9 des *Multi BiGRU* Netzes zeigt eine erhöhte Erkennungsrate der beiden häufigsten Klassen *Seit* und *Unt*, welche sich auch im Balanced-Accuracy Wert widerspiegeln. Dennoch ist zu beobachten, dass alle anderen Klassen meistens falsch klassifiziert werden. Das Loss-Epochen Diagramm D.8 zeigt deutliches Overfitting nach wenigen Epochen, da der Trainingsverlust stetig abnimmt, während der Validierungsverlust zunimmt.

7.3 Vergleich klassischer Methoden und RNNs

Es konnten sowohl klassische Methoden wie Naive Bayes, SVM, Random Forest und Logistic Regression als auch zwei rekurrente neuronale Netze trainiert und ausgewertet werden. Im abschließenden Vergleich werden die jeweils besten Modelle für die binäre und Multi-Class Klassifizierung ausgewählt. Verglichen wird mithilfe der am Anfang des Kapitels vorgestellten Metriken. Hauptziele sind eine geringe False-Positive Rate, sowie ein hoher Recall, damit möglichst wenig *nicht-destruktive* Interaktionen falsch klassifiziert werden und möglichst viele *destruktive* Interaktionen erkannt werden. Des Weiteren wird Balanced Accuracy hinzugefügt, um eine Gesamtaussage über die Performanz der Klassifizierer treffen zu können. Es werden nur Ergebnisse aus Testdatensätzen miteinander verglichen, da diese eine reale Situation simulieren.

7.3.1 Vergleich der binären Klassifikatoren

	F1	Precision	Recall	FP Rate	Balanced Acc.
SVM	**0,11**	**0,06**	**0,71**	0,39	**0,66**
Rand Forest	**0,11**	**0,06**	0,64	**0,35**	0,64
BiGRU	0,07	0,04	0,56	0,44	0,56

Tabelle 7.9: Vergleich der besten binären Klassifikatoren

Tabelle 7.9 zeigt die Auswahl der besten Klassifikatoren aus den jeweiligen Ansätzen. SVM und Random Forest erzielten die beste Kombination aus Recall, False-Positive Rate und Balanced Accuracy unter den klassischen Methoden. Die Konfusionsmatrizen der beiden klassischen Verfahren werden in Anhang C.1 dargestellt. Das *BiGRU* Netz erziehlt höhere Ergebnisse als das *LSTM+Dense* Netz und wird deshalb für den Vergleich ausgewählt. Alle Modelle wurden mit 50:50 Datensätzen trainiert. Es ist erkennbar, dass das neuronale Netz deutlich schlechtere Ergebnisse aufweist als die klassischen Methoden. Das Verlust-Epochen Diagramm D.3 zeigt, dass das Netz schnell zum Overfitting neigt, da der Verlust der Validierungsdaten zunimmt, während der Verlust der Trainingsdaten weiter abnimmt. Die SVM zeigt den höchsten Recall mit 71% gefolgt vom Random Forest mit 64%. Allerdings hat dieser eine um 4% bessere False-Positive Rate. Vergleicht man alle Metriken der Klassifizierer hat die SVM den höchsten Recall und die höchste Balanced Accuracy mit 66%. Daraus kann gefolgert werden, dass die SVM die beste Klassifizierung auf Grundlage des gegebenen Datensatzes erzielt.

7.3.2 Vergleich der Multi-Class Klassifikatoren

	F1 micro	FP micro	Balanced Accuracy
Multi SVM	**0,49**	**0,10**	**0,36**
Multi BiGRU	0,44	0,12	0,28

Tabelle 7.10: Vergleich der besten Multi-Class Klassifizierer

Die Multi-Class Klassifizierer werden in Tabelle 7.10 verglichen. Unter den klassischen Methoden erzielt die SVM die besten Ergebnisse in allen Metriken. Das *BiGRU* Netz hat eine 1% schlechtere False-Positive Rate als das *LSTM+Dense* Netz, allerdings liegen Recall und Balanced Accuracy deutlich höher. Im Vergleich zwischen *BiGRU* Netz und SVM fällt auf,

dass sich Recall und False Posititve Rate nicht signifikant voneinander unterscheiden. Dies ist darauf zurückzuführen, dass beide Klassifizierer viele Samples in die zwei häufigsten Klassen, *Seit* und *Unt*, eingeordnet haben und der Micro-Average diese beiden Klassen stärker wertet als andere. Die SVM weist jedoch eine um 8% höhere Balanced Accuracy von 36% auf. Insgesamt ist die SVM auch in der Multi-Class Klassifizierung der beste Klassifizierer.

7.4 Prototypische Implementierung

Die wichtigsten Ergebnisse dieser Arbeit, bezogen auf die Klassifizierung *destruktiver* Interaktionen, sollen in einem Software Prototyp zusammengefasst werden. Es wird ein Prototyp in Form einer Konsolenanwendung erstellt. Dieser nutzt die besten Klassifikatoren, um Interaktionen in Form von kurzen Audioausschnitten zu klassifizieren. Es wird davon ausgegangen, dass ein Audioausschnitt eine Interaktion enthält. Im ersten Schritt wird geprüft, ob es sich um eine *destruktive* Interaktion handelt. Ist dieser Test positiv, muss eine Kategorisierung in einen der sechs act4teams Codes erfolgen.

7.4.1 Implementierung von Klassifikatoren

Zuerst werden Klassifizierer für die binäre und die Multi-Class Klassifizierung des Prototyps benötigt. Die Auswahl beruht auf der Evaluation in diesem Kapitel und des Vergleichs der besten Klassifikatoren in Abschnitt 7.3. Sowohl in der binären als auch Multi-Class Klassifizierung erzielten SVMs die besten Ergebnisse. Diese Modelle erreichten im Vergleich sowohl den höchsten Recall als auch eine vergleichbar niedrige False-Positive Rate. Aus diesen Gründen werden für beide Klassifizierungsmethoden die Support Vector Machines genutzt. Der Prototyp ist wie alle weiteren Skripte dieser Arbeit in der Programmiersprache Python [67] geschrieben. Die bereits trainierten Modelle sind gespeichert und werden beim Programmstart geladen.

7.4.2 Nutzung der Software

Für die Nutzung der Software müssen einige Abhängigkeiten installiert werden. Die Installationsanleitung findet sich in Anhang E wieder. Die zu analysierende Audiodatei muss im WAV-Format vorliegen. Diese sollte nur eine sinnhaft geschlossene Aussage oder Interaktion enthalten. Der Prototyp benötigt beim Start den Dateipfad der zu klassifizierenden Audiodatei. Im ersten Schitt werden automatisch Features extrahiert. Danach erfolgt die Klassifizierung zwischen *destruktiver* und *nicht-destruktiver* Interaktion. Wird eine *destruktive* Interaktion erkannt, startet die Multi-Class Klassifizierung und ordnet die Interaktion in eine der sechs *destruktiven* Kategorien ein. Am Ende werden die entsprechenden Vorhersagen ausgegeben.

Kapitel 8

Diskussion

Der erste Teil dieses Kapitels ordnet die Ergebnisse der Evaluation in einen übergeordneten Kontext ein. Im zweiten Teil werden Limitierungen der Ansätze und ihre möglichen Ursachen genannt.

8.1 Vergleich mit naiven Klassifizierungsmethoden

Die erfolgreichsten Modelle für die Klassifizierung *destruktiver* Interaktionen wurden in Abschnitt 7.3 bestimmt. Mithilfe der trainierten Support Vector Machines konnten die besten Ergebnisse im Bezug auf die Evaluationsmetriken erzielt werden. Um die Qualität dieser Modelle einordnen zu können, erfolgt ein Vergleich der zwei *naiven Klassifizierungsmethoden*. Diese gehören nicht zu den maschinellen Lernmethoden. Der *Random Guess Classifier* klassifiziert Daten durch das zufällige Raten der Klassen. Das Raten dieser Methode erfolgt ohne Kenntnisse des Datensatzes, somit ist die Wahrscheinlichkeit für alle Klasse gleichverteilt. Der *Majority Class Classifier* ordnet alle Daten der größten Klasse zu. Diese naiven Methoden dienen als Ausgangspunkt („Baseline“) für die Einordnung der anderen Klassifizierer [21].

	F1	Precision	Recall	FP Rate	Balanced Acc.
Random Guess	0,06	0,03	0,5	0,5	0,5
Majority Class	0,0	0,0	0,0	**0,0**	0,5
SVM	**0,11**	**0,06**	**0,71**	0,39	**0,66**

Tabelle 8.1: Ergebnisse der binären Klassifizierung im Vergleich mit naiven Klassifizierern.

In Tabelle 8.1 sind die Ergebnisse Testergebnisse des 3:97 Datensatzes von den beiden naiven Klassifizierern und der SVM zu sehen. Der *Random Guess*

Classifier hat eine Precision von 3%, welche dem Anteil der Minderheitsklasse entspricht. Recall und False-Positive Rate liegen bei 50%. Es ergibt sich ein F1 Wert von 0,06 und eine Balanced Accuracy von 0,5, weil jeweils die Hälfte der Daten beider Klassen korrekt klassifiziert werden. Der *Majority Class Classifier* erreicht ebenfalls eine Balanced Accuracy von 50%. Da alle Interaktionen *nicht-destruktiv* klassifiziert werden, sind F1, Precision, Recall und False-Positive Rate jeweils 0,0.
Die SVM hat eine Precision von 0,06, einen Recall von 0,71 und somit einen F1 Wert von 0,11. Damit liegen die Ergebnisse jeweils höher als die der beiden naiven Methoden. Die False-Positive Rate liegt mit 39% zwischen den Ergebnissen der beiden anderen Klassifizierern und ist um 11% besser als die des *Random Guess Classifiers*. Die SVM erzielt eine Balanced Accuracy von 66% und ist somit 16% höher als die der naiven Methoden. Verglichen mit einer perfekten Klassifizerung von 100% ist dies keine große Steigerung. Grund dafür ist vor allem die hohe False-Positive Rate und die damit zusammenhängende niedrige Precision von 6%. Dennoch ist die SVM besser als die naiven Klassifikatoren, welche auf Methoden wie beispielsweise zufälligem Raten basieren.

	F1 micro	FP micro	Balanced Accuracy
Random Guess	0,17	0,17	0,17
Majority Class	0,39	0,12	0,17
Multi SVM	**0,49**	**0,10**	**0,36**

Tabelle 8.2: Binäre Klassifizierung im Vergleich mit naiven Klassifizierern

Tabelle 8.2 stellt die Ergebnisse der Kreuzvalidierung des Multi-Class Datensatzes von den beiden naiven Klassifizierern und der SVM dar. Der F1 Wert ist der Micro-Average aller F1 Werte der sechs Klassen. Somit ist dieser Wert identisch mit Recall und Precision. Die False-Positive Rate ist ebenfalls als Micro-Average angegeben. Die Multi-Class Klassifizierung fällt auf den ersten Blick schlechter aus als die binäre Klassifizierung. Die niedrigeren Ergebnisse (Balanced Accuracy, Recall) hängen jedoch mit der höheren Schwierigkeit des Problems zusammen. Die Wahrscheinlichkeit, eine korrekte Vorhersage zu machen, ist bei 6 Klassen deutlich geringer als bei zwei Klassen. Dies wird anhand der Ergebnisse der naiven Klassifikatoren deutlich.
Der *Random Guess Classifier* verteilt alle Samples gleichmäßig auf alle Klassen. Dadurch ergeben sich gleiche Werte für F1, Precision, Recall, False-Positive Rate und Balanced Accuracy von $1/6 = 0,1\overline{6} \approx 17\%$. Der *Majority Class Classifier* ordnet alle Daten der größten Klasse *Seit* (Seitengespräch) zu. Daraus ergibt sich ein Recall von 39% (Anteil der größten Klasse). Die False-Positive Rate liegt bei 0,12. Die Balanced Accuracy ist mit 17% genau

so hoch wie die des *Random Guess Classifier*. F1, Recall und Precision liegen bei der SVM um 0,10 höher. Außerdem hat sie den geringsten False-Positive Micro-Average mit 10%. Die Balanced Accuracy liegt mit 36% mehr als zweimal so hoch wie die der naiven Methoden. Anhand der Konfusionsmatrix C.2 kann beobachtet werden, dass die SVM die größten Klassen *Seit* und *Unt* besser als andere Klassen erkennt. Daraus wird jedoch deutlich, dass aussagekräftige Vorhersagen aller Klassen schwierig und unwahrscheinlich sind.

8.2 Limitierungen

Es wurde festgestellt, dass die Klassifizierer besser als naive Methoden funktionieren. Dennoch weisen die Metriken auf eine relativ unpräzise Klassifizierung hin. In den folgenden Anschnitten werden die verschiedenen Limitierungen dieser Arbeit zusammengefasst und mögliche Ursachen genannt.

8.2.1 Datensatz

Die Ausgangsdaten bilden die wichtigste Grundlage für die Klassifizierung. Ein großer Teil dieser Arbeit umfasst die Aufbereitung der verwendeten Rohdaten, bestehend aus Audioaufnahmen und den dazugehörigen Annotationsdateien. Die Qualität der Audioaufnahmen unterscheidet sich teilweise erheblich. Grund dafür sind unter anderem die Qualität und Position des Kameramikrofons, die Raumakustik und Umgebungsgeräusche. Es wurde versucht die Qualität der Aufnahmen im Nachhinein auszugleichen, jedoch können Informationen, die durch die Aufnahme verloren gingen, nicht wiederhergestellt werden. Ein weiteres Problem liegt in den Annotationsdateien, welche teilweise nicht präzise genug auf die Audiodateien referenzieren. Diese Qualitätsunterschiede der Audioaufnahmen und Annotationen führen zu einer Beeinflussung der Klassifizierung. Das Ausmaß der Auswirkungen kann nicht genau bestimmt werden, da kein Vergleichsdatensatz vorliegt.
Das Verhältnis zwischen *destruktiven* und *nicht-destruktiven* Interaktionen wirkt sich ebenfalls auf die Klassifizierung aus. Der beste binäre Klassifizierer weist einen relativ hohen Recall von 71% auf. Auffällig niedrig ist die Precision von nur 6%, welche in der Validierung bei 64% lag. Die Ursache für den Abfall kann primär durch die relativ hohe False-Positive Rate und das unausgewogene Klassenverhältnis des Testdatensatzes erklärt werden. Das Klassenverhältnis von 3:97 des binären Datensatzes entspricht jedoch realen Beobachtungen und muss zum Testen verwendet werden. Allerdings wird durch dieses Verhältnis die Schwierigkeit des Klassifizierungsproblems enorm erhöht. Die False-Positive Rate der meisten Klassifizierer von ungefähr 30-40% ist deutlich zu hoch, um anhand eines 3:97 Datensatzes präzise Vorhersagen treffen zu können. In der Multi-Class Klassifizierung ist das

Klassenvehältnis ebenfalls nicht ausgeglichen, allerdings besitzen die beiden Klassen *Seit* und *Unt*, sowie die Klassen *KI, TD, AL, J* untereinander ähnliche Verhältnisse. Die Datengrundlage fällt jedoch mit 854 *destruktiven* Interaktionen relativ gering aus, wodurch der Lernprozess erschwert wird.

8.2.2 Klassische Klassifikatoren

Innerhalb der binären Klassifizierung erreichte die 50:50 Trainingsmethode hohe Recall und False-Positive Werte. Die hohe False-Positive Rate kann durch das Undersampling des Testdatensatzes und dem damit vorhandenen *Bias* entstehen. Die 3:97 Trainingsmethode führte aufgrund der hohen Anzahl *nicht-destruktiver* Interaktionen zu jeweils sehr niedrigen Werten. Der Test zweier unterschiedlicher Ansätze hat gezeigt, dass die klassischen Methoden hier wahrscheinlich ihre Grenzen bezüglich der gegebenen Features erreicht haben. Bereits die PCA Visualisierung in 5.2.2 stellte dar, dass die Klassifizierungen eine hohe Komplexität aufweisen. Aufgrund dessen wurden rekurrente neuronale Netze mit einem neuen Featuredatesatz getestet.

8.2.3 Rekurrente neuronale Netze

Neuronale Netze sind theoretisch dazu in der Lage, tiefere Zusammenhänge in den Daten zu erkennen. Dadurch können sie auch klassische Methoden [4] übertreffen. Dies konnte jedoch mit den RNNs in dieser Arbeit, aufgrund mehrerer möglicher Ursachen, nicht gezeigt werden. Die Netze wurden nach dem Vorbild einiger bereits erfolgreicher Architekturen entworfen. Allerdings existieren eine Vielzahl weiterer Optimierungsmöglichkeiten, welche aufgrund des beschränken Umfangs dieser Arbeit nicht getestet werden konnten. Des Weiteren benötigen neuronale Netze im Vergleich zu klassischen Methoden viel mehr Lerndaten [4]. Die hohe Anzahl an Featuredaten ($431*20$ pro Interaktion) kann die Klassifizierung erschweren, wenn nicht genügend unterschiedliche Samples zur Verfügung stehen. Das Netz kann dann nicht ausreichend generalisieren und neigt zum Auswendiglernen, wie vor allem anhand der Verlust-Epochen Diagramme D.3 und D.8 beobachtet werden konnte. Die geringe Anzahl der *destruktiven* Samples kann einer der Gründe sein, weshalb die RNNs schlechter als die klassischen Methoden waren.
Es war zu beobachten, dass das *LSTM+Dense* Netz nicht mit dem 50:50 Trainingsdatensatz konvergierte. Ein Grund neben der geringen Datenmenge kann die einfache Architektur dieses Netzes sein, welche nicht so lernfähig ist wie das *BiGRU* Netz. Im 3:97 Training der RNNs performten die beiden Netze fast identisch mit deutlich sinkenden Verlust-Epochen Graphen. Dabei ist zu berücksichtigen, dass der 3:97 Datensatz die Klassifikatoren beeinflusst. Die sehr niedrigen False-Positive und Recall Werte bestätigen, dass beide Netze gelernt haben, die meisten Daten als *nicht-destruktiv* zu klassifizieren. Dieser Lernprozess wird in den Verlustdiagrammen D.4 und D.5 dargestellt.

Kapitel 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

Ziel dieser Arbeit war es, anhand von Audiosignalen *destruktive* Interaktionen in Entwicklermeetings zu erkennen und genauer einzuordnen. Qualitative Methoden wie Natural Language Processing sollten nicht genutzt werden. Audioaufnahmen aus 38 Entwicklermeetings dienten als Datengrundlage und wurden im Voraus von Psychologen nach dem act4teams Kodierungsschema analysiert und annotiert.
Die Audiodaten variierten in ihrer Qualität. Es erfolgte eine Aufbereitung der Daten mittels geeigneter Audiofilter und Anpassung an die Annotationen. Danach erfolgte eine Aufteilung der Audiospuren in einzelne Semgente mittels der Annotationen. Es konnte eine Vielzahl von Features aus den Daten extrahiert und und in Featuredatensätzen für die klassischen Lernmethoden und die rekurrenten neuronalen Netze bereitgestellt werden.
Im Anschluss erfolgte eine Auswahl mehrerer Klassifikatoren, darunter vier klassische Methoden und zwei rekurrente neuronale Netze. Die Grundlage dieser Auswahl bildeten wissenschaftlich Arbeiten im Gebiet der Audio-, Emotionen- und Sentimentanalyse. Die Klassifizierer wurden mit jeweils einem Teil der Datensätze traininert, validiert und getestet.
Die Evaluation zeigte, dass Support Vector Machines die besten Ergebnisse in der binäre und Multi-Class Klassifizierung liefern. Jedoch ergab die Diskussion der Ergebnisse, dass die Klassifizierer noch keine sicheren Vorhersagen liefern können. Mögliche Gründe dafür können unter anderem das stark unausgewogene Klassenverhältnis, die geringe Anzahl an *destruktiven* Daten zum Lernen und die hohe Komplexität des Problems sein. Mit den momentanen Modellen konnte gezeigt werden, dass eine grobe aber nicht aussagekräftige Erkennung und Einordnung von *destruktiven* Interaktionen anhand von Audiosignalen möglich ist.

9.2 Ausblick

Die in dieser Arbeit trainierten Klassifizierer können keine präzise Erkennung und Klassifizierung *destruktiver* Interaktionen vornehmen. Zukünftig gibt es mehrere Möglichkeiten diesen Ansatz zu verbessern.

Eine Möglichkeit besteht in der Sammlung weiterer Rohdaten für die Klassifizierung mittels rekurrenter neuronaler Netze. In dieser Arbeit konnte jeweils ein Meeting pro Team genutzt werden, welches während eines frühen Projektstadiums aufgezeichnet wurde. Der Anteil *destruktiver* Interaktionen fällt mit durchschnittlich 3% eher gering aus. Es stellt sich die Frage, wie sich der Anteil der *destruktiven* Interaktionen während der gesamten Projektdauer verändert. Hier besteht weiterer Forschungsbedarf.

Zudem können die neuronalen Netze zukünftig verbessert, angepasst und optimiert werden. Diese Arbeit konzentrierte sich nur auf rekurrente Neuronale Netze. Es existieren auch weitere Ansätze wie Convolutional Neural Networks (CNN), welche ebenfalls für die Sentiment- und Emotionserkennung genutzt werden [4]. CNNs benötigen jedoch wieder andere Repräsentationen von Eingabedaten und konnten im Umfang dieser Arbeit nicht näher betrachtet werden.

Des Weiteren ist es möglich die trainierten Klassifizierer mit anderen Ansätzen wie der Videoanalyse oder Natural Language Processing zu kombinieren. Der Abgleich mehrerer Methoden erhöht die Wahrscheinlichkeit einer korrekten Aussage. Die Arbeit von Obeidi [49] bildet hier eine passende Grundlage. Bisher eignet sich der erstellte Software Prototyp für die Analyse bereits segmentierter Interaktionsausschnitte. Dieser könnte auch in der Echtzeitanalyse genutzt werden, wenn vorher eine Aufteilung einer Audiospur in Audiosegmente erfolgt. Dafür kommt zum Beispiel das von Herrmann [26] entwickelte Tool infrage, welches in Echtzeitgespräche in einzelne Aussagen zerteilt.

In diesem Abschnitt wurde gezeigt, dass noch offene Fragen im Forschungsgebiet der automatischen Meetinganalyse existieren. Auch die Auswirkungen der verschiedenen Interaktionen müssen noch genauer untersucht werden. Diese Arbeit dient als eine Grundlage für die zukünftige Forschung im Gebiet der automatischen Interaktionserkennung und Analyse von Meetings.

Anhang A

Grundlagen

Destruktive Interaktionen

act4teams Code	Bedeutung
AE	Betonung autoritärer Elemente
AL	Phrase
E	Abbruch
J	Jammern
KI	Kein Interesse an Veränderungen
R	Reputation
S	Schuldigensuche
Seit	Seitengespräche
TD	Tadeln/Lästern
Unt	Unterbrechung

Tabelle A.1: act4teams Codes für *destruktive* Interaktionen. Die ausgegrauten Codes können in dieser Arbeit nicht genutzt werden, weil im verwendeten Datensatz zu wenige dieser Interaktionen vorhanden sind (siehe 4.3).

Aufbau Konfusionsmatrix

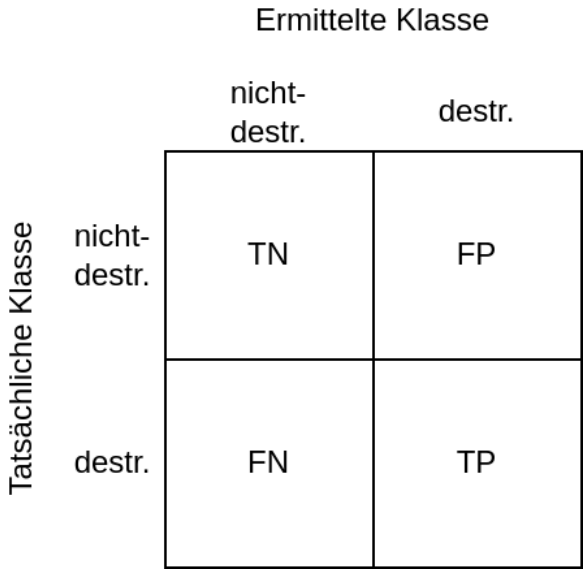


Abbildung A.1: Beispielhafter Aufbau einer Konfusionsmatrix anhand der beiden Klassen *destruktiv* und *nicht-destruktiv*.

Anhang B

Datensätze

2D Feature Visualisierung

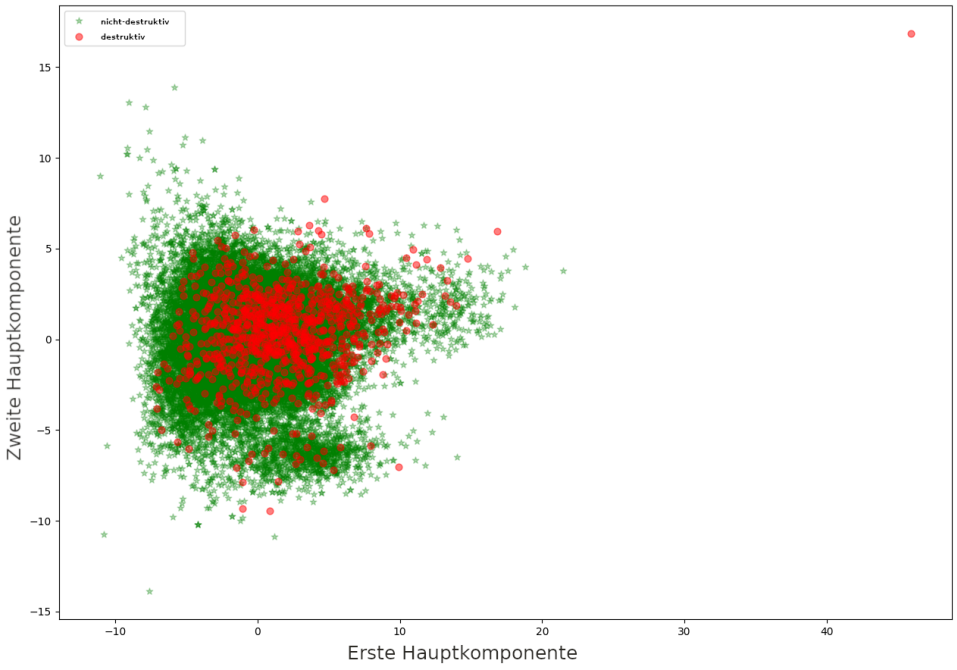


Abbildung B.1: 2D PCA Visualisierung des binären Datensatzes der klassischen Methoden.

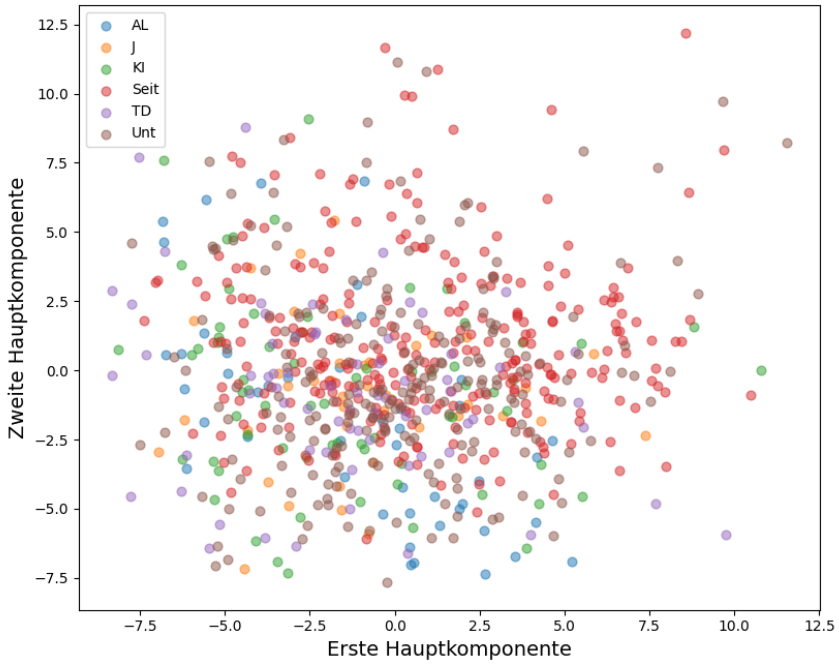
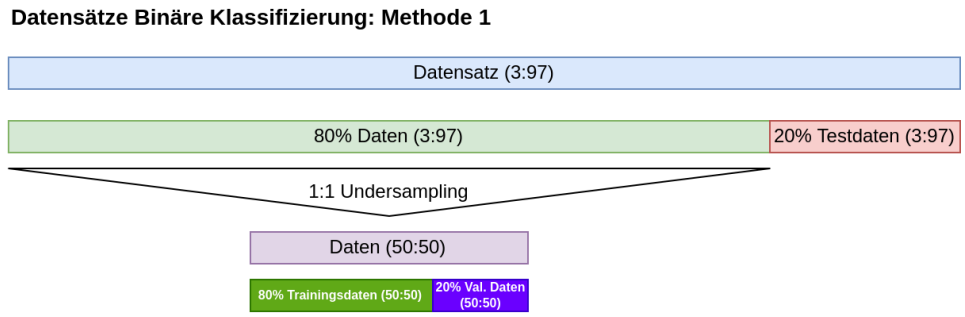


Abbildung B.2: 2D PCA Visualisierung des Multi-Label Datensatzes der klassischen Methoden.

Trainings-, Validierungs- und Testdatensätze



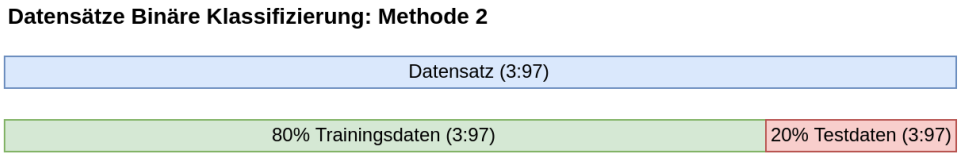


Abbildung B.3: Konzeptionelle Darstellung der Datensatzerstellung der zwei binären Klassifizierungsmethoden. Die Klassenverhältnisse sind in Klammern angegeben. In der Multi-Class Klassifizierung wird eine 80:20 Teilung wie in der zweiten Methode gewählt (die Klassenverhältnisse unterscheiden sich).

Anhang C

Klassische Methoden

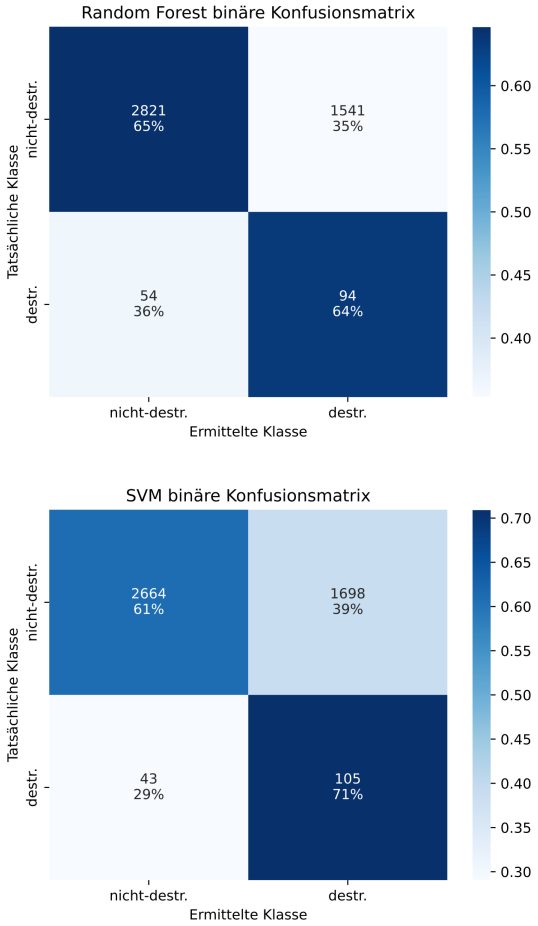


Abbildung C.1: Binäre Konfusionsmatrizen von *Random Forest* (oben) und *SVM* (unten).

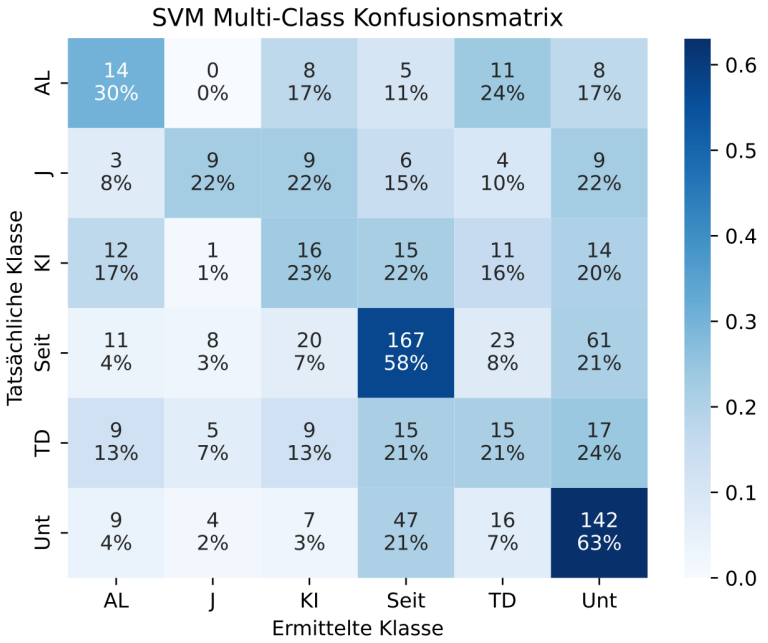


Abbildung C.2: Konfusionsmatrix der *Multi SVM*.

Anhang D

Rekurrente neuronale Netze

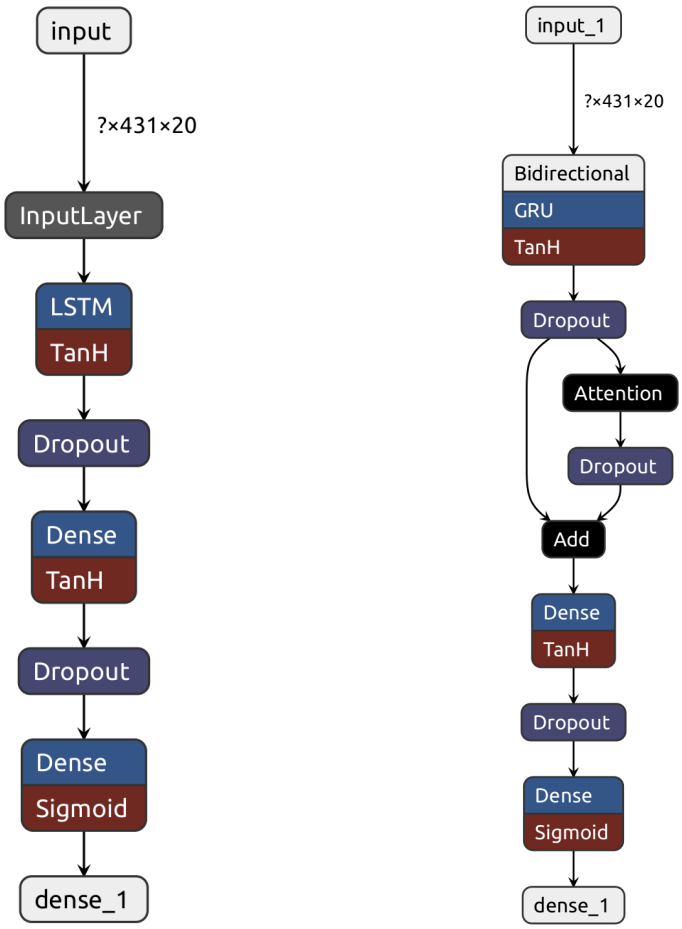


Abbildung D.1: *LSTM+Dense* Netz (links) und *BiGRU* Netz (rechts). Die Visualisierung zeigt den schematischen Aufbau der Netze für die binäre Klassifizierung.

Binäre Klassifizierung

Methode 1: 50:50 Training

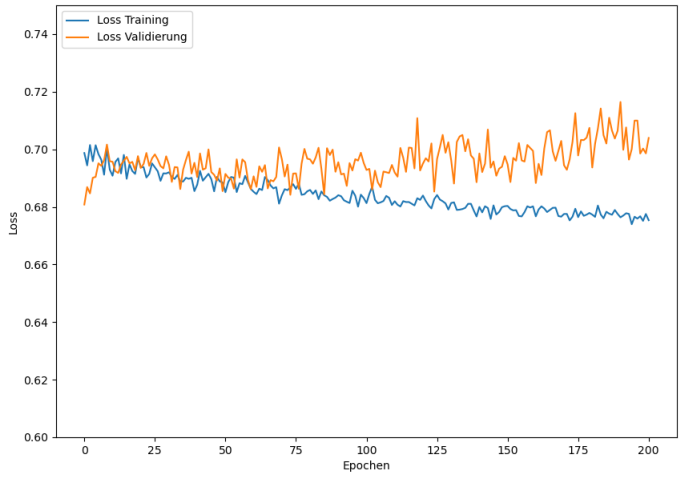


Abbildung D.2: Verlust-Epochen Diagramm des *LSTM+Dense* Netzes mit 50:50 Trainingsdatensatz.

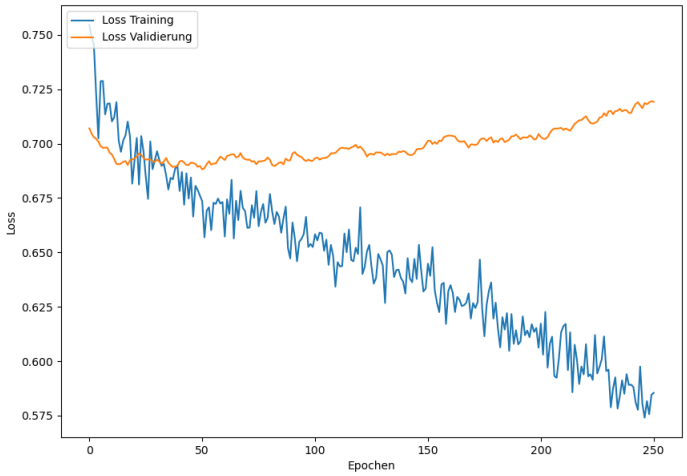


Abbildung D.3: Verlust-Epochen Diagramm des *BiGRU* Netzes mit 50:50 Trainingsdatensatz.

Methode 2: 3:97 Training

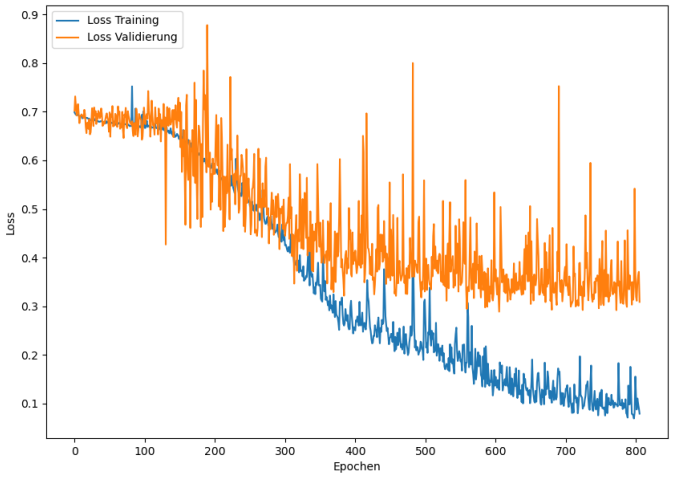


Abbildung D.4: Verlust-Epochen Diagramm des *LSTM+Dense* Netzes mit 3:97 Trainingsdatensatz.

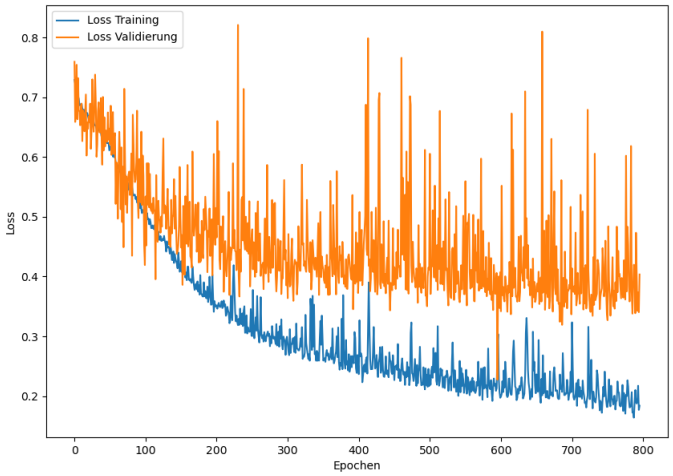


Abbildung D.5: Verlust-Epochen Diagramm des *BiGRU* Netzes mit 3:97 Trainingsdatensatz.

Multi-Class Klassifizierung

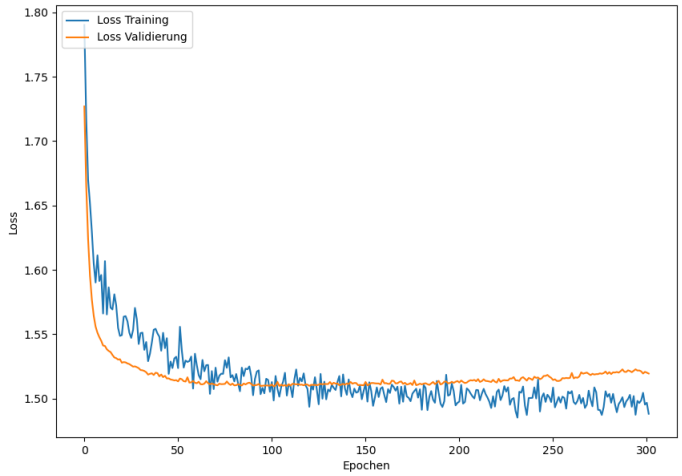


Abbildung D.6: Verlust-Epochen Diagramm des *Multi LSTM+Dense* Netzes.

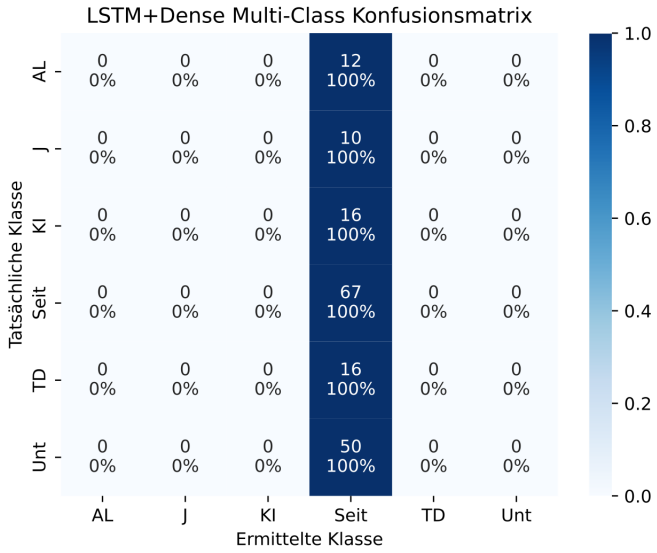


Abbildung D.7: Konfusionsmatrix des *Multi LSTM+Dense* Netzes.

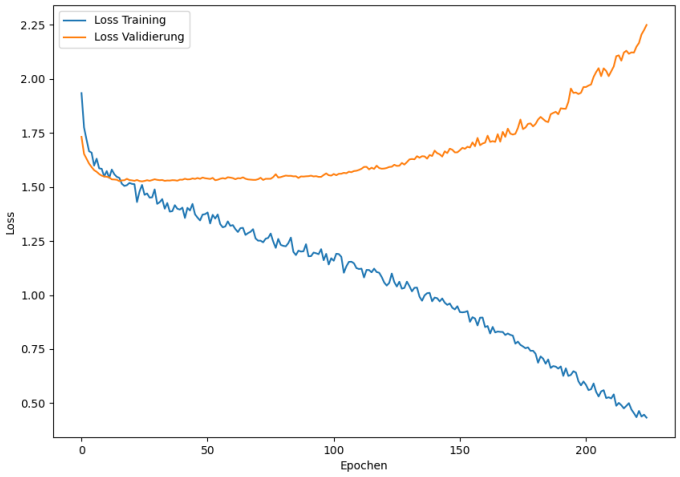


Abbildung D.8: Verlust-Epochen Diagramm des *Multi BiGRU* Netzes.

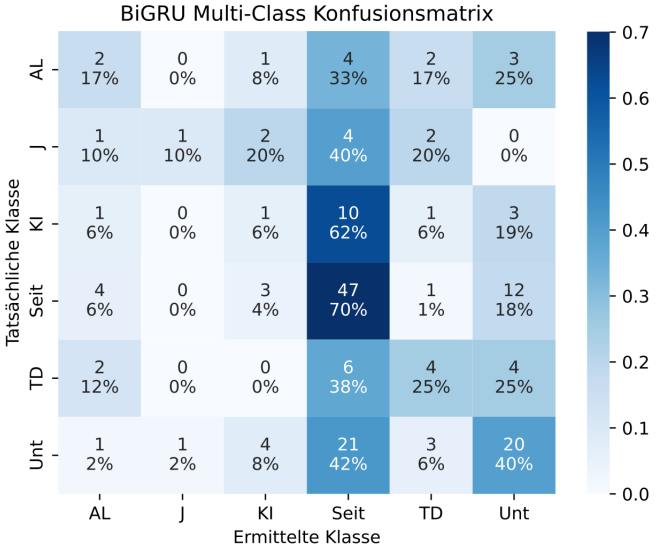


Abbildung D.9: Konfusionsmatrix des *Multi BiGRU* Netzes.

Anhang E

Installationsanleitung des Prototyps

Repository herunterladen

Das Repository **auto-team-interaction-recognition** kann von der SE GitLab Seite geklont oder heruntergeladen werden.

Öffnen Sie als nächstes das Repository:

```
$ cd auto-team-interaction-recognition/
```

Virtuelle Umgebung und Abhängigkeiten

Zuerst müssen folgende Programme installiert werden:

- Python (ab Version 3,8)
- Praat

Es wird empfohlen eine virtuelle Python Umgebung zu nutzen. Installation der virtuellen Umgebung mit allen benötigten Bibliotheken:

```
$ sudo apt install python3-pip
$ python3 -m pip install virtualenv
$ virtualenv .venv
$ source .venv/bin/activate
$ pip install -r requirements.txt
```

Prototyp bedienen

Zuerst muss das Verzeichnis des Prototyps geöffnet werden:

```
$ cd cli-prototype/
```

Der Prototyp wird über den folgenden Befehl gestartet:

```
$ python interactionAnalyser.py <FILENAME.wav>
```

Bitte ersetzen Sie <FILENAME.wav> durch eine entsprechende WAV-Datei, welche eine Interaktion enthalten sollte. Beispiele finden sich im aktuellen Ordner.
Der Prototyp analysiert die Datei und führt eine automatische Erkennung *destruktiver* Teaminteraktionen anhand von Audiosignalen durch. Eine Beispielausgabe könnte wie folgt aussehen:

```
$ python interactionAnalyser.py Team39_272_Unt.wav
Erkennung: Destruktive Interaktion erkannt.
Kategorie: Unterbrechung (74.2%)

Wahrscheinlichkeiten der Kategorien:

| Unt   | Seit  | TD    | KI    | AL    | J     |
-------------------------------------------------
| 74.2% | 14.8% | 3.8%  | 3.1%  | 2.4%  | 1.6%  |
```

Literaturverzeichnis

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Anaconda Software Distribution, 2020. Computer software. Vers. 2-2.4.0. Anaconda, Nov. 2016. Web. `https://docs.anaconda.com/`.

[3] Audacity-Team. Audacity Manual. `https://manual.audacityteam.org/man/index_of_effects_generators_and_analyzers.html`. letzer Zugriff: Januar 2022.

[4] M. E. Basiri, S. Nemati, M. Abdar, E. Cambria, and U. R. Acharya. Abcdm: An attention-based bidirectional cnn-rnn deep model for sentiment analysis. *Future Generation Computer Systems*, 115:279–294, 2021.

[5] M. Bekkar, H. K. Djemaa, and T. A. Alitouche. Evaluation measures for models assessment over imbalanced data sets. *J Inf Eng Appl*, 3(10), 2013.

[6] P. Boersma and V. Van Heuven. Speak and unspeak with praat. *Glot International*, 5(9/10):341–347, 2001.

[7] Brockhaus. Interaktion (Psychologie). In *Brockhaus Enzyklopädie Online*. NE GmbH | Brockhaus, 2022.

[8] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE, 2010.

[9] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, et al. API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

[10] F. Chollet et al. Keras. `https://keras.io`, 2015.

[11] K. Choo, E. Greplova, M. H. Fischer, and T. Neupert. *Maschinelles Lernen ohne neuronale Netzwerke*, pages 7–24. Springer Fachmedien Wiesbaden, Wiesbaden, 2020.

[12] M. S. Cole, F. Walter, and H. Bruch. Affective mechanisms linking dysfunctional behavior to performance in work teams: a moderated mediation study. *The Journal of applied psychology*, 93 5:945–58, 2008.

[13] P. P. Dahake, K. Shaw, and P. Malathi. Speaker dependent speech emotion recognition using mfcc and support vector machine. In *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, pages 1080–1084. IEEE, 2016.

[14] Z. M. Dan and F. S. Monica. A study about mfcc relevance in emotion classification for srol database. In *2013 4th International Symposium on Electrical and Electronics Engineering (ISEEE)*, pages 1–4. IEEE, 2013.

[15] N. H. De Jong and T. Wempe. Praat script to detect syllable nuclei and measure speech rate automatically. *Behavior research methods*, 41(2):385–390, 2009.

[16] T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[17] J. Frochte. *Entscheidungsbäume*, chapter 6, pages 117–160. Carl Hanser Verlag GmbH Co KG, 2019.

[18] J. Frochte. *Statistische Grundlagen und Bayes-Klassifikator*, chapter 4, pages 68–87. Carl Hanser Verlag GmbH Co KG, 2019.

[19] J. Frochte. *Support Vector Machines*, chapter 10, pages 286–303. Carl Hanser Verlag GmbH Co KG, 2019.

[20] R. Fu, Z. Zhang, and L. Li. Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328. IEEE, 2016.

[21] S. Gauher. Is your Classification Model making lucky guesses? `https://blog.revolutionanalytics.com/2016/03/classification-models.html`. letzer Zugriff: Januar 2022.

[22] M. Ghai, S. Lal, S. Duggal, and S. Manik. Emotion recognition on speech signals using machine learning. In *2017 international conference on big data analytics and computational intelligence (ICBDAC)*, pages 34–39. IEEE, 2017.

[23] A. Handl. *Hauptkomponentenanalyse*, pages 115–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[24] C. H. Hansen. Fundamentals of acoustics. *Occupational Exposure to Noise: Evaluation, Prevention and Control. World Health Organization*, pages 23–52, 2001.

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

[26] M. Herrmann. Automatische Klassifikation von Aussagen in Meetings von Entwicklungsteams. Bachelor's Thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2021.

[27] T. R. Hoens and N. V. Chawla. *Imbalanced Datasets: From Sampling to Classifiers*, chapter 3, pages 43–59. John Wiley & Sons, Ltd, 2013.

[28] J. Horstmann. Computer-gestützte Analyse des Kommunikationsverhaltens in Entwicklerteams unter Berücksichtigung digitaler Medien. Master's Thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2019.

[29] H. Hung and D. Gatica-Perez. Estimating cohesion in small groups using audio-visual nonverbal behavior. *IEEE Transactions on Multimedia*, 12(6):563–575, 2010.

[30] Ixnay. File:Gated Recurrent Unit.svg — Wikimedia Commons, the free media repository. `https://commons.wikimedia.org/w/index.php?title=File:Gated_Recurrent_Unit.svg&oldid=472324516`, 2017. letzer Zugriff: Januar 2022.

[31] Ixnay. File:Long Short-Term Memory.svg — Wikimedia Commons, the free media repository. `https://commons.wikimedia.org/w/index.php?title=File:Long_Short-Term_Memory.svg&oldid=488381453`, 2017. letzer Zugriff: Januar 2022.

[32] Ixnay. File:Recurrent neural network unfold.svg — Wikimedia Commons, the free media repository. `https:`

//commons.wikimedia.org/w/index.php?title=File:Recurrent_neural_network_unfold.svg&oldid=605590767, 2017. letzer Zugriff: Januar 2022.

[33] T. Johnstone and K. R. Scherer. The effects of emotions on voice quality. In *Proceedings of the XIVth international congress of phonetic sciences*, pages 2029–2032. Citeseer, 1999.

[34] S. Kauffeld and N. Lehmann-Willenbrock. Meetings Matter Effects of Team Meetings on Team and Organizational Success. *Small Group Research*, 43:130–158, 04 2012.

[35] T. Kim and B. Lee. *Multi-Attention Multimodal Sentiment Analysis*, page 436–441. Association for Computing Machinery, New York, NY, USA, 2020.

[36] K. K. Kishore and P. K. Satish. Emotion recognition in speech using mfcc and wavelet features. In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pages 842–847. IEEE, 2013.

[37] J. Klünder. *Analyse der Zusammenarbeit in Softwareprojekten mittels Informationsflüssen und Interaktionen in Meetings.* Dissertation, Gottfried Wilhelm Leibniz Universität Hannover, Berlin, 2019.

[38] J. Klünder, N. Prenner, A.-K. Windmann, M. Stess, M. Nolting, F. Kortum, L. Handke, K. Schneider, and S. Kauffeld. *Do You Just Discuss or Do You Solve? Meeting Analysis in a Software Project at Early Stages*, page 557–562. Association for Computing Machinery, New York, NY, USA, 2020.

[39] S. W. Knox. *Survey of Classification Techniques*, chapter 4, pages 33–95. John Wiley & Sons, Ltd, 2018.

[40] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[41] I. Kononenko and M. Kukar. Chapter 7 - data preprocessing. In I. Kononenko and M. Kukar, editors, *Machine Learning and Data Mining*, pages 181–211. Woodhead Publishing, 2007.

[42] H. S. Kumbhar and S. U. Bhandari. Speech emotion recognition using mfcc features and lstm network. In *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–3, 2019.

[43] R. Lackes. Definition: Informatik, Feb 2018. letzer Zugriff: Januar 2022.

[44] G. Lemaître, F. Nogueira, and C. K. Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[45] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory Undersampling for Class-Imbalance Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

[46] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

[47] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.

[48] C. Meyer. Verbesserung von evolutionären Algorithmen zur Klassifikation von schriftlicher Kommunikation in Entwicklungsteams. Master's Thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2020.

[49] Z. Obeidi. Automatisierte Erkennung von destruktiven Äußerungen in Meetings von Softwareprojekten. Master's Thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2021.

[50] Y. Pan, P. Shen, and L. Shen. Speech emotion recognition using support vector machine. *International Journal of Smart Home*, 6(2):101–108, 2012.

[51] R. Parmar. Better Dialogue: Compression for dialogues in films and documentaries. - Ronak Parmar. `https://ronakparmar.com/compression-for-dialogues/`, Apr. 2020. letzer Zugriff: November 2021.

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

[53] S. Poria, E. Cambria, D. Hazarika, N. Majumder, A. Zadeh, and L.-P. Morency. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 873–883, 2017.

[54] PreSonus. How To Use Dynamics Processing: Getting Started with Compressors, Gates, and More. `https://www.presonus.com/learn/technical-articles/How-To-Use-Dynamics-Processing-`

`Getting-Started-With-Compressors-Gates-and-More`. letzer Zugriff: November 2021.

[55] R. S. Pressman. *Software engineering: a practitioner's approach*, chapter 1, pages 1–13. Palgrave macmillan, 2005.

[56] W. Ramadhan, S. A. Novianty, and S. C. Setianingsih. Sentiment analysis using multinomial logistic regression. In *2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC)*, pages 46–49. IEEE, 2017.

[57] P. Refaeilzadeh, L. Tang, and H. Liu. Cross-validation. *Encyclopedia of database systems*, 5:532–538, 2009.

[58] C. Sammut and G. I. Webb, editors. *Cross-Validation*, pages 249–249. Springer US, Boston, MA, 2010.

[59] R. T. Sataloff. The human voice. *Scientific American*, 267(6):108–115, 1992.

[60] K. Schneider, J. Klünder, F. Kortum, L. Handke, J. Straube, and S. Kauffeld. Positive Affect through Interactions in Meetings: The Role of Proactive and Supportive Statements. *Journal of Systems and Software*, 143, 05 2018.

[61] T. Seehapoch and S. Wongthanavasu. Speech emotion recognition using support vector machines. In *2013 5th international conference on Knowledge and smart technology (KST)*, pages 86–91. IEEE, 2013.

[62] S. Sharma, S. Sharma, and A. Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[63] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig. Performance Evaluation of Deep Neural Networks Applied to Speech Recognition: RNN, LSTM and GRU. *Journal of Artificial Intelligence and Soft Computing Research*, 9(4):235–245, 2019.

[64] V. Shrivastava, V. Richhariya, and V. Richhariya. Puzzling out emotions: A deep-learning approach to multimodal sentiment analysis. In *2018 International Conference on Advanced Computation and Telecommunication (ICACAT)*, pages 1–6. IEEE, 2018.

[65] Y. Sun, A. K. Wong, and M. S. Kamel. Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence*, 23(04):687–719, 2009.

[66] S. Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006.

[67] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[68] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.