# Genauigkeitsschwankungen bei einer Sentimentanalyse mittels Naïve Bayes durch verschiedene Textverarbeitungs- und Textaufbereitungsmethoden

Daniela N. Pillwein


MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2015

# Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 28. September 2015

Daniela N. Pillwein

# Inhaltsverzeichnis

# Vorwort

Diese Masterarbeit entstand im Rahmen meines Masterprojektes – die Sentimentanalyse für kurze Nachrichten zu verbessern – des Studiengangs Interactive Media am Campus Hagenberg der Fachhochschule Oberösterreich. In diesem Projekt ging es um die Entwicklung beziehungsweise Verbesserung eines Klassifikations-Modelles für kurze Nachrichten, wie Tweets, welche in zwei verschiedene Kategorien, „positiv“ und „negativ“, eingeteilt werden sollen. Dabei sollte ein Klassifikations-Modell für die automatisierte Zuordnung in „positiv“ oder „negativ“ eingesetzt werden, welches unterschiedliche Wissensbasen für die Klassifikation kombiniert verwendet, und somit eine verbesserte Sentimentanalyse-Applikation geschaffen werden.

Mit Hilfe des „Naïve Bayes“ Verfahrens (ein maschinelles Lernverfahren), welches Wissensbasen mit Datensätzen, die einer Klasse zugeordnet sind, verwendet, um daraus zu lernen, soll dem Klassifikationsprozess Abhilfe geschaffen werden. Durch die Verwendung von unterschiedlichen Wissensbasen, insbesondere eine mit kurzen Nachrichten und eine weitere mit längeren Nachrichten (Filmkritiken oder in Folge auch Movie-Reviews genannt), sollen neue Möglichkeiten für eine bessere Klassifikation geboten werden.

Als besondere technische Herausforderung gilt das Entwickeln einer ausgereiften, intelligenten Computerlinguistik-Applikation, welche die beiden Klassifikations-Modelle optimal kombiniert. Um überhaupt die Wissensbasen für die Einstufung in positiv und negativ zu verwenden, müssen deren Datensätze vorher erst verarbeitet und aufbereitet werden. Diese Aufbereitung soll natürlich automatisiert werden, wobei eine ausführliche Textanalyse unumgänglich ist. Die aufbereiteten Texte dienen zunächst als Basis für die Klassifikations-Modelle. Alle Inhalte der verwendeten Wissensbasen sollten beziehungsweise müssen auf gleiche Art und Weise verarbeitet werden, da sonst eine Kombination der daraus entstandenen Modelle undenkbar ist. Es wird untersucht, wie sich unterschiedliche Textaufbereitungsmethoden auf die Qualität der Klassifikation auswirken und ob eine Kombination unterschiedlicher Modelle zu einer Verbesserung in Sachen Performance führt. Im Speziellen wird dabei auf die Verarbeitung der kurzen Nachrichten eingegangen, da diese von Twitter stammen und oftmals Kürzel, Slang, Hyperlinks, Eigennamen, Bilder oder Emoticons aufweisen. Darüber hinaus

wurde versucht, auf Hindernisse in diesem Bereich einzugehen. Bei der Verwendung von mehreren Wissensbasen stellt die optimale Kombination eine Schwierigkeit dar, da das Optimum lediglich für eine Nachricht oder eine bestimmte Nachrichtengruppe gefunden werden kann. Dieses dann allgemein für alle Nachrichten zu finden, ist ein kompliziertes Unterfangen, welches das Kernstück dieser Arbeit darstellt.

Ein besonderer Dank gilt der Fachhochschule Oberösterreich, Campus Hagenberg, die es mir ermöglicht hat, im Rahmen meines Masters in Interactive Media mein Projekt und meine Arbeit abzuwickeln. Im Speziellen möchte ich mich bei DI Martin Harrer für die gute Zusammenarbeit während des Projektes und des Schreibens der Arbeit und für das Übernehmen der Betreuung bedanken. Ein weiterer Dank gilt den Professoren des Studienganges Interactive Media und meinen Studienkollegen des Studienganges Interactive Media 2013, die mir immer mit Rat und Tat zur Seite standen. Für die aufwendige Korrektur meiner Arbeit geht ein großes Dankeswort an OStR Mag. Uwe Wolf. Nicht zuletzt möchte ich mich bei meiner Familie, meinem Partner und meinen Freunden für die tatkräftige moralische Unterstützung und ebenso das Korrekturlesen bedanken.

# Kurzfassung

In der heutigen Zeit stellt die Sentimentanalyse ein wichtiges Unterfangen dar, welches, vor allem für Unternehmen, immer mehr an Bedeutung gewinnt. Die Meinungen über eine bestimmte Dienstleistung, ein Produkt oder andere Angebote haben durch die zunehmende Verfügbarkeit auf Blogs, Bewertungen bei Onlineshops oder auch durch kurze Statements auf sozialen Netzwerken einen deutlich höheren Stellenwert als gedacht erlangt. Aus diesem Grunde ist eine automatisierte Bewertung beziehungsweise Einteilung in positive und negative Erläuterungen schon beinahe unerlässlich.

Eine solche Analyse ist speziell für Nachrichten auf Twitter entwickelt worden und der Versuch, andere bereits existierende Lösungen zu übertreffen stellt ein weiteres Ziel dar. Hierfür wurde Naïve Bayes als maschinelles Lernverfahren ausgewählt, welches sich bereits für die Anwendung bei einer Sentimentanalyse bewährt hat. Mittels verschiedener Textverarbeitungs- und Textaufbereitungsmethoden ist eine Verbesserung zu erzielen. Des Weiteren ist die Verwendung von unterschiedlichen Daten zum Anlernen des Verfahrens ausschlaggebend. Zusätzlich zu den Daten von Twitter werden Kinofilmkritiken herangezogen, da der Versuch gestartet wurde, hiermit die Klassifikation gravierend zu verbessern.

Zunächst liefern die ausschließlich mit Tweets trainierten Klassifikations-Modelle die ersten Testergebnisse, um deren Performance zu ermitteln. Diese Resultate dienen als Startwert, welche es mittels dem kombinierten Datenset aus Tweets und Movie-Reviews zu steigern gilt. Darüber hinaus sind Klassifikations-Modelle nur aus diesen Reviews erstellt und getestet worden, um zu überprüfen, ob sich so ein Datenset überhaupt für die Einteilung von Tweets in positiv oder negativ eignet. Die Klassifikatoren trainiert mit dem kombinierten Datenset sind natürlich ausführlich getestet worden. Auf die performantesten Modelle sind auch noch unterschiedliche Textverarbeitungsmethoden angewandt, welche die Performance steigern. Für alle Modelle sind die Texte vorher aufbereitet worden, so arbeiten einige mit Unigrammen, einige mit Bigrammen und wieder andere mit Uni- und Bigrammen zusammen. Dies soll auch Informationen darüber liefern, welche Textaufbereitung für eine Sentimentanalyse am sinnvollsten ist.

# Abstract

Nowadays, the sentiment analysis itself is an important undertaking – especially for companies – and gains more and more importance. All opinions about a specific service, product or other offers have a higher priority than imagined by the increasing availability of opinion-statements at blogs, online-shops or even though social networks. Due to the amount of data, an automated categorization in positive and negative is essential.

Such an analysis especially for messages on Twitter has been developed. Another aim was to outdo existing sentiment analysis solutions in performance. For this purpose, the Naïve Bayes Machine Learning method that already proven itself in sentiment analysis tasks, is chosen. Different text processing methods need to produce improvements in performance. Furthermore, the use of different data sets to train the Naïve Bayes classifier is decisive to gain such growth in performance. In addition to the Tweets used for training, a combination with movie reviews is tested for getting better categorization results.

Classification models trained with messages from Twitters Public Stream present the first test results to determine the models' performance to serve a starting value for possible outstrips by training the classifiers with a combined dataset of Tweets and movie reviews. Before such a combination, the movie review dataset itself is tested. The classification models have been created, trained and tested with movie reviews to check, if these data set is suitable for Tweet-classification purposes. Afterwards all classifiers trained with the combined data have been tested extensively. The training data of the best performing models are proved after filtering all words with different text processing methods, to see if the performance increases. All data have been prepared in advance to generate different models working with unigrams or either bigrams or both together. All tested classifiers provide information for the best working text processing and preparing methods, to see which combination delivers best performance.

# Kapitel 1

# Einleitung

Ein wichtiger Teil des heutigen Informationsbeschaffungsverhaltens ist beziehungsweise war schon immer herauszufinden, was andere Leute über ein bestimmtes Thema, Produkt oder ähnliches denken. Mit der zunehmenden Verfügbarkeit an Meinungen, in beispielsweise Blogs, Online-Bewertungen oder auch in sozialen Netzwerken, entstehen neue Herausforderungen: diese schnell in verschiedene Kategorien einzuteilen und auf eventuell schlechte Meinungen einzugehen. Durch die eben erwähnten Schwierigkeiten entstanden neue Bereiche in der digitalen Informationsverarbeitung: „Opinion Mining“ und „Sentimentanalyse“, welche sich mit Meinungen, Stimmungen und Subjektivität in Texten beschäftigt.

## 1.1 Problemstellung und Relevanz der Sentimentanalyse

Die Wichtigkeit dieser Themen wird durch die wachsende Anzahl an kostenpflichtigen Tools für die automatisierte Stimmungserkennung von Unternehmen wie „DataSift[1]“, „Brandwatch[2]“ und massenhaft anderen noch unterstrichen. Viele Unternehmen bauen heutzutage schon auf die automatisierte und schnelle Möglichkeit, Feedback für Ihre Produkte und Dienstleistungen zu bekommen. Außerdem wird im Bereich der Sentimentanalyse aktuell sehr intensiv Forschung betrieben. Die „Stanford Natural Language Processing Group[3]“ befasst sich an der Stanford Universität mit diesem Thema beziehungsweise arbeitet an Algorithmen, die Computer beim Verarbeiten und Verstehen von menschlicher Sprache unterstützen sollen. Des Weiteren ist seit Februar 2014 ein Wettbewerb auf www.kaggle.at gelistet, bei dem verschiedene Teams oder auch Einzelpersonen ein Tool entwickeln sollen, welches Filmkritiken in fünf verschiedene Kategorien einteilt.

[1] http://www.datasift.com/
[2] http://www.brandwatch.com/de/
[3] http://nlp.stanford.edu/

## 1.2 Motivation

Warum gerade maschinelle Lernverfahren für die Sentimentanalyse verwendet werden, beschreibt folgende Passage aus [27]:

> Indeed, applying machine learning techniques based on unigram models can achieve over 80% in accuracy, which is much better than the performance based on hand-picked keywords reported above.

Es wird deutlich gemacht, dass die Verwendung von einzelnen Wörtern der zu kategorisierenden Nachricht schon sehr gute Ergebnisse liefert. Das Heranziehen von Wortgruppen würde demnach die Richtigkeit der Kategorisierung noch steigern, da positive Wörter in Kombination mit einer Verneinung folglich negativ wären.

Als Entwickler einer Sentimentanalyse-Applikation steht man oft vor der Frage, ob eine Verbesserung der Klassifikation noch möglich wäre. Es gibt bereits viele Ansätze, die versuchen, eine hervorragende Genauigkeit zu erzielen, indem verschiedene Ansätze oder Modelle kombiniert werden. Dies klingt zwar verlockend, die Ausführung ist jedoch nicht so einfach. Natürlich ist eine immer besser werdende Genauigkeit beziehungsweise Richtigkeit der Klassifikation erstrebenswert, um eine vollkommen automatisierte Applikation anzubieten, die Unternehmen die Arbeit erleichtert, online schnell auf *alle* negativen Kommentare und Rezensionen einzugehen.

Kostenpflichtige Tools und Applikationen von Unternehmen wie „DataSift“ und „Brandwatch“ liefern bereits in Bezug auf die Genauigkeit und Richtigkeit einer Klassifikation sehr gute Lösungen. Allerdings sind diese Angebote aufgrund der großen Nachfrage, des geringen Angebots an Applikationen und des daraus resultierenden stattlichen Preises für viele Unternehmen – vor allem für kleinere Firmen – unerschwinglich. Eine intensive Forschung in dem Bereich der Computerlinguistik und somit auch der Sentimentanalyse und das Publizieren der Arbeiten soll es zukünftig ermöglichen, das Angebot an Applikationen zu erweitern und somit auch diese Tools für alle beziehungsweise viele zu einem zuträglichen Preis zu erhalten.

## 1.3 Zielsetzung der Arbeit

Im Rahmen dieser Arbeit soll auf die Klassifikation, in „positiv“ und „negativ“, von kurzen Nachrichten, welche in sozialen Netzwerken, im Speziellen Twitter, vorkommen, eingegangen werden. Der Schwerpunkt liegt auf der Verwendung von maschinellen Lerntechniken, die zum Einteilen der Nachrichten in die verschiedenen Klassen verwendet werden und wie sich die unterschiedliche Aufbereitung der Texte beziehungsweise Nachrichten zum Anlernen des Klassifikations-Modells auf die Ergebnisse der Einstufung auswirken. Im Speziellen soll untersucht werden, wie sich die Veränderungen

auf die Qualität der Klassifikation auswirken und es soll versucht werden, durch die Kombination verschiedener Modelle eine verbesserte Lösung zu finden.

Im Detail sollen unterschiedliche Wissensbasen, welche Nachrichten von Twitter oder Filmkritiken enthalten, verwendet werden, um ein Klassifikations-Modell zu erstellen. Diese Nachrichten müssen bereits vor der Erstellung des Modells als „positiv“ oder „negativ“ gekennzeichnet sein, da es sonst nicht generiert werden kann, weil dieser Klassifikator auf einem maschinellen Lernverfahren basiert. Es soll lediglich *ein* maschinelles Lernverfahren gewählt und dessen Verwendung mittels unterschiedlicher Aufbereitung der Inhalte der Wissensbasen in Punkto Qualität der Klassifikation untersucht werden. Um eine Aussage über die Qualität der Sentimentanalyse machen zu können, stehen unterschiedliche „Kennzahlen“, die berechnet werden können, zur Verfügung [42, 48]:

- *Accuracy* (Genauigkeit),
- $FPR$ (Falsch-Positiv-Rate),
- $FNR$ (Falsch-Negativ-Rate),
- $TPR$ (Richtig-Positiv-Rate),
- $TNR$ (Richtig-Negativ-Rate),
- $F_1$ *Score* (Maß der Genauigkeit, harmonische Mittel von *Precision* und *Recall*),
- $LR_N$ (negative Wahrscheinlichkeits-Rate),
- $LR_P$ (positive Wahrscheinlichkeits-Rate),
- *Precision* (Präzision),
- *Recall* (Rückruf),
- *Sensitivity* (Sensitivität) und
- $Specificity$ (Spezifität).

Auf die zur Evaluierung des entwickelten Klassifikations-Modelles verwendeten Kennzahlen wird später in Abschnitt 5.1 noch ausführlich eingegangen.

## 1.4 Aufbau der Arbeit

Der inhaltliche Aufbau dieser Arbeit gliedert sich in vier Hauptteile, zwei Evaluierungsteile und den Schluss. Das erste Kapitel, die Einleitung, beschreibt die Entstehung der Arbeit, Motivation, Ziele und den Aufbau.

Kapitel 2 gibt eine Einführung in die Materie und beschreibt eine Sentimentanalyse im Detail. Neben der allgemeinen Erklärung der Sentimentanalyse (Abschnitt 2.1) wird ein Einblick in maschinelle Lernverfahren gegeben (Abschnitte 2.2, 2.3 und 2.4). Klassische Textverarbeitungs- und Textaufbereitungs-Verfahren werden ebenfalls erläutert (Abschnitt 2.5), angefangen von Sprachidentifikation, N-Grammen, Textnormalisierung, Stem-

ming, die Berücksichtigung von Verneinungen, Sarkasmus Erkennung und Stoppwort-Entfernung sind vertreten (Abschnitte 2.5.1–2.5.7).

Eine Erklärung des eigenen Ansatzes zur Steigerung der Qualität einer Sentimentanalyse ist in Kapitel 3 vertreten. Es wird auf das Klassifikations-Modell eingegangen (Abschnitt 3.1). Des Weiteren werden die verwendeten Trainingsdaten in Abschnitt 3.2 näher beschrieben und die Speicherung und Verarbeitung dieser erklärt (Abschnitte 3.3 und 3.4). Nun kann der Ablauf der Klassifikation besprochen und auf ähnliche Ansätze eingegangen werden (Abschnitte 3.5 und 3.6).

Das vierte Kapitel bildet den praktischen Hauptteil dieser Arbeit. Hier wird der im Rahmen des Masterprojektes entwickelte Prototyp einer verbesserten Sentimentanalyse vorgestellt und auf die Implementierung im Detail näher eingegangen. Nähere Implementierungsdetails, die verwendeten Bibliotheken und Allgemeines zu den Trainingsdaten-Sets sind als Einleitung in den Abschnitten 4.1, 4.2 und 4.3 zu finden. Die Aufbereitung der Trainingsdaten wird anschließend in Abschnitt 4.4 genau erklärt. Zuletzt wird die Klassifikation in Abschnitt 4.5 beschrieben und genau darauf eingegangen.

Anschließend an den inhaltlichen Hauptteil wird im Evaluierungsteil aufgezeigt, wie gut der eigene Ansatz zur Verbesserung der Sentimentanalyse durch Kombination verschiedener Klassifikations-Modelle abschneidet. In Kapitel 5 werden Tests anhand von verschiedenen Kennzahlen durchgeführt und die Ergebnisse aufgezeigt. Zu Beginn werden die verschiedenen Kennzahlen genau erklärt (Abschnitt 5.1). Anschließend werden die Tests in drei Kategorien unterteilt:

- Tests mit dem Tweet-Datenset (Abschnitt 5.2),
- Tests mit dem Filmkritiken-Datenset (Abschnitt 5.3) und
- Tests mit beiden Datensets in Kombination (Abschnitt 5.4).

Alle Testergebnisse werden in Kapitel 6 zusammengefasst und ausführlich diskutiert. Im letzten Kapitel werden Schlussbemerkungen angehängt, welche den eigenen Ansatz kurz zusammenfassen, ein Fazit zu den Testergebnissen beziehungsweise der Arbeit und einen Ausblick auf weitere Forschungsmöglichkeiten geben.

# Kapitel 2

# Theoretischer Hintergrund

Um die folgenden Kapitel dieser Arbeit verstehen zu können, muss zunächst eine Grundlage für die Teile einer Sentimentanalyse geschaffen werden. Zunächst muss die Sentimentanalyse im Allgemeinen verstanden werden, danach kann auf die einzelnen Teile wie maschinelle Lernverfahren, Textverarbeitungsmethoden, Textaufbereitungsmethoden eingegangen werden.

## 2.1 Was ist eine Sentimentanalyse?

Die Sentimentanalyse im Allgemeinen kann mit folgendem Zitat aus [43] beschrieben werden:

> Die Sentimentanalyse versucht, ein Stimmungsbild anhand einzelner Quellen und der darin verwendeten sprachlichen Mittel zu deuten. Es wird zwischen neutralem, positivem und negativem Inhalt in Zusammenhang mit dem Schlüsselwort unterschieden. Dadurch können bei einem negativen Stimmungsbild genaue Gründe genannt werden und man kann dementsprechende Gegenmaßnahmen unternehmen.

Im Detail ist es eine Technologie, bei der Schlüsse über Inhalte gezogen werden, welche beispielsweise bei Facebook[1], Twitter[2], Tumblr[3], Youtube[4] und vielen anderen Social-Web-Plattformen publiziert wurden. Ob es sich dabei um Kritik oder Lob, also positive oder negative Nachrichten, handelt, steht im Vordergrund der Analyse. Des Weiteren können auch Inhalte aus Foren, Rezensions-Seiten oder Ähnlichem analysiert werden. Als Grundlage für einen solchen Analyse-Vorgang dienen Polarität, Subjektivität oder die Art der gewählten Wörter. Mit Hilfe von unterschiedlichen Algorithmen können Emotionen, auf einen bestimmten Text bezogen, zumindest

[1] https://www.facebook.com/
[2] https://twitter.com/
[3] https://www.tumblr.com/
[4] https://www.youtube.com/

tendenziell erkannt werden. Die Genauigkeit beziehungsweise Qualität heutiger Sentimentanalyse-Applikationen sind eher schwankend und weit von 100% entfernt [33]. Die heutigen Genauigkeitswerte liegen allerdings schon bei circa 80%, was ein akzeptables, aber noch verbesserungsfähiges Ergebnis darstellt [27].

## 2.2 Maschinelles Lernen

Da bereits in Abschnitt 2.1 erklärt wurde, was eine Sentimentanalyse überhaupt ist und wie sie im Allgemeinen gesehen wird, kann nun auf maschinelle Lernverfahren eingegangen und ein Zusammenhang zwischen den beiden Bereichen hergestellt werden. Zunächst muss auf das Lernen generell eingegangen werden: „Was ist Lernen überhaupt?“. Eine solche Frage stellt eher ein philosophisches Problem dar, auf welchem nicht der Schwerpunkt liegt. Allerdings ist es gerade hier interessant zu sehen, wie schwierig eine solche Definition oder Begriffserklärung erscheinen kann. Laut dem Wörterbuch „Duden[5]“ hat „lernen“ unterschiedliche Bedeutungen:

- sich Wissen, Kenntnisse aneignen,
- sich, seinem Gedächtnis einprägen,
- Fertigkeiten erwerben,
- im Laufe der Zeit (durch Erfahrungen, Einsichten) zu einer bestimmten Einstellung, einem bestimmten Verhalten gelangen und
- (ein Handwerk) erlernen [41].

Diese Definitionen weisen allerdings einige Schwierigkeiten in Kombination mit Maschinen – im Speziellen Computer – auf. Es ist schier unmöglich zu testen, ob das „Lernen“ erreicht wurde, wenn die ersten beiden Definitionen angewandt werden, da nicht überprüft werden kann, ob ein Computer Kenntnis von etwas hat. Es ist lediglich möglich, Fragen zu stellen, allerdings wird hier nicht das Erlernte getestet, sondern die Fähigkeit, Fragen zu beantworten. Fertigkeiten erwerben oder etwas erlernen, diese Aufgaben sind für eine Maschine der heutigen Zeit triviale Aufgaben. Es ist wichtig, die Leistung zu Verbessern beziehungsweise das Potenzial auf „neue“ Situationen „gut“ zu reagieren, deshalb kann in diesem Fall lernen mit folgendem Zitat aus [34] definiert werden:

> Things learn when they change their behavior in a way that makes them perform better in the future.

Hier wird das Lernen auf eine andere Art und Weise beschrieben. Grundsätzlich ist es das Erkennen von Mustern und das Treffen von intelligenten Entscheidungen, basierend auf einer Menge von Daten. Der Schwerpunkt liegt hier auf der Automatisierung des Verfahrens, was zu der Anwendung

[5] http://www.duden.at

von Lernalgorithmen führt, welche durch Hinzuziehen einer Wissensbasis angelernt werden. Somit ist maschinelles Lernen ein zentraler Teilbereich der „künstlichen Intelligenz[6]“. Es ist sehr unwahrscheinlich, dass ein System jede Art von intelligenten Problemen lösen kann, da die Art des Problems, die Sprache oder andere Parameter eine Verallgemeinerung schier unmöglich machen. Deshalb wird meist für eine einzelne Aufgabe eine intelligente Lernapplikation entwickelt. Ein System kann nur dann intelligent entscheiden, wenn es vorher von bereits vorhandenen Daten gelernt hat, welche auch von Menschen aufbereitet werden müssen [46].

## 2.3 Naïve Bayes als maschinelles Lernverfahren

Naïve Bayes wurde aus folgendem Grund als das in dieser Arbeit verwendete Lernverfahren ausgewählt: der Bayes Lernalgorithmus, der explizite Wahrscheinlichkeiten für Hypothesen berechnet, wie der Naïve Bayes Klassifikator, gehören zu den mit Abstand praktischsten Ansätzen, um unterschiedliche maschinelle Lernprobleme zu lösen [24]. In „Machine Learning, Neural and Statistical Classification“ werden verschiedene Lernverfahren mit dem Naïve Bayes Klassifikator verglichen und aufgezeigt, dass im Wettbewerb mit den anderen Verfahren, wie beispielsweise dem Entscheidungsbäume Algorithmus und dem Neural Netzwerk Algorithmus, das Naïve Bayes Verfahren in einigen Vergleichspunkten die anderen übertrifft [23].

Um zu verstehen, wie der Naïve Bayes Klassifikator funktioniert, muss zunächst erklärt werden, was in diesem Fall mit einer Klassifikation überhaupt gemeint ist, aber weit essentieller ist auch das Verstehen des Bayes Theorems.

### 2.3.1 Klassifikation

Bei einer Klassifikation werden Ausdrücke, Gegenstände, Erfahrungen oder Ähnliches in unterschiedliche Klassen (Gruppen) oder auch Unterklassen systematisch eingeteilt [41]. Angenommen in einem kleinen Klassifikationssystem existieren zwei Klassen, „positiv“ und „negativ“, die für eine Zuordnung vorhanden sind, dann können mit diesem System beispielsweise Textnachrichten mit „positiv“ oder auch „negativ“, gekennzeichnet werden.

### 2.3.2 Bayes Theorem

Beim maschinellen Lernen ist oftmals das Bestimmen der geeignetsten Hypothese in einem bestimmten Raum ($H$), durch Berücksichtigung aller Trainingsdaten ($D$) für einen konkreten Ausdruck ($T$) von Interesse. Um die

[6] Mehr Informationen dazu sind unter http://wirtschaftslexikon.gabler.de/Definition/kuenstliche-intelligenz-ki.html zu finden.

**Tabelle 2.1:** Grundnotationen der Wahrscheinlichkeitsrechnung [45].

| | |
|---|---|
| $P$ | Maß der Wahrscheinlichkeit; vom Englischen „probability“ abgeleitet. |
| $A$, $B$ | Ereignis, wofür die Wahrscheinlichkeit des Eintretens berechnet werden soll. |
| $P(A)$ | Wahrscheinlichkeitswert für das Eintreten des Ereignisses $A$, welcher immer eine reelle Zahl $\mathbb{R}$: $0 \leq P(A) \leq 1$ ist. Es existieren Extremfälle die Sicherheit bieten:<br>• $P(A) = 1$: $A$ tritt zu 100% ein und<br>• $P(A) = 0$: $A$ tritt zu 100% *nicht* ein. |
| $P(A \mid B)$ | Bedingte Wahrscheinlichkeit: Wahrscheinlichkeitswert für das Eintreten des Ereignisses $A$ unter der Bedingung, dass Ereignis $B$ bereits eingetreten ist. |

beste Hypothese herauszufinden, kann die wahrscheinlichste Hypothese berechnet werden, die wiederum die Wahrscheinlichkeitswerte mittels der Trainingsdaten bestimmt und zusätzlich die vorherigen Wahrscheinlichkeiten der verschiedenen Hypothesen des bestimmten Raumes berücksichtigt. Mittels des Bayes Theorems kann eine solche Wahrscheinlichkeitsberechnung durchgeführt werden beziehungsweise bietet es eine Möglichkeit, eine solche Hypothese mit Hilfe

- der früheren Wahrscheinlichkeit,
- die Wahrscheinlichkeiten der Hypothesen, die aus der Beobachtung von bestimmten Daten resultieren und
- die beobachteten Daten an sich

aufzustellen [24]. Auf die Erklärung wird später in diesem Kapitel noch genauer eingegangen und, um sie besser zu verstehen, wird ein Beispiel aufgezeigt.

Bevor auf das Bayes Theorem noch genauer eingegangen werden kann, müssen die dafür benötigten Notationen der Wahrscheinlichkeitsrechnung verstanden werden. In Tabelle 2.2 werden diese aufgelistet und kurz erklärt [45, 24]. Wenn diese Grundnotationen noch nicht klar sind, können in Tabelle 2.1 die grundlegenden Schreibweisen und Parameter der Wahrscheinlichkeitsrechnung eingesehen werden. Es wird hauptsächlich auf die Erklärung der Grundwahrscheinlichkeit und dessen Abhängigkeiten eingegangen und diese erklärt.

**Tabelle 2.2:** Die Grundnotationen der Wahrscheinlichkeitsrechnung, die für das Bayes Theorem benötigt werden, im Überblick [45, 24].

| | |
|---|---|
| $h$ | Hypothese |
| $D$ | Datensätze der Trainingsdaten |
| $P(h)$ | Wahrscheinlichkeit, dass $h$ wahr ist, bevor die Trainingsdaten berücksichtigt wurden. Wird auch vorherige Wahrscheinlichkeit von $h$ genannt und beschreibt das Hintergrundwissen, der Wahrscheinlichkeit, dass $h$ richtig ist. |
| $P(D)$ | Vorherige Wahrscheinlichkeit, zu welcher die Trainingsdaten $D$ beobachtet werden (gibt keine Auskunft darüber, welche Hypothese h zu dem Datensatz der Trainingsdaten $D$ gehört). |
| $P(D \mid h)$ | Wahrscheinlichkeit, dass $D$ zutrifft (wahr ist) unter der Bedingung, dass $h$ bereits bekannt und wahr ist. |
| $P(h \mid D)$ | Spätere Wahrscheinlichkeit von $h$, bei dem das Vertrauen, dass $h$ zutrifft, nachdem die Trainingsdaten $D$ erfasst wurden, ermittelt wird. Der Einfluss der Trainingsdaten $D$ wird reflektiert, wobei die Wahrscheinlichkeit bei $P(h)$ unabhängig von $D$ ist. |

Das Bayes Theorem ist die Grundlage für Bayes Lernmethoden, da es eine Möglichkeit bietet, die bedingte Wahrscheinlichkeit $P(h \mid D)$ zu berechnen. Dazu werden die in Tabelle 2.2 erwähnten Wahrscheinlichkeiten benötigt und wie folgt in die Gleichung

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)} \tag{2.1}$$

gebracht und verwendet. Wird die Gleichung genauer betrachtet und analysiert, sind folgende Änderungen des Wahrscheinlichkeitswertes zu erkennen:

- zunehmend, je größer $P(h)$ und $P(D \mid h)$ werden und
- fallend, je größer $P(D)$ wird.

Diese Änderungen beziehen sich darauf, dass $P(D)$ einen sich nicht ändernden Wert aufweist und unabhängig von $h$ ist.

Doch wie ist nun der Zusammenhang des Bayes Theorems mit dem maschinellen Lernen? Im Normalfall wird ein Set mit Hypothesen ($H$) zusammengestellt und anschließend soll für ein Datenset ($D$) die Hypothese, die am Wahrscheinlichsten ist (oder eine der maximalen Wahrscheinlichkeiten, wenn

mehrere existieren), gefunden werden ($h \in H$). Die maximale Wahrscheinlichkeit wird dann „maximum a posteriori“ (MAP) Hypothese genannt und kann folgendermaßen durch das Anwenden des Bayes Theorems für $P(h \mid D)$ für jede mögliche Hypothese ermittelt werden [24]:

$$h_{MAP} \equiv \arg\max P(h \mid D) \tag{2.2}$$

$$= \arg\max \frac{P(D \mid h)P(h)}{P(D)} = \arg\max P(D \mid h)P(h). \tag{2.3}$$

Das Bayes Theorem ist ein sehr komplexer Teil der Wahrscheinlichkeitsrechnung und somit genauer zu erklären. Bei vielen Ereignissen oder auch Phänomenen ist es schwierig, sichere Aussagen über ihr Eintreffen anzustellen. Gute Beispiele dafür sind das Wetter, die Gesundheitsentwicklung oder der Gesundheitszustand eines Menschen, Glücksspiele oder ähnliches. Um dennoch Äußerungen über das Zutragen solcher Situationen zu treffen, wird die Wahrscheinlichkeitsrechnung herangezogen, wobei Werte von Zufallsexperimenten als Modelle für reale Vorgänge dienen. Die Verwendung dieser Werte lässt allerdings ein Problem auftreten: Es können keine Wahrscheinlichkeitswerte „an sich“ verwendet werden, nur die bisher aufgetretenen Häufigkeitswerte der Versuche. Des Weiteren müssten solche Experimente gleicherweise unendlich oft wiederholt werden, was in der Praxis jedoch ein unmögliches Unterfangen darstellt. Es wird oft auf Modelle der Wahrscheinlichkeitsrechnung zurückgegriffen, welche Eigenschaften nur ungenau bekannt sind [45]. Um diese Art von Wahrscheinlichkeitsberechnung zu verstehen, soll eine simple Beispielsituation durchgedacht werden, wobei oft die bedingte Wahrscheinlichkeit im Vordergrund der Erklärungen beziehungsweise Berechnungen stehen wird.

### Bayes Theorem – ein Beispiel

Um das Bayes Theorem besser zu verstehen, soll ein Beispiel Schritt für Schritt genau durchgearbeitet werden. Dafür wird folgendes Zitat aus [29] als Vorlage für Angabe dienen:

> Assume that you're presented with three coins, two of them fair and the other a counterfeit that always lands heads. If you randomly pick one of the three coins, the probability that it's the counterfeit is 1 in 3. This is the prior probability of the hypothesis that the coin is counterfeit. Now after picking the coin, you flip it three times and observe that it lands heads each time. Seeing this new evidence that your chosen coin has landed heads three times in a row, you want to know the revised posterior probability that it is the counterfeit. The answer to this question, found using Bayes's theorem (calculation mercifully omitted), is 4 in 5. You thus revise your probability estimate of the coin's being counterfeit upward from 1 in 3 to 4 in 5.

**Tabelle 2.3:** Verhalten der einzelnen Münzen Kopf oder Zahl nach einem Wurf zu zeigen.

| | Kopf | Zahl |
|---|---|---|
| Faire Münze 1 | $0,5$ | $0,5$ |
| Faire Münze 2 | $0,5$ | $0,5$ |
| Unfaire Münze | $1,0$ | $0,0$ |

Dieses Zitat beschreibt das Bayes Theorem, welches in dem Buch „The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from two centuries of controversy" erklärt wird, folgendermaßen [21]: Es wird die Annahme getätigt, dass drei Münzen zu Verfügung stehen. Zwei dieser Münzen sind fair (eine $50:50$ Chance, Kopf oder Zahl zu werfen) und eine ist so manipuliert, dass nach einem Wurf immer Kopf angezeigt wird. Danach wird eine der drei Münzen per Zufall ausgewählt, die Wahrscheinlichkeit, dass es sich hierbei um die manipulierte Münze handelt, ist $1/3$. Auch die Wahrscheinlichkeit, dass die erste beziehungsweise die zweite faire Münze gewählt wird, ist je $1/3$. Dies ist die vorherige Wahrscheinlichkeit der Hypothese, dass es sich um die manipulierte Münze handelt. Nach dem Auswählen einer Münze wird diese dreimal geworfen, wobei nach den Würfen immer Kopf zu sehen ist. Mit diesen Erkenntnissen soll nun die vorherige Wahrscheinlichkeit, ob es sich um die manipulierte Münze handelt oder nicht, neu ermittelt werden. Diese Berechnung kann mit Hilfe des Bayes Theorems durchgeführt werden. Aber wie kommt dieser neue Wahrscheinlichkeitswert zustande? Diese Frage soll nun mittels Durchrechnen des Beispiels beantwortet werden.

Laut der Angabe sind drei Münzen mit unterschiedlichen Chancen, Kopf zu werfen, vorhanden. Die Wahrscheinlichkeitswerte für diese Münzen sind in Tabelle 2.3 strukturiert dargestellt. Nun stellt sich folgende Frage: Wurde eine der fairen Münzen oder die unfaire geworfen, wenn dreimal hintereinander Kopf gezeigt wurde? Hier geht es nicht um die Wahrscheinlichkeitsberechnung eines Versuchsausganges, sondern darum, vom Ergebnis des Versuchs Rückschlüsse auf das Zufallsexperiment an sich zu tätigen. Laut Tabelle 2.3 kann mit allen Münzen Kopf geworfen werden, deshalb kann man nicht mit 100%iger Sicherheit sagen, welche Münze verwendet wurde. Allerdings ist es möglich, eine Wahrscheinlichkeitssausage darüber zu tätigen und diese zu berechnen. Die bedingte Wahrscheinlichkeit wird verwendet, um zu berechnen wie wahrscheinlich es ist, dass eine der fairen Münzen oder die unfaire Münze geworfen wurde, wenn Kopf gezeigt wird. Dies lässt sich für das Beispiel in Tabelle 2.3 ausdrücken durch

$$P(K \mid F) = 0,5 \qquad \text{und} \qquad P(K \mid U) = 1,0\,. \tag{2.4}$$

**Tabelle 2.4:** Definition der Apriori-Wahrscheinlichkeit für das Wählen einer Münze.

| | |
|---|---|
| $P_a(F)$ | Wahrscheinlichkeit, eine faire Münze zu wählen |
| $P_a(U)$ | Wahrscheinlichkeit, die unfaire Münze zu wählen |

Hier steht $K$ für „Kopf geworfen“, $F$ für eine der fairen Münzen und $U$ für die unfaire Münze. Diese Werte ergeben sich aus der oben angeführten Angabe und sind somit auch Tabelle 2.3 zu entnehmen. Die bedingte Wahrscheinlichkeit von $P(K \mid F)$ bezieht sich lediglich auf *einen* Wurf. Somit wird zuerst die Wahrscheinlichkeit für das einmalige Werfen der Münze berechnet. Auch die weiterführenden Berechnungen werden mit diesen durchgeführt, um zu überprüfen, ob auch schon nach *einem* Wurf eine Aussage über die geworfene Münze angestellt werden kann

Diese berechneten Wahrscheinlichkeiten sind keine Lösung zu der gestellten Frage, da diese für die Wahrscheinlichkeit der geworfenen Seite, unter der Bedingung, dass die Münze bekannt ist, stehen. Um die Wahrscheinlichkeit, welche Münze verwendet wurde, unter der Bedingung, dass Kopf geworfen wurde, zu berechnen, wird noch dazu die Wahrscheinlichkeit, nachdem der Münzwerfer eine Münze auswählt, benötigt. Diese wird im Allgemeinen die „Apriori-Wahrscheinlichkeit“ genannt und wie in Tabelle 2.4 definiert.

Da nun die Apriori-Wahrscheinlichkeiten erklärt wurden, kann das Experiment als Ganzes betrachtet werden. Zu Beginn muss eine der Münzen ausgewählt werden, was entsprechend der Apriori-Wahrscheinlichkeiten getan wird, siehe Abbildung 2.1. Die oberen Pfade (gelb, grün und blau) stellen die Wahl einer Münze dar. Erst nach der Auswahl kann mit dem Werfen begonnen werden. Kopf kann mit allen drei Münzen geworfen werden (siehe Abbildung 2.1 gelb, grün und blaue Pfade von der Münze zu „K“). Für das Zufallsexperiment sind also der gelbe, grüne und blaue Pfad relevant, alle anderen können außer Acht gelassen werden. Diesen Pfaden können nun folgende Wahrscheinlichkeiten zugeordnet werden:

$$P(F \wedge K) = P_a(F) \cdot 0,5, \tag{2.5}$$

$$P(U \wedge K) = P_a(U) \cdot 1,0 = P_a(U). \tag{2.6}$$

Die erste Wahrscheinlichkeitsberechnung (Gleichung 2.5) kann für den gelben und grünen Pfad in Abbildung 2.1 verwendet werden und die zweite (Gleichung 2.6) für den blauen Pfad. Diese Multiplikationen folgen aus den allgemeinen Regeln der Wahrscheinlichkeitsrechnung, die in dieser Arbeit nicht besprochen werden.[7]

[7] Für eine detaillierte Erklärung der allgemeinen Regeln siehe http://www.mathe-online.at/mathint/wstat1/i.html unter dem Punkt "Baumdiagramme“.

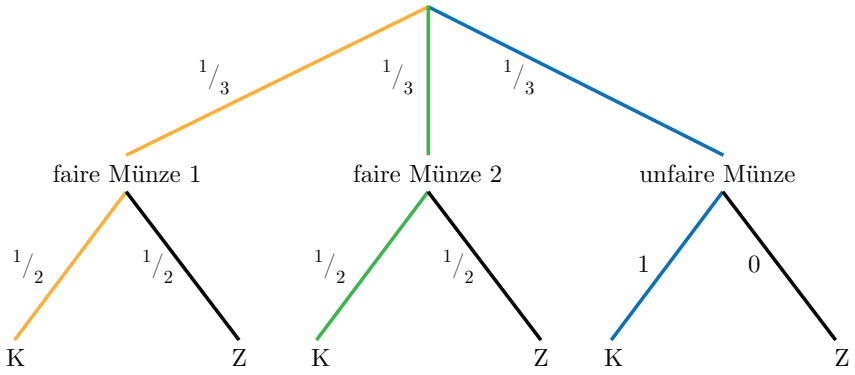



**Abbildung 2.1:** Ein Baumdiagramm für den Ablauf des Zufallsexperimentes (Wahl einer Münze und Werfen der Münze). Prinzipiell sind sechs Versuchsausgänge möglich: a) Wahl der ersten fairen Münze und das Werfen von Kopf (gelber Pfad), b) Wahl der ersten fairen Münze und das Werfen von Zahl, c) Wahl der zweiten fairen Münze und das Werfen von Kopf (grüner Pfad), d) Wahl der zweiten fairen Münze und das Werfen von Zahl, e) Wahl der unfairen Münze und das Werfen von Kopf (blauer Pfad) und f) Wahl der unfairen Münze und das Werfen von Zahl. Allerdings ist bei dem Wurf der unfairen Münze der Ausgang Zahl nicht möglich, da die Wahrscheinlichkeit gleich 0 ist und somit immer Kopf erscheint.

Um zur Beantwortung der eigentlichen Frage zu kommen, sollten alle Versuchsausgänge, bei denen nicht Kopf geworfen wurde, ignoriert werden. Dies führt dann zu den bedingten Wahrscheinlichkeiten

$$P(F \wedge K) = P(F \mid K) \cdot P(K), \tag{2.7}$$
$$P(U \wedge K) = P(U \mid K) \cdot P(K). \tag{2.8}$$

Die Berechnung dieser bedingten Wahrscheinlichkeiten kann aus der Multiplikationsregel

$$P(A \wedge B) = P(A \mid B) \cdot P(B) \tag{2.9}$$

herbeigeführt werden.[8]

Aus den Gleichungen 2.7 und 2.8 können die zwei bedingten Wahrscheinlichkeitsberechnungen $P(F \mid K)$ und $P(U \mid K)$ entnommen werden, welche genau die Frage aus der Angabe beantworten, wie wahrscheinlich es ist, dass die faire beziehungsweise die unfaire Münze bei dem Experiment (bei drei Würfen, dreimal Kopf zu werfen) verwendet wurde, unter der Bedingung,

[8] Um eine genauere Erklärung dieser Regel zu erhalten, siehe http://www.mathe-online.at/mathint/wstat1/i.html unter dem Punkt „Bedingte Wahrscheinlichkeit“ und „Multiplikationsregel für Wahrscheinlichkeiten“.

dass Kopf geworfen wurde. Werden diese Berechnungen mit Hilfe der Gleichungen 2.5 und 2.6 umgeformt, so ergeben sich:

$$P(F \mid K) = \frac{P(F \wedge K)}{P(K)} = \frac{P_a(F) \cdot 0.5}{P(K)}, \tag{2.10}$$

$$P(U \mid K) = \frac{P(U \wedge K)}{P(K)} = \frac{P_a(U) \cdot 1.0}{P(K)}. \tag{2.11}$$

Die Summe der drei Größen[9] $2 \cdot P(F \mid K)$ und $P(U \mid K)$ ergeben 1 und so kann die Normierungsbedingung

$$\frac{P_a(F) \cdot \frac{1}{2}}{P(K)} + \frac{P_a(F) \cdot \frac{1}{2}}{P(K)} + \frac{P_a(U) \cdot 1}{P(K)} = \frac{\frac{1}{3} \cdot \frac{1}{2}}{P(K)} + \frac{\frac{1}{3} \cdot \frac{1}{2}}{P(K)} + \frac{\frac{1}{3}}{P(K)} = 1\,,$$

verwendet werden, um die Konstante $P(K)$ der Gleichungen 2.10 und 2.11 zu berechnen:

$$P(K) = \tfrac{1}{3} \cdot \tfrac{1}{2} + \tfrac{1}{3} \cdot \tfrac{1}{2} + \tfrac{1}{3} = \tfrac{4}{6}. \tag{2.12}$$

Da die Konstante $P(K)$ nun berechnet wurde, können die Wahrscheinlichkeiten für

a) das Verwenden der ersten fairen Münze, unter der Bedingung, dass Kopf geworfen wurde,
b) das Verwenden der zweiten fairen Münze, unter der Bedingung, dass Kopf geworfen wurde und
c) das Verwenden der unfairen Münze, unter der Bedingung, dass Kopf geworfen wurde

berechnet werden, wobei die Werte für a) und b) gleich sind. Die Wahrscheinlichkeit für das Verwenden der fairen Münze, unter der Bedingung, dass Kopf geworfen wurde, wird mittels

$$P(F \mid K) = \frac{P_a(F) \cdot \frac{1}{2}}{P(K)} = \frac{\frac{1}{3} \cdot \frac{1}{2}}{\frac{4}{6}} = \tfrac{1}{6} \cdot \tfrac{6}{4} = \tfrac{1}{4}. \tag{2.13}$$

berechnet. Es ist zu sehen, dass die erste faire Münze mit einer Wahrscheinlichkeit von 1/4 also 0.25 verwendet wurde, ebenso die zweite faire Münze. Die Wahrscheinlichkeit für das Werfen der unfairen Münze unter selber Bedingung ergibt

$$P(U \mid K) = \frac{P_a(U) \cdot 1}{P(K)} = \frac{\frac{1}{3}}{\frac{4}{6}} = \tfrac{1}{3} \cdot \tfrac{6}{4} = \tfrac{1}{2}, \tag{2.14}$$

[9] Die Wahrscheinlichkeit $P(F \mid K)$ für die Verwendung der fairen Münze unter der Bedingung, dass Kopf geworfen wurde wird zweimal verwendet, da auch zwei faire Münzen bei dem Experiment vorhanden sind.

wobei hier ein Wahrscheinlichkeitswert von 1/2 oder 0.5 auftritt.

Zusammenfassend ist zu sagen, dass mit einer 50%igen Wahrscheinlichkeit die unfaire Münze für das Experiment verwendet wurde. Eine faire Münze jedoch wurde nur mit einer Wahrscheinlichkeit von 25% verwendet. Da es aber zwei faire Münzen gibt und dessen Wahrscheinlichkeitswerte addiert werden müssen, ergibt dies wiederum einen Wert von 50% und ist somit gleich wahrscheinlich, wie das Verwenden der unfairen Münze. Nach nur *einem Wurf* kann also auf keinen Fall eine Aussage über die verwendete Münze angestellt werden. Deshalb wird auch nachfolgend noch die Wahrscheinlichkeit für das *dreimalige* Werfen *einer* Münze berechnet. Des Weiteren ist aus den beiden Gleichungen 2.10 und 2.11 zu entnehmen, dass eine Verallgemeinerung in der Form

$$P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A) \tag{2.15}$$

möglich ist. Diese Gleichung wird Satz von Bayes genannt, weshalb die Berechnungen des Beispiels auf das Bayes Theorem zurückzuführen sind.

Beim dreimaligen Werfen der Münze ändern sich die beiden Wahrscheinlichkeitswerte $P(K \mid F)$ und $P(K \mid U)$, welche lediglich das Eintreffen von Kopf unter den Bedingungen, dass die faire oder die unfaire Münze verwendet wurde, darstellen:

$$P((K \mid F) \wedge (K \mid F) \wedge (K \mid F)) = \tfrac{1}{2} \cdot \tfrac{1}{2} \cdot \tfrac{1}{2} = \tfrac{1}{8}, \tag{2.16}$$

$$P((K \mid U) \wedge (K \mid U) \wedge (K \mid U)) = 1,0 \cdot 1,0 \cdot 1,0 = 1,0. \tag{2.17}$$

Durch diese Anpassung ändert sich auch der Wert von $P(K)$, welcher mittels der Normierungsbedingung

$$\frac{P_a(F) \cdot \frac{1}{8}}{P(K)} + \frac{P_a(F) \cdot \frac{1}{8}}{P(K)} + \frac{P_a(U) \cdot 1}{P(K)} \frac{\frac{1}{3} \cdot \frac{1}{8}}{P(K)} + \frac{\frac{1}{3} \cdot \frac{1}{8}}{P(K)} + \frac{\frac{1}{3}}{P(K)} = 1$$

ermittelt wurde. Die Wahrscheinlichkeit hieführ

$$P(K) = \tfrac{1}{3} \cdot \tfrac{1}{8} + \tfrac{1}{3} \cdot \tfrac{1}{8} + \tfrac{1}{3} = \tfrac{5}{12}. \tag{2.18}$$

beläuft sich nun auf 5/12.

Jetzt ist es möglich, die Wahrscheinlichkeiten für das Verwenden der fairen Münze unter der Bedingung, dass dreimal hintereinander Kopf geworfen wurde und das Verwenden der unfairen Münze ebenso unter der Bedingung, dass dreimal hintereinander Kopf geworfen wurde, zu berechnen. Für das Werfen der fairen Münze unter der Bedingung, dass Kopf geworfen wurde, wurde folgende Berechnung durchgeführt:

$$P(F \mid K) = \frac{P_a(F) \cdot \frac{1}{2}}{P(K)} = \frac{\frac{1}{3} \cdot \frac{1}{8}}{\frac{5}{12}} = \tfrac{1}{24} \cdot \tfrac{12}{5} = \tfrac{1}{10}. \tag{2.19}$$

Die Berechnung für das Verwenden der unfairen Münze unter selber Bedingung sieht wie folgt aus:

$$P(U \mid K) = \frac{P_a(U) \cdot 1}{P(K)} = \frac{\frac{1}{3} \cdot 1}{\frac{5}{12}} = \frac{1}{3} \cdot \frac{12}{5} = \frac{4}{5}. \tag{2.20}$$

Aus den Ergebnissen der beiden Berechnungen für $P(F \mid K)$ (Gleichung 2.19) und $P(U \mid K)$ (Gleichung 2.20) ist zu entnehmen, dass mit einer Wahrscheinlichkeit von 4/5 die unfaire Münze verwendet wurde und nur zu 1/5 eine der beiden fairen Münzen. Dies ist nun ein eindeutiges Ergebnis, somit wurde zu 80% mit der unfairen Münze dreimal hintereinander Kopf geworfen.

Für eine genaue und auch amüsante Herleitung des Bayes Theorems beziehungsweise dessen Geschichte empfiehlt sich das Buch von Sharon B. McGrayne „The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from two centuries of controversy“ [21].

### 2.3.3 Naïve Bayes-Klassifikation

Die Naïve Bayes-Klassifikation ist die einfachste und am weitesten verbreitete Methode, um einen Text oder Texte zu klassifizieren. Sie berechnet die spätere Wahrscheinlichkeit für eine Klasse, basierend auf der Verteilung der Wörter in dem Dokument, welches klassifiziert werden soll. Dazu wird das Bayes Theorem (welches ausführlich in Abschnitt 2.3.2 erklärt wurde) verwendet und damit die Wahrscheinlichkeit, dass ein gegebenes Dokument „$d$“ zu einer bestimmten Klasse „$c$“ gehört, berechnet [22]. Daraus folgt die Berechnung folgender Wahrscheinlichkeit [28]:

$$P(c \mid d) = \frac{P(d \mid c) \cdot P(c)}{P(d)}. \tag{2.21}$$

Wird Gleichung 2.21 genauer betrachtet, so ist $P(c)$ die Apriori-Wahrscheinlichkeit der Klasse und es ist zu erkennen, dass $P(d)$ keine Rolle bei der Wahl der Klasse „$c^*$“ ($c^* = \arg\max_c P(c \mid d)$ beziehungsweise Gleichung 2.2–2.3) spielt. Um $P(d \mid c)$ zu ermitteln, werden die Apriori-Wahrscheinlichkeiten $f_i$ aufgeteilt und angenommen, dass diese bedingt unabhängig gegenüber der Klasse sind. Aus dieser Aufteilung folgt [28]

$$P_{\mathrm{NB}}(c \mid d) := \frac{P(c)}{P(d)} \cdot \prod_{i=1}^{m} [P(f_i \mid c)]^{n_i(d)}. \tag{2.22}$$

Diese Methode beruht auf relativen Frequenzschätzungen von $P(c)$ und $P(f_i \mid c)$. Trotz der Einfachheit von Naïve Bayes und der Tatsache, dass die bedingte Unabhängigkeit nur als Annahme getätigt wird und nicht in realen Situationen auftritt, liefert der Naïve Bayes Klassifikator überraschend

gute Ergebnisse [16]. Die Annahme der bedingten Unabhängigkeiten von $P(c)$ und $P(f_i \mid c)$ erleichtert das Ermitteln dieser Wahrscheinlichkeitswerte, da jedes dieser Attribute für sich betrachtet werden kann [23, S. 40]. Um die komplexen Gleichungen verständlicher auszudrücken, wird folglich ein kleines Beispiel durchgerechnet.

### Naïve Bayes Klassifikation – ein Beispiel

Bei dem folgenden Beispiel soll simpel eine Klassifikation mit Hilfe des Naïve Bayes Klassifikators durchgeführt werden. Dafür soll ein Wort mit Hilfe der Berechnung als „positiv“ oder „negativ“ eingestuft werden. Angenommen es existieren zwei Klassen

- *positiv* und
- *negativ*,

nun soll das zugehörige Label aus den zwei Klassen für das Wort „fantastisch“ durch die Wahrscheinlichkeitsannahme mittels des Bayes Theorems ermittelt werden. Die Klassifikation als „positiv“ oder „negativ“ ist mit der einfachen Frage

> Ist bei dem Wort „fantastisch“ eher eine „positive“ oder „negative“ Einstufung wahrscheinlicher?

gleichzusetzen. Diese Frage führt wiederum zu der mathematisch dargestellten Frage ob $P(\textit{positiv} \mid \textit{fantastisch})$ oder $P(\textit{negativ} \mid \textit{fantastisch})$ größer ist, welche mathematisch wie folgt ausgedrückt werden:

$$\textit{positiv} = P(\textit{positiv} \mid \textit{fantastisch}) > P(\textit{negativ} \mid \textit{fantastisch}), \tag{2.23}$$

$$\textit{negativ} = P(\textit{positiv} \mid \textit{fantastisch}) < P(\textit{negativ} \mid \textit{fantastisch}). \tag{2.24}$$

Um diese Berechnungen durchzuführen, wird ein umfangreiches Datenset mit bereits klassifizieren Wörtern, in „positiv“ oder „negativ“, benötigt. Für dieses Beispiel wird jedoch nur eine sehr kleine Sammlung verwendet (8 Datensätze), um die Berechnungen etwas verständlicher zu gestalten. Die Wörter des Beispieldatensets und deren zugewiesenes Sentiment können in Tabelle 2.5 eingesehen werden.

Nun kann mit der Berechnung der beiden Wahrscheinlichkeiten für „fantastisch“ $P(\textit{positiv} \mid \textit{fantastisch})$ oder $P(\textit{negativ} \mid \textit{fantastisch})$ begonnen und geprüft werden, welche der beiden Annahmen wahrscheinlicher ist. Diese Wahrscheinlichkeiten können mittels Bayes Theorem, welches in Abschnitt 2.3.2 erklärt wurde und dem Satz von Bayes (Gleichung 2.15), ermittelt werden. Die Wahrscheinlichkeit, zu welcher „fantastisch“ „positiv“ ist, kann wie

**Tabelle 2.5:** Datenset mit bereits „positiv“ oder „negativ“ klassifizierten Wörtern, die aus verschiedenen als „positiv“ oder „negativ“ eingestuften Dokumenten stammen.

| Wort | Klasse |
|---|---|
| fantastisch | negativ |
| super | positiv |
| fantastisch | positiv |
| fantastisch | positiv |
| schlecht | negativ |
| gut | positiv |
| menschlich | positiv |
| traurig | negativ |

folgt ermittelt werden:

$$P(positiv \mid fantastisch) = \frac{P(fantastisch \mid positiv) \cdot P(positiv)}{P(fantastisch)} \tag{2.25}$$

$$= \frac{\frac{2}{5} \cdot \frac{5}{8}}{\frac{3}{8}} = \frac{2}{5} \cdot \frac{5}{8} \cdot \frac{8}{3} = \frac{2}{3}. \tag{2.26}$$

Dafür werden drei unterschiedliche Wahrscheinlichkeitswerte benötigt:

1. die Wahrscheinlichkeit, zu der „fantastisch“ auftritt unter der Bedingung, dass es „positiv“ ist $P(fantastisch \mid positiv)$,
2. die Wahrscheinlichkeit, zu der ein Wort „positiv“ ist $P(positiv)$ und
3. die Wahrscheinlichkeit, dass das Wort „fantastisch“ ist $P(fantastisch)$.

Der erste Wahrscheinlichkeitswert ist simpel die Anzahl von „fantastisch“ als „positiv“ gekennzeichnet, was laut Tabelle 2.5 2 von 5 Mal ist, also 2/5. Der zweite Wert ist die Anzahl der „positiv“ gekennzeichneten Wörter in Relation zu allen Wörtern der Datensammlung, welcher laut Tabelle 2.5 5/8 ist. Die letzte Wahrscheinlichkeit bezieht sich nur auf das Wort an sich, was lediglich die Anzahl des Vorkommens von „fantastisch“ gegenüber aller Wörter ist, also 3/8.

Die Berechnungen für die Wahrscheinlichkeit, dass der Term „fantastisch“ „negativ“ ist, belaufen sich auf folgende Gleichung:

$$P(negativ \mid fantastisch) = \frac{P(fantastisch \mid negativ) \cdot P(negativ)}{P(fantastisch)} \tag{2.27}$$

$$= \frac{\frac{1}{3} \cdot \frac{3}{8}}{\frac{3}{8}} = \frac{1}{3}. \tag{2.28}$$

Hier müssen wiederum drei Wahrscheinlichkeitswerte in Relation gebracht werden:

1. Wahrscheinlichkeit, zu der „fantastisch“ auftritt unter der Bedingung, dass es „negativ“ ist $P(\textit{fantastisch} \mid \textit{negativ})$,
2. Wahrscheinlichkeit, zu der ein Wort „negativ“ ist $P(\textit{negativ})$ und
3. Wahrscheinlichkeit, dass das Wort „fantastisch“ ist $P(\textit{fantastisch})$.

Der erste Wert ist die Anzahl von „fantastisch“ als „negativ“ gekennzeichnet in Relation zu allen „negativ“ gekennzeichneten Wörtern, was laut Tabelle 2.5 $\frac{1}{3}$ ist. Die zweite Wahrscheinlichkeit ist lediglich alle „negativ“ gekennzeichneten Wörter allen Wörtern gegenübergestellt, also 3/8. Der letzte Wert ist gleich wie bei der vorigen Berechnung, die Anzahl des Wortes „fantastisch“ im Vergleich mit allen Wörtern des Datensets, wobei hier eine Wahrscheinlichkeit von 3/8 auftritt. Diese drei Wahrscheinlichkeiten ergeben einen Wahrscheinlichkeitswert von 1/3 zu dem „fantastisch“ „negativ“ ist:

Werden jetzt beide berechneten Wahrscheinlichkeitswerte der Gleichungen 2.26 und 2.28 laut Gleichungen 2.23 und 2.24 verglichen, so stellt sich heraus, dass das Wort „fantastisch“ nach Naïve Bayes als „positiv“ gekennzeichnet wird.

### Das Wort ist nicht im Datenset, was tun?

Angenommen ein Wort, welches nicht im Datenset zum Training des Naïve Bayes Klassifikators enthalten ist, soll einer Klasse zugeordnet werden. Kann eine solche Einteilung überhaupt durchgeführt werden? Die Frage kurz und knapp beantwortet, ja, eine Klassifikation ist möglich. Dazu muss folgende Wahrscheinlichkeitsberechnung der Gleichung 2.21 betrachtet werden: $P(d \mid c)$. Wie bei dem Klassifikationsbeispiel schon erklärt, ist die Berechnung hier die Anzahl des Wortes $d$ als Klasse $c$ gekennzeichnet:

$$P(d_i \mid c_j) = \frac{\text{count}(d_i, c_j)}{\sum\limits_{d \in D} \text{count}(d, c_j)}. \tag{2.29}$$

Wenn das zu klassifizierende Wort nicht im Datenset vorkommt, wäre der Zähler dieser Berechnung 0 und der resultierende Wahrscheinlichkeitswert wäre somit auch 0 und würde die Klasse $c$ als möglichen Klassifikationsausgang eliminieren [37].

Sehen wir uns diesen Fall anhand des bereits bekannten Beispiels an. Dafür wird die Wahrscheinlichkeit berechnet, zu welchem Wert „fantastisch“ positiv ist, also $P(\textit{fantastisch} \mid \textit{positiv})$. Dafür wird aber ein neuer Datensatz benötigt, in dem das Wort „fantastisch“ nicht als „positiv“ gekennzeichnet vorkommt, siehe Tabelle 2.6. Hier sind wiederum acht Wörter enthalten, aber diesmal ist „fantastisch“ nur einmal gelistet und als „negativ“ gekennzeichnet.

**Tabelle 2.6:** Datenset mit bereits „positiv“ oder „negativ“ klassifizierten Wörtern, die aus verschiedenen als „positiv“ oder „negativ“ eingestuften Dokumenten stammen.

| Wort | Klasse |
|---|---|
| fantastisch | negativ |
| super | positiv |
| super | positiv |
| gut | positiv |
| schlecht | negativ |
| gut | positiv |
| menschlich | positiv |
| traurig | negativ |

Nun kann mit den Berechnungen begonnen werden. Zunächst soll das Auftreten von „fantastisch“ als „positiv“ gekennzeichnetes Wort ermittelt werden:

$$P(\mathit{fantastisch} \mid \mathit{positiv}) = \frac{\text{count}(\mathit{fantastisch}, \mathit{positiv})}{\sum_{d \in D} \text{count}(d, \mathit{positiv})} = \frac{0}{5}. \tag{2.30}$$

Hier ist zu erkennen, dass dieser Wert bei 0/5 liegt und somit 0 ergibt.

Wie wirkt sich diese Wahrscheinlichkeit nun auf die Berechnung von $P(\mathit{positiv} \mid \mathit{fantastisch})$ aus? Diese frage kann mit Berechnung folgender berechnung beantwortet werden:

$$P(\mathit{positiv} \mid \mathit{fantastisch}) = \frac{P(\mathit{fantastisch} \mid \mathit{positiv}) \cdot P(\mathit{positiv})}{P(\mathit{fantastisch})} \tag{2.31}$$

$$= \frac{\frac{0}{5} \cdot \frac{5}{8}}{\frac{1}{8}} = 0. \tag{2.32}$$

Es wird aufgezeigt, dass die komplette Wahrscheinlichkeitsberechnung 0 ergibt und somit „positiv“ als Klassenkennzeichnung für „fantastisch“ eliminiert wurde.

Im Falle einer Klassifikation will eine Elimination jedoch vermieden werden. Hierfür gibt es eine simple Lösung: „Laplace (add-1) smoothing“. Es wird lediglich eine 1 der Wahrscheinlichkeit aus Gleichung 2.29 hinzugefügt und ergibt:

$$P(d_i \mid c_j) = \frac{\text{count}(d_i, c_j) + 1}{\sum_{d \in D} (\text{count}(d, c_j) + 1)} = \frac{\text{count}(d_i, c_j) + 1}{(\sum_{d \in D} \text{count}(d, c_j)) + \mid D \mid}. \tag{2.33}$$

Somit ändert sich auch die Berechnung des Wahrscheinlichkeitswertes aus Gleichung 2.30 wie folgt:

$$P(\textit{fantastisch} \mid \textit{positiv}) = \frac{\text{count}(\textit{fantastisch}, \textit{positiv}) + 1}{(\sum_{d \in D} \text{count}(d, \textit{positiv})) + \mid D \mid} = \frac{1}{13} \tag{2.34}$$

Diese Änderung nimmt auch auf die Wahrscheinlichkeitsberechnung aus Gleichung 2.31 Einfluss und lässt auch das Endergebnis anders aussehen:

$$P(\textit{positiv} \mid \textit{fantastisch}) = \frac{P(\textit{fafantastischntastisch} \mid \textit{positiv}) \cdot P(\textit{positiv})}{P(\textit{fantastisch})} \tag{2.35}$$

$$= \frac{\frac{1}{13} \cdot \frac{5}{8}}{\frac{1}{8}} = \frac{1}{13} \cdot \frac{5}{8} \cdot \frac{8}{1} = \frac{5}{13}. \tag{2.36}$$

Durch die abgeänderte Wahrscheinlichkeitsberechnung $P(d \mid c)$ mittels „Laplace (add-1) smoothing“ kann auch für Wörter eine Annahme einer Klassenzugehörigkeit getätigt werden, welche nicht im Datenset, welches für das Trainieren des Klassifikators verwendet wird, enthalten sind.

## 2.4 Andere maschinelle Lernverfahren im Überblick

Die Naïve Bayes Klassifikation ist natürlich nicht das einzige maschinelle Lernverfahren. Es existieren noch zahlreiche andere, davon sind in nachfolgender Liste einige strukturiert angeführt:

- Supervised Learning:
  - Probabilisitic Classifiers (Naïve Bayes Classifier, Bayesian Network, Maximum Entropy Classifier),
  - Linear Classifiers (Support Vector Machine, kurz SVM),
  - Decision Tree Classifiers,
  - Rule-Based Classifiers;
- Weakly, Semi and Unsupervised Learning.

Diese Lernverfahren sollen in diesem Kapitel in folgenden Unterkapiteln kurz erklärt werden. Für einen strukturierten Vergleich dieser Lernverfahren siehe „Sentiment analysis algorithms and applications: A survey“ [22]. Eine wirklich ausführliche Erklärung der in diesem Kapitel kurz beschriebenen Methoden ist in „Machine Learning, Neural and Statistical Classification“ zu finden [23].

### 2.4.1 Supervised Learning

Supervised Learning bedeutet nichts anderes als überwachte Lernmethoden. Diese Gruppe von Methoden ist von kategorisierten Trainingsdaten abhängig. Ohne solche Daten könnte keine Klassifikation durchgeführt werden [22]. Es gibt viele unterschiedliche überwachte Lernmethoden, wobei die für eine Sentimentanalyse am häufigsten verwendeten kurz beschrieben werden. Die Naïve Bayes Klassifikation gehört auch zu diesem Typ von maschinellen Lernverfahren, wurde aber bereits ausführlich erklärt.

#### Probabilistic Classifiers

Bei dieser Klassifikationsmethode wird für die Einstufung in die unterschiedlichen Klassen ein Mischmodell verwendet. Es wird angenommen, dass jede Klasse ein Teil der Mischung ist und die Wahrscheinlichkeit für die Klassifikation eines Terms in die bestimmte Klasse herangezogen wird. Diese Modelle werden oft auch als generative Klassifikatoren bezeichnet [22]. Nachfolgend werden zwei dieser näher erklärt:

- Bayesian Network (BN) und
- Maximum Entropy Classifier (ME).

Der Naïve Bayes Klassifikator gehört zu dieser Gruppe von maschinellen Lernverfahren und wird hier nicht mehr erklärt, da diese Methode schon ausführlich in Abschnitt 2.3 beschrieben wurde.

Das Bayes-Netzwerk-Modell ist ein gerichteter azyklischer Graph, dessen Knoten Zufallsvariablen und Kanten bedingte Abhängigkeiten repräsentieren. Ein solches Netzwerk wird als vollständiges Modell für dessen Variablen und ihre Beziehungen gesehen. Deshalb wird eine vollständige „joint probability distribution“ (JPD) oder auf Deutsch „gemeinsame Wahrscheinlichkeitsverteilung“ über alle Variablen festgelegt. Es wird in der Praxis nicht so häufig eingesetzt, da der Rechenaufwand für dieses System enorm ist [2].

Der Maximum Entropy Classifier ist eine alternative Methode, die sich in einer Reihe von natürlichen Sprachverarbeitungsanwendungen mehr als bewährt hat [4]. Die Wahrscheinlichkeit $P(c \mid d)$ hat folgende Form [28]:

$$P_{\mathrm{ME}}(c \mid d) := \frac{1}{Z(d)} \cdot \exp\Big(\sum_i \lambda_{i,c} F_{i,c}(d,c)\Big), \tag{2.37}$$

wenn $Z(d)$ eine Normalisierungsfunktion ist. $F_{i,c}$ ist ein Feature beziehungsweise eine Klassen-Funktion für $f_i$ und Klasse $c$ und wird folgendermaßen definiert:

$$F_{i,c}(d,c') := \begin{cases} 1 & \text{für } n_i(d) > 0 \text{ und } c' = c, \\ 0 & \text{sonst.} \end{cases} \tag{2.38}$$

Wichtig ist, dass im Gegensatz zu Naïve Bayes, bei der Maximum Entropy Klassifikation keine Annahmen über die Beziehungen zwischen den Features

angestellt werden, somit könnte diese Methode besser als die Naïve Bayes Klassifikation funktionieren, wenn die bedingten Unabhängigkeitsannahmen nicht durchzuführen sind. Der $\lambda_{i,c}$ ist ein Feature-Gewichtungs-Parameter. Aus Gleichung 2.37, die eine Definition von $P_{ME}$ darstellt, ist zu schließen, dass ein großer Wert von $\lambda_{i,c}$ bedeutet, dass $f_i$ ein starker Indikator für Klasse $c$ ist. Die Werte der einzelnen Parameter sind so eingestellt, dass die Entropie der induzierten Verteilung, unter der Bedingung, dass die erwarteten Werte der Features in Bezug auf das Model gleich mit den erwarteten Werten in Bezug auf die Trainingsdaten sind [28].

Eine ausführliche Erklärung der Maximum Entropie Klassifikation ist in „A Maximum Entropy Approach to Natural Language Processing“ zu finden [4, S. 39–71].

### Linear Classifiers

Bei der linearen Klassifikation stehen wieder viele verschiedene Methoden zur Auswahl. Es wird hier auf „Support Vector Machine Classifiers (SVM)“ etwas näher eingegangen, um die Funktionsweise der linearen Klassifikation zu verstehen.

Eine SVM ist eine sehr wirksame Methode der herkömmlichen Textkategorisierung. Im Falle einer Zwei-Klassen-Einstufung ist die Grundidee des Trainingsverfahrens eine Hyperebene, die durch einen Vektor $\boldsymbol{w}$ dargestellt wird, zu finden. Diese soll nicht nur die Dokumentenvektoren in einer Klasse von den anderen trennen, sondern auch die Trennung beziehungsweise den Abstand so groß wie möglich halten. Die daraus resultierende Forderung stellt ein Optimierungsproblem dar: Sei eine Klasse $c_j \in \{-1, 1\}$ (positiv oder negativ) die zugewiesene Klasse zu $d_j$, kann die Lösung mit Hilfe von

$$\boldsymbol{w} := \sum_j \alpha_j \cdot c_j \cdot \boldsymbol{d_j}, \qquad \text{für } \alpha_j \geq 0 \tag{2.39}$$

gefunden werden.

Die Unbekannte $\alpha_j$ kann durch das Lösen des dualen Optimierungsproblems ermittelt werden. Jedoch nur wenn $\alpha_j$ größer als 0 ist, wird das Verfahren „Support Vector Machine“ genannt [28]. Eine ausführliche Erklärung dieser Methode ist in „Incremental and Decremental Support Vector Machine Learning“ [7] zu finden.

### Decision Tree Classifiers

Bei einem „Decision Tree Classifier“ oder zu Deutsch „Entscheidungsbaum Klassifikator“ werden die Trainingsdaten auf hierarchische Art und Weise als Baum dargestellt. Die Unterteilung des Datenbaumes ist rekursiv, bis die Knoten eine bestimmte, minimale Anzahl von Datensätzen enthält, die für die Einstufung verwendet werden [22].

### Rule-Based Classifiers

„Rule-Based Classifier“ bedeutet nichts anderes als „regelbasierter Klassifikator“, wobei bei dieser Methode innerhalb eines Datenraumes verschiedene Regeln definiert werden. Eine solche Regeldefinition teilt sich in folgende zwei Bereiche:

- die Bedingung, welche das Feature erfüllt beziehungsweise nicht erfüllt, die in disjunktiver Normalform[10] ausgedrückt wird und
- die Klasse, in welche das Feature eingeordnet gehört, wenn die Bedingung erfüllt wird.

Eine solche Bedingung wird nur mit der Prüfung von existierenden Termen gebildet, eine Inexistenz eines Terms wird so gut wie nie in die Bedingung aufgenommen, da eine solche Information nicht informativ ist [22].

Um einen regelbasierten Klassifikator korrekt zu erstellen, werden mehrere Schritte benötigt. Der komplette Algorithmus für diesen Vorgang wird „CBA“ (Classification Based on Associations) genannt und besteht grundsätzlich aus zwei Teilen:

- Regel-Generierung und
- Klassifikator-Generierung.

Um eine Regel zu erstellen, müssen viele unterschiedliche Kriterien beachtet werden, auf welche hier nicht näher eingegangen wird. In der Trainingsphase werden die Regeln unter Einhaltung dieser konstruiert. Die Kriterien zur Regelbildung können unter „Integrating Classification and Association Rule Mining“ in Kapitel „Generating the Complete Set of CARs“ eingesehen werden [17].

## 2.4.2 Weakly, Semi and Unsupervised Learning

Ziel der Text-Kategorisierung ist es, Dokumente einer bestimmten Klasse, aus einem definierten Set von Klassen, zuzuordnen. In vielen Arbeiten wird eine große Datenmenge mit bereits zu einer Klasse zugehörigen Datensätzen verwendet, um „Supervised Learning“ Methoden für die Klassifikation einzusetzen, welche bereits in Abschnitt 2.4.1 erklärt wurden. Die Beschaffung von gekennzeichneten Trainingsdaten führt in vielen Fällen zu einem Problem, wobei es im Gegenzug sehr einfach ist, ungekennzeichnete Daten aufzutreiben. Die keiner Klasse zugeordneten Daten lassen wiederum ein nächstes Problem auftreten: Die manuelle Klassifikation dieser. Um auch diesem Hindernis entgegenzusteuern, wird auf „unsupervised learning“ Methoden zurückgegriffen [15, 25].

[10] DNF, eine Gleichung die aus Disjunktionen von Konjunktionstermen besteht. Die Gleichung besteht (auf oberster Ebene) nur aus „ODER“-Verknüpfungen. Mehr dazu ist unter http://www.mathepedia.de/Disjunktive_Normalform.aspx zu finden.

So wird beispielsweise in „Automatic Text Categorization by Unsupervised Learning“ [15] eine Methode beschrieben, bei der die einzelnen Dokumente der Trainingsdaten in Sätze aufgeteilt werden. Danach wird jeder dieser Sätze anhand einer Schlüsselwörterliste für jede Kategorie und einer Ähnlichkeitsanalyse der Sätze kategorisiert. Diese zuvor gekennzeichneten Sätze werden für das Training des Klassifikators verwendet [15].

Mehr Informationen über diesen Typ der Klassifikation können aus „Sentiment analysis algorithms and applications: A survey“ [22] entnommen werden.

## 2.5 Textverarbeitung und Textaufbereitung

Bei der Verarbeitung und Aufbereitung von Texten handelt es sich um zwei unterschiedliche Verfahren, die in verschiedensten Bereichen Anwendung finden. In Bezug auf diese Arbeit und die durchgeführte Sentimentanalyse sind die beiden Terme lediglich bei

- den zum Training herangezogenen und
- den zu klassifizierenden Texten

relevant. Außerdem ist der Unterschied dieser von Bedeutung.

Textverarbeitung bedeutet nichts anderes als Texte mittels eines rationalisierten Verfahrens zu formulieren, diktieren, schreiben, vervielfältigen oder ähnliches [41]. Mit Textaufbereitung ist gemeint, die großen Textmengen so zu bearbeiten, dass sie effizient zum Trainieren des Naïve Bayes Klassifikators und zur Klassifikation verwendet werden können. Hierzu spielen die nachfolgend erklärten Verfahren eine große Rolle.

### 2.5.1 Sprachidentifikation

Die Sprachidentifikation an sich gehört nicht direkt zur Textaufbereitung, allerdings gewinnt sie an Wichtigkeit, wenn riesige Mengen an Text, in derselben Sprache, automatisiert verwendet werden sollen. Um dies zu ermöglichen, muss die Sprache des Textes ermittelt werden. Ein Mensch kann beim Lesen die Sprache erkennen, sofern er dieser mächtig ist. Sollen aber mehrere tausend Texte identifiziert werden, ist eine manuelle Erkennung extrem zeitintensiv und aufwendig. Schon vor über 50 Jahren wurde aus diesem Problem heraus Forschung in dem Bereich der automatisierten Sprachidentifizierung betrieben [30].

Mittels der Wissenschaft entstanden verschiedene Ansätze, die Sprache eines geschriebenen Textes zu ermitteln und es stellte sich heraus, dass es eine Form von Text-Klassifikation ist. Die am meisten verbreitete Lösung ist die Verwendung von maschinellen Lernverfahren, genauer genommen SVMs (support vector machines), welche bereits in Abschnitt 2.4.1 etwas genauer beschrieben wurden. Um ein maschinelles Lernverfahren zu verwenden, wird

wiederum ein Datenset mit bereits klassifizierten Wörtern in deren Sprache benötigt. Dazu werden oft eines oder mehrere der folgenden Datensammlungen verwendet [6]:

- International Corpus of Learner English [13] (Englisch),
- Cambridge Learner Corpus [35] (Englisch) und
- Falko Corpus [18] (Deutsch).

Am meisten verwendet wird allerdings das erste Datenset der Liste, was vermutlich auf die kostenpflichtige Verwendung der zweiten Sammlung zurückzuführen ist. Das dritte Set ist nur mit Wörtern in deutscher Sprache verfügbar. Solange eine Klassifikation lediglich mit einer einzelnen Sprache durchgeführt werden soll, funktioniert dieser Ansatz. Bei Dokumenten, die mehrere unterschiedliche Sprachen enthalten, tritt allerdings ein Problem bei der Klassifikation auf. Hierzu wird beispielsweise von Lui, Han Lau und Baldwin eine Lösung geboten [19]. Es wird eine „multi-label“ Klassifikation durchgeführt, wobei für ein Dokument alle enthaltenen Sprachen identifiziert und deren Anteile ermittelt werden. Für die genaue Funktionsweise dieser Lösung siehe „Automatic Detection and Language Identification of Multilingual Documents“ [19].

### 2.5.2 Features und N-Gramme

In dem Bereich der elektronischen Datenverarbeitung wird ein Feature als ein besonderes Merkmal für etwas angesehen, das eine bestimmte Sache auszeichnet [41]. Deshalb werden Features auch für die Sentimentanalyse verwendet, um Texte zu klassifizieren. Somit existieren Features für „positive“ Texte oder Textteile und natürlich auch welche für „negative“.

In der Textverarbeitung treten Features als Vektoren auf, welche aus geschlossenen Texten beziehungsweise einer Nachricht erstellt werden. Wichtig ist die richtige Auswahl der Features, was vor allem beim Maschinellen Lernen eine bedeutende Rolle spielt [11]. Jeder Term des Feature Vektors besitzt eine eigene „frequency“, also die Auftrittshäufigkeit des Terms im Text. In manchen Fällen wird nicht auf die Häufigkeit an sich gesetzt, sondern nur auf das Auftreten oder Nichtauftreten eines Wortes. Hier wird ein Term des Feature Vektors nur mit „true“ (das Wort ist im Text enthalten) oder „false“ (das Wort ist nicht im Text enthalten) gekennzeichnet [27, 28].

Wie schon zuvor erwähnt, wird ein Feature Vektor aus einem Text generiert, wobei ein Feature Term unterschiedlich aussehen kann. Kurz gesagt, aus dem Text werden „n-Gramme“ geformt und diese prägen dann den Term des Feature Vektors. Für jedes n-Gramm wird dann die Häufigkeit des Auftretens ermittelt.

Ein n-Gramm ist eine „n“-lange Wortsequenz, die aus einem Text heraus erstellt wird, allerdings beginnt die erste Sequenz mit dem ersten Wort des Textes und ist dann „n“ Wörter lang. Die zweite erstellte Wortfolge beginnt

**Tabelle 2.7:** Unigramme, Bigramme und Trigramme für die Textsequenz „Das ist ein Beispiel für n-Gramme“.

| Unigramm | Bigramm | Trigramm |
|---|---|---|
| Das | Das ist | Das ist ein |
| ist | ist ein | ist ein Beispiel |
| ein | ein Beispiel | ein Beispiel für |
| Beispiel | Beispiel für | Beispiel für n-Gramme |
| für | für n-Gramme | |
| n-Gramme | | |

mit dem zweiten Wort und weist wieder eine Länge von „n“ Wörtern auf. Dies wird bis zu dem Wort fortgeführt, mit welchem das letzte Mal die „n“ lange Sequenz vervollständigt werden kann. Üblicherweise werden die Sequenzen für ein Wort „Unigramm“, für zwei Wörter „Bigramm“, für drei Wörter „Trigramm“ bezeichnet und für die restlichen höheren n-Gramme jeweils die Länge als Zahl, also „4-Gramm“, „5-Gramm“, ... [36]. Als Beispiel für Unigramme, Bigramme und Trigramme eines kurzen Satzes siehe Tabelle 2.7.

Es ist nicht immer von Vorteil die n-Gramme höherer Ordnung zu verwenden, da die Performance nicht stets, die mit niedriger Ordnung übertrifft. Werden beispielsweise Kinofilmkritiken als „positiv“ oder „negativ“ klassifiziert, so liefern Unigramme bessere Ergebnisse als Bigramme [28], aber bei der Klassifikation von Produktbewertungen sind Bigramme und Trigramme den Unigrammen stark überlegen [10].

Des Weiteren werden nicht immer alle Features verwendet, sondern nur bestimmte ausgewählt. In vielen Studien werden beispielsweise nur Adjektive verwendet, da diese wirklich akkurate Ergebnisse bei einer Sentimentanalyse liefern [27].

### 2.5.3 Textnormalisierung

Heutzutage werden sehr viele Texte online erfasst und publiziert, wobei Blogs, Soziale Medien und Foren dominieren. In diesen Bereichen wird aber oft auf die standardisierte Sprache verzichtet und Rechtschreibung, Grammatik und richtige Satzzeichensetzung ignoriert. Auch Abkürzungen treten stark in den Vordergrund. Der Vorgang, solche Texte wieder in die Standardsprache zu bringen, wird „Text Normalisierung“ genannt [32]. In verschiedenen Studien wurde versucht, eine vollautomatisierte, statistische Lösung zu bieten, die nicht standardisierte, sprich regionsbezogene oder in Mundart verfasste Wörter, wieder in die Normalform bringt. Die Forschung in diesem

Bereich konzentriert sich oftmals auf die (Wieder)-Erkennung und Übersetzung beziehungsweise Transliteration von Eigennamen [8]. Bereits im Jahr 1999 wurde in „Normalization of non-standard words“ [32] versucht, auf Akronyme und Abkürzungen Rücksicht zu nehmen, allerdings wurden Jargon und saloppe Wörter außer Acht gelassen. Außerdem heben sie heraus, dass die Text Normalisierung kein Problem ist, welchem viel Aufmerksamkeit zugesprochen wird, sondern eher als lästige Pflicht angesehen wird [32].

Schon in den frühen 90er Jahren wurde versucht, dieses Problem zu lösen, indem Rechtschreibfehler automatisch korrigiert werden sollten. Hierfür wurde das „Noisy Channel“-Modell herangezogen, welches auch 1991 von IBM verwendet wurde, siehe „Context based spelling correction“ [20]. Später wurde dann der Versuch gestartet, nicht nur die Rechtschreibung zu berücksichtigen, sondern auch andere Faktoren wie Abkürzungen, Akronyme, Satzzeichen, Jargon, Wortspiele und noch einige andere [38]. Dazu wurde beispielsweise im Jahr 2011 eine Lösung in „Text normalization in social media: progress, problems and applications for a pre-processing system of casual English“ [8] vorgestellt, wo ein „Casual English classification system (CECS)“ entwickelt wurde.

### Noisy Channel Modell

Bei diesem Modell wird für ein falsch geschriebenes Wort eine Liste von möglichen Kandidaten als richtig geschrieben ermittelt, wobei für jeden Kandidat berechnet wird, wie wahrscheinlich es ist, dass dieser gemeint war und nicht das falsch geschriebene Wort. Zum Ermitteln der Kandidaten wird eine „minimale Bearbeitungsdistanz“ zwischen dem Wort und dem Kandidaten ermittelt. Diese Distanz ist durch folgende Arbeitsschritte geprägt:

- Einfügen eines Buchstaben,
- Löschen eines Buchstaben,
- Austauschen eines Buchstaben mit einem anderen und
- Vertauschen zweier benachbarter Buchstaben [38].

Dieses Modell wird von Dan Jurafsky in einem Video sehr gut erklärt, siehe https://class.coursera.org/nlp/lecture/22 [38].

### CECS von Clark und Araki

Dieses System ist für Fehler und die nicht standardisierte englische Sprache, die vor allem in Sozialen Medien auftritt, entwickelt worden. Ein mehrdimensionaler Ansatz wurde verwendet, um diese Texte wieder in Hochsprache zu bringen. Dazu wurden verschiedene Kategorien entwickelt [8]:

1. Abkürzungen (Kurzformen): „gn8“ wird zu „gute Nacht“.
2. Abkürzungen (Akronyme): „lol“ wird zu „laugh out loud“.
3. Tipp- oder Rechtschreibfehler: „wilk“ wird zu „will“.

4. Satzzeichenfehler: vor allem im Englischen bei „im“ statt „i'm“.
5. Slang (nicht im Wörterbuch): „fix oida“ wird zu „sicher alter“.
6. Wortspiele: „das war sooooo gut“ wird zu „das war so gut“.
7. Zensur: Wortendungen die durch „*“ oder andere Zeichen ersetzt wurden, werden wieder in die Ursprungsform gebracht.
8. Emoticons: „:)“ wird zu „lachendes Gesicht“ oder „<3“ wird zu „Herz“.

### 2.5.4 Stemming

Das Stemming wird verwendet, um den Kern eines Wortes zu ermitteln. Hierzu werden die Affixe eines Wortes abgetrennt und ignoriert. Dieser Prozess ist somit stark von der Sprache des Textes beziehungsweise der Wörter abhängig [39]. So wird beispielsweise aus dem Wort „lieben“ – nach dem Stemming – „lieb“, aus „meins“ wird „mein“ und aus „automatisch“ oder „automatik“ wird „automat“.

Der einfachste und am häufigsten verwendete Stemmer für Englisch und Deutsch ist „Porter's“ Stemming-Algorithmus. Hier werden für jedes Wort verschiedene Regeln abgearbeitet und so entschieden wo „gestemmt“ wird. Eine genauere Erklärung der Funktionsweise dieses Stemmingverfahrens ist in dem Video von Dan Jurafsky über „Word Normalization and Stemming“ unter https://class.coursera.org/nlp/lecture/126 zu finden [39]. Auch wirklich gut erklärt wird das Porter's Stemmingverfahren in der Präsentation von Éva Mújdricza und Ganna Syrota [44].

### 2.5.5 Verneinungen

Verneinungen in Sätzen spielen eine wichtige Rolle bei einer Sentimentanalyse, da beispielsweise ein „nicht“ der Aussage eine komplett andere Bedeutung geben kann. Bei der Erstellung von Features müsste somit bei Wörtern, die oft in Sätzen mit Verneinungen vorkommen, ein „nicht“ angehängt und es zusätzlich in den Feature Vektor aufgenommen werden [9]. So würde beispielsweise der Satz „Ich mag mein neues Handy nicht“ in Unigramme aufgeteilt und zusätzlich würde noch „mag-NICHT“ als Feature generiert werden.

Zusätzliche Schwierigkeiten bringen Verneinungen in eine Sentimentanalyse, da mit ihnen oft Sarkasmus und Ironie ausgedrückt wird [27]. Auf die Sarkasmus-Erkennung in Texten wird in Abschnitt 2.5.6 etwas näher eingegangen.

### 2.5.6 Sarkasmus Erkennung

Sarkasmus an sich ist eine scharfe, bittere oder schneidende Bemerkung, welche Spott oder Hohn zum Ausdruck bringt. Die Qualität von Sarkasmus ist in dem gesprochenen Wort enthalten und wird durch den Tonfall

manifestiert. Die Inhalte hängen stark vom Kontext der Situation ab [14].

Zum Beispiel könnte die Nachricht „Gab es das Kleid nicht in deiner Größe?“ eine sarkastische Aussage sein, wenn eine mollige Frau ihr neues Outfit präsentiert und eine andere Person mit dieser Frage darauf reagiert. Die Frau wird beleidigt und so ist der Sarkasmus in der Aussage klar [47].

Noch dazu muss zwischen Sarkasmus und Ironie unterschieden werden, wobei Ironie als Stilmittel und oftmals im Zusammenhang mit Sarkasmus verwendet wird. Bei einer ironischen Aussage wird der Umstand durch das Gegenteil ausgedrückt. Hier könnte die sarkastische und ironische Aussage „Überarbeiten Sie sich nicht!“ als solche gelten, wenn der Chef dies zu seinem Mitarbeiter sagt, der untätig dasitzt [47].

Die Erkennung von Sarkasmus in Tweets ist ein Textanalyse-Problem, welches mit sehr viel Training und Aufwand verbunden ist. Da Twitter Nachrichten sehr kurz sind (max 140 Zeichen), ist es nicht einfach, Sarkasmus zu erkennen. Bei einigen Nachrichten müssen Meta-Informationen der Tweets verwendet werden. Die Methode von Ingle u. a. ist für die Erkennung von Sarkasmus in Tweets zugeschnitten und besteht aus folgenden Schritten:

- Lexikalische Analyse mit
  - Hashtags, die Sarkasmus ausdrücken (#Sarkasmus, . . . ) und
  - populären Phrasen, die Sarkasmus ausdrücken,
- Analyse der Likes und Dislikes von Tweets,
- Faktenverdrehung (Nachrichten, die Tatsachen widersprechen) und
- zeitbasierte Nachrichten, die Events verfälschen (Nachrichten, die ein zeitbasiertes Event als falsch darstellen).

Für eine genaue Beschreibung der Methodik siehe „Sentiment Analysis: Sarcasm Detection of Tweets“ Kapitel 4 [14].

### 2.5.7 Stoppwörter

Ein Stoppwort ist ein Wort, welches hochfrequent in einem Text vorkommt und wenig lexikalische Inhalte aufweist. Rein durch ihre Anwesenheit in einem Text wird es schwieriger, ihn von anderen zu unterscheiden. Deshalb ist es meist von Vorteil, diese Wörter aus einem Text herauszufiltern. Stoppwörter können von jedem Anwender in der Regel selbst definiert werden, allerdings hat sich herausgestellt, dass die dadurch entstandenen Stoppwortlisten immer ähnlich sind. Deshalb gibt es bereits unzählige vorgefertigte Listen. Auch das Natural Language Processing Toolkit biete solch eine Liste an. Die englische Liste beinhaltet beispielsweise die Worte „a“, „all“, „also“, „and“, „as“, „both“, „each“, „few“, „how“, „i“, „it“, „its“, „me“, „my“, „no“, „our“, „them“, „we“, „you“ und „your“ [5]. Eine vollständige Liste der englischen Stoppwortliste des NLTK ist unter http://www.nltk.org/book/ch02.html zu finden.

# Kapitel 3

# Eigener Ansatz für die Verbesserung der Sentimentanalyse

Um die Sentimentanalyse für Nachrichten von Twitter zu verbessern, wurden unterschiedliche „Designs“ für das Klassifikations-Modell verwendet. Für die Designs an sich können folgende Parameter unterschiedlich sein:

- Trainingsdaten- und Testdaten-Set (nur Tweets, nur Movie-Reviews oder Tweets und Movie-Reviews in Kombination),
- Datenaufbereitung (Stemming oder kein Stemming),
- Tokenisierung (n-Gramme),
  - Unigramme (alle oder $10,000$ mit den meisten Informationen),
  - Bigramme (alle oder 200 mit den meisten Informationen) und
  - Unigramme und Bigramme (alle oder $10,000$ Unigramme und 200 Bigramme mit den meisten Informationen).

Durch diese unterschiedlichen Möglichkeiten, das Klassifikations-Modell zu generieren, können viele verschiedene Tests durchgeführt werden, um zu ermitteln, welches Design die lukrativsten Ergebnisse liefert. Allerdings soll sich auf die Klassifikation mit den Tweets und Movie-Reviews als Trainigsdaten und Tweets als Testdaten konzentriert werden, da versucht werden soll, durch eine Kombination unterschiedlicher Trainingsdaten eine Verbesserung der Sentimenterkennung zu erreichen. Bei *allen* Klassifikations-Modellen wird das Naïve Bayes Verfahren verwendet, um das Label (positiv oder negativ) für die jeweilige Nachricht zu ermitteln.

**Tabelle 3.1:** Tabelle mit allen Applikations-Designs der Trainingsdaten separat und deren Unterschiede. T = Tweets, Uni = Unigramm, Bi = Bigramm, alle = Verwendung der Features von allen Trainingsdaten, teil = Verwendung der Features mit den meisten Informationen.

| Dateiname | Trainings- | Test-daten | n-gram | Feat |
|---|---|---|---|---|
| sa_T_u_all | 3000 T | 1000 T | Uni | alle |
| sa_T_u_sel | 3000 T | 1000 T | Uni | teil |
| sa_T_b_all | 3000 T | 1000 T | Bi | alle |
| sa_T_b_sel | 3000 T | 1000 T | Bi | teil |
| sa_T_ub_all | 3000 T | 1000 T | Uni, Bi | alle |
| sa_T_ub_sel | 3000 T | 1000 T | Uni, Bi | teil |

## 3.1 Die Klassifikations-Modelle

Wie bereits erwähnt, werden für den Versuch zur Verbesserung der Sentimentanalyse bei Nachrichten von Twitter verschiedene Klassifikations-Modelle erstellt und ausgetestet, um zu sehen wie sich die einzelnen Parameter auf die Ergebnisse auswirken. Diese können in unterschiedliche Kategorien eingestuft und somit unterschieden werden. Es wird zunächst bei der Verwendung von

- Tweets als Trainingsdaten (siehe Tabelle 3.1),
- Movie-Reviews als Trainingsdaten (siehe Tabelle 3.2) und
- Tweets und Movie-Reviews als Trainingsdaten (siehe Tabelle 3.3)

differenziert.

Bei dem Klassifikations-Modell trainiert mit Tweets wurde lediglich auf die Einstufung – in positiv oder negativ – von Tweets getestet. Es wurden sechs verschiedene Modelle erstellt, welche in der Verwendung von Unigrammen oder Bigrammen oder beiden in Kombination (Tweets und Movie-Reviews) Unterschiede aufweisen. Des Weiteren werden die Modelle mit allen Trainingsdaten (allen Unigrammen oder Bigrammen oder beiden) und einem Teil der Daten (die N-Gramme mit den meisten Informationen) erstellt und ausprobiert. Eine kurze Übersicht, der Modell-Designs für das Klassifikations-Modell trainiert und getestet mit Tweets, ist in Tabelle 3.1 vorhanden.

Bei den Sentimentanalyse-Modellen erstellt und getestet mit Movie-Reviews wird ebenso ein Unterschied bei der Verwendung von Unigrammen, Bigrammen und dessen Kombination gemacht. Des Weiteren wird genauso die Verwendung von allen n-Grammen oder nur jene mit den meisten In-

**Tabelle 3.2:** Tabelle mit allen Applikations-Designs der Trainingsdaten separat und deren Unterschiede. R = Movie-Reviews, Uni = Unigramm, Bi = Bigramm, alle = Verwendung der Features von allen Trainingsdaten, teil = Verwendung der Features mit den meisten Informationen.

| Dateiname | Trainings-daten | Test-daten | n-gram | Feat |
|---|---|---|---|---|
| sa_R_u_all | 1500 R | 500 R | Uni | alle |
| sa_R_u_sel | 1500 R | 500 R | Uni | teil |
| sa_R_b_all | 1500 R | 500 R | Bi | alle |
| sa_R_b_sel | 1500 R | 500 R | Bi | teil |
| sa_R_ub_all | 1500 R | 500 R | Uni, Bi | alle |
| sa_R_ub_sel | 1500 R | 500 R | Uni, Bi | teil |

formationen differenziert. Hier ist zu erwähnen, dass insgesamt 2000 Movie-Reviews aus der bereits vorhandenen Sammlung des Natural Language Processing Toolkits verwendet wurden. Das Set teilt sich in zwei Teile, wobei der erste Teil 3/4 der Daten zum Training verwendet, was 1500 Reviews entspricht, und das verbleibende 1/4 (500 Reviews) den zweiten Teil bildet und für die Tests herangezogen wird. Auch für diese Klassifikations-Modelle ist in Tabelle 3.2 eine kurze Übersicht gegeben.

Nach der Erstellung der Sentimentanalyse-Applikationen, die mit Tweets und Movie-Reviews trainiert und getestet wurden, sind diese beiden Datensets kombiniert und daraus neue Klassifikations-Modelle erstellt und getestet worden. Die neuen Applikationen wurden jeweils mit 3000 Tweets und 1500 Movie-Reviews trainiert und mit folgenden unterschiedlichen Datensammlungen getestet:

- Test mit 1000 Tweets,
- Test mit 500 Movie-Reviews und
- Test mit 1000 Tweets und 500 Movie-Reviews.

Außerdem werden hier, wie bei den vorherigen Klassifikations-Modellen, bei der Erstellung Unterschiede zwischen der Verwendung von Unigrammen, Bigrammen und dessen Kombination beim Training und Testen gemacht. Der Gebrauch von allen n-Grammen ist von der Verwendung der Features mit den meisten Informationen zu differenzieren. Es entstanden 18 unterschiedliche Applikationen, die logischerweise auch unterschiedliche Testergebnisse liefern werden. Eine Übersicht aller Applikationen trainiert mit einem kombinierten Datenset aus Tweets und Movie-Reviews ist in Tabelle 3.3 zu finden.

**Tabelle 3.3:** Tabelle mit allen Applikations-Designs der kombinierten Trainingsdaten und deren Unterschiede. Modell trainiert mit 3000 Tweets und 1500 Movie-Reviews. T = Tweets, R = Movie-Reviews (Filmkritiken), Uni = Unigramm, Bi = Bigramm, alle = Verwendung der Features von allen Trainingsdaten, teil = Verwendung der Features mit den meisten Informationen.

| Dateiname | Testdaten | n-gram | Features |
|---|---|---|---|
| sa_TR_u_all | 1000 T | Uni | alle |
| sa_TR_u_sel | 1000 T | Uni | teil |
| sa_TR_b_all | 1000 T | Bi | alle |
| sa_TR_b_sel | 1000 T | Bi | teil |
| sa_TR_ub_all | 1000 T | Uni, Bi | alle |
| sa_TR_ub_sel | 1000 T | Uni, Bi | teil |
| sa_TR_u_all | 500 R | Uni | alle |
| sa_TR_u_sel | 500 R | Uni | teil |
| sa_TR_b_all | 500 R | Bi | alle |
| sa_TR_b_sel | 500 R | Bi | teil |
| sa_TR_ub_all | 500 R | Uni, Bi | alle |
| sa_TR_ub_sel | 500 R | Uni, Bi | teil |
| sa_TR_u_all | 1000 T + 500 R | Uni | alle |
| sa_TR_u_sel | 1000 T + 500 R | Uni | teil |
| sa_TR_b_all | 1000 T + 500 R | Bi | alle |
| sa_TR_b_sel | 1000 T + 500 R | Bi | teil |
| sa_TR_ub_all | 1000 T + 500 R | Uni, Bi | alle |
| sa_TR_ub_sel | 1000 T + 500 R | Uni, Bi | teil |

Bei all den bisher erstellten Klassifikations-Modellen für eine Sentimentanalyse bei Tweets wurde kein Stemming angewandt. Allerdings soll auch der Einfluss dieser Textaufbereitungsmethode auf die Performance der Einteilung in positiv oder negativ untersucht werden, weshalb weitere Sentimentanalyse-Applikationen erstellt wurden. Diese Modelle wurden nur für das Training mit Tweets und dem kombinierten Set aus Tweets und Movie-Reviews erstellt und auch nur auf die Klassifikation von Tweets getestet. Somit ergeben sich die Applikationen, die in Tabelle 3.4 aufgelistet sind. Für eine Erklärung von Stemming siehe Abschnitt 2.5.4.

Es wurden 12 Sentimentanalyse-Applikationen erstellt, bei denen auf die Trainings- und Testdaten Stemming angewandt wurde. Diese Modelle

**Tabelle 3.4:** Tabelle mit allen Applikations-Designs der kombinierten Trainingsdaten und deren Unterschiede. Modell trainiert mit 3000 Tweets und 1500 Movie-Reviews, wobei auf die Trainings- und Testdaten Stemming angewandt wurde. T = Tweets, R = Movie-Reviews (Filmkritiken), Uni = Unigramm, Bi = Bigramm, alle = Verwendung der Features von allen Trainingsdaten, teil = Verwendung der Features mit den meisten Informationen.

| Dateiname | Trainingsdaten | n-gram | Features |
|---|---|---|---|
| sa_T_u_all_s | 3000 T | Uni | alle |
| sa_T_u_sel_s | 3000 T | Uni | teil |
| sa_T_b_all_s | 3000 T | Bi | alle |
| sa_T_b_sel_s | 3000 T | Bi | teil |
| sa_T_ub_all_s | 3000 T | Uni, Bi | alle |
| sa_T_ub_sel_s | 3000 T | Uni, Bi | teil |
| sa_TR_u_all_s | 3000 T + 500 R | Uni | alle |
| sa_TR_u_sel_s | 3000 T + 500 R | Uni | teil |
| sa_TR_b_all_s | 3000 T + 500 R | Bi | alle |
| sa_TR_b_sel_s | 3000 T + 500 R | Bi | teil |
| sa_TR_ub_all_s | 3000 T + 500 R | Uni, Bi | alle |
| sa_TR_ub_sel_s | 3000 T + 500 R | Uni, Bi | teil |

unterscheiden sich wiederum in den Trainingsdaten folgendermaßen:

- Training mit 3000 Tweets und
- Training mit 3000 Tweets und 1500 Movie-Reviews.

Des Weiteren ist bei dem Gebrauch von Unigrammen, Bigrammen oder beiden in Kombination zu differenzieren. Es wird auch bei der Verwendung von allen Features oder nur jene mit den meisten Informationen ein Unterschied gemacht.

Bei allen bisherigen Sentimentanalyse-Applikationen wurden die Stoppwörter in den Trainings- und Testdaten belassen. Diese Wörter haben aber meist keine Informationen beziehungsweise Aussagekraft über das Sentiment eines Textes. Somit sollte auch bei Tweets getestet und ermittelt werden, ob die Entfernung von Stoppwörtern Einfluss auf die Performance der Klassifikation hat.

Es wurden, wie auch beim Stemming, 12 unterschiedliche Sentimentanalyse-Applikationen erstellt, bei denen die Trainings- und Testdaten von allen Stoppwörtern befreit wurden. Diese Modelle unterscheiden sich wiederum in den Trainingsdaten, wie jene Applikationen, auf welche Stemming angewandt wurde, wie folgt:

**Tabelle 3.5:** Tabelle mit allen Applikations-Designs der kombinierten Trainingsdaten und deren Unterschiede. Modell trainiert mit 3000 Tweets und 1500 Movie-Reviews, wobei auf Trainings- und Testdaten von Stoppwörtern befreit wurden. T = Tweets, R = Movie-Reviews (Filmkritiken), Uni = Unigramm, Bi = Bigramm, alle = Verwendung der Features von allen Trainingsdaten, teil = Verwendung der Features mit den meisten Informationen.

| Dateiname | Trainingsdaten | n-gram | Features |
|---|---|---|---|
| sa_T_u_all_st | 3000 T | Uni | alle |
| sa_T_u_sel_st | 3000 T | Uni | teil |
| sa_T_b_all_st | 3000 T | Bi | alle |
| sa_T_b_sel_st | 3000 T | Bi | teil |
| sa_T_ub_all_st | 3000 T | Uni, Bi | alle |
| sa_T_ub_sel_st | 3000 T | Uni, Bi | teil |
| sa_TR_u_all_st | 3000 T + 500 R | Uni | alle |
| sa_TR_u_sel_st | 3000 T + 500 R | Uni | teil |
| sa_TR_b_all_st | 3000 T + 500 R | Bi | alle |
| sa_TR_b_sel_st | 3000 T + 500 R | Bi | teil |
| sa_TR_ub_all_st | 3000 T + 500 R | Uni, Bi | alle |
| sa_TR_ub_sel_st | 3000 T + 500 R | Uni, Bi | teil |

- Training mit 3000 Tweets und
- Training mit 3000 Tweets und 1500 Movie-Reviews.

Es werden bei diesen Applikationen auch Unigramme, Bigramme, die Kombination dieser und die Verwendung von allen Features oder nur jene mit den meisten Informationen unterschieden. In Tabelle 3.5 sind alle Sentimentanalyse-Applikationen, mit einer kurzen Beschreibung der Parameter aufgelistet, bei denen Stoppwörter entfernt wurden.

Um auch den Einfluss von Stemming und die Entfernung von Stoppwörtern zu ermitteln, werden wiederum 12 eigene Sentiment-analyse-Applikationen erstellt, bei denen diese zwei Parameter berücksichtigt wurden und deren Auswirkungen ermittelt werden. Somit ergeben sich die in Tabelle 3.6 aufgelisteten Sentimentanalyse-Modelle zur Klassifikation. Hier wurde wiederum bei der Verwendung von Unigrammen, Bigrammen und dessen Kombination unterschieden, sowie das Heranziehen aller n-Gramme oder nur jene mit den meisten Informationen.

Abschließend ist noch zu erwähnen, dass die Trainings- und Testdaten in der Verwendung von Unigrammen, Bigrammen oder beiden immer gleich sein müssen. Dies ist anhand eines kleinen Beispieles einfacher zu verstehen:

**Tabelle 3.6:** Tabelle mit allen Applikations-Designs der kombinierten Trainingsdaten und deren Unterschiede. Modell trainiert mit 3000 Tweets und 1500 Movie-Reviews, wobei auf Trainings- und Testdaten von Stoppwörtern befreit wurden und Stemming angewandt wurde. T = Tweets, R = Movie-Reviews (Filmkritiken), Uni = Unigramm, Bi = Bigramm, alle = Verwendung der Features von allen Trainingsdaten, teil = Verwendung der Features mit den meisten Informationen.

| Dateiname | Trainingsdaten | n-gram | Features |
|---|---|---|---|
| sa_T_u_all_ss | 3000 T | Uni | alle |
| sa_T_u_sel_ss | 3000 T | Uni | teil |
| sa_T_b_all_ss | 3000 T | Bi | alle |
| sa_T_b_sel_ss | 3000 T | Bi | teil |
| sa_T_ub_all_ss | 3000 T | Uni, Bi | alle |
| sa_T_ub_sel_ss | 3000 T | Uni, Bi | teil |
| sa_TR_u_all_ss | 3000 T + 500 R | Uni | alle |
| sa_TR_u_sel_ss | 3000 T + 500 R | Uni | teil |
| sa_TR_b_all_ss | 3000 T + 500 R | Bi | alle |
| sa_TR_b_sel_ss | 3000 T + 500 R | Bi | teil |
| sa_TR_ub_all_ss | 3000 T + 500 R | Uni, Bi | alle |
| sa_TR_ub_sel_ss | 3000 T + 500 R | Uni, Bi | teil |

Angenommen zum Training werden nur Unigramme verwendet und aus den Testdaten werden lediglich nur Bigramme erstellt, dann sind jene Features, die aus den Testdaten erstellt wurden nie in den Features aus den Trainingsdaten zu finden. Dies macht einen Vergleich unmöglich und somit wäre eine Klassifikation auf keinen Fall denkbar.

## 3.2 Trainings- und Testdaten

Um eine der in Abschnitt 3.1 erklärten Sentimentanalyse-Applikationen zu erstellen und zu testen, werden Trainings- und Testdaten benötigt, welche, wie bereits im zuvor genannten Kapitel erwähnt, fürs Trainieren und Testen je drei unterschiedliche Datensammlungen sind und aus zwei Sets generiert werden. Auch die Trainings- und Testdatensets werden auf gleiche Art und Weise erstellt. Somit ergeben sich folgende sechs Datensammlungen, die in den Sentimentanalyse-Applikationen verwendet werden:

- 3000 Tweets fürs Training,
- 1500 Movie-Reviews fürs Training,

- 3000 Tweets + 500 Movie-Reviews fürs Training,
- 1000 Tweets fürs Testen,
- 500 Movie-Reviews fürs Testen und
- 1000 Tweets + 500 Movie-Reviews fürs Testen.

Alle Datensets innerhalb einer Applikation müssen gleich behandelt werden, das heißt die Trainingsdaten, die für die Generierung des Klassifikations-Modelles verwendet werden und die Daten zum Testen müssen dieselbe Strukturierung, Aufbereitung und Verarbeitung aufweisen. Auf die genaue Strukturierung und die Textverarbeitung wird in den Abschnitten 4.3 und 4.4 genau eingegangen.

### 3.2.1 Twitter im Allgemeinen

Twitter[1] als online Plattform ist ein kostenloser Service, wo registrierte Mitglieder, so genannte Benutzer (User) oder „Tweeter“, in der Lage sind kurze Nachrichten, so genannte „Tweets“ zu publizieren und Tweets von anderen registrierten Benutzern zu folgen. Dabei können verschiedene Geräte (Mobiltelefone, Tablets, Computer,...) und Plattformen (Website oder Apps) verwendet werden. Kurz zusammengefasst, Twitter ist ein einfaches soziales Netzwerk [14].

#### Tweets und Retweets

Wie bereits erwähnt, wird eine Nachricht, die auf Twitter von einem registrierten Benutzer publiziert wird, „Tweet“ genannt. Jeder Tweet ist in der Länge auf 140 Zeichen limitiert. Dieses kurze und kompakte Format ist die spezielle Charakteristik von Twitter. Eine Nachricht ermöglicht so informelle Zusammenarbeit (mittels Hashtags) und schnelle Informationsverbreitung, die E-Mails in der Neuigkeitenverbreitung in den Schatten stellt. Ein Retweet hingegen ist simpel eine Nachricht, die von einem Benutzer bereits publiziert und von einem zweiten ein weiteres Mal veröffentlicht wurde [14].

#### Hashtags

Hashtags „#“ werden bei Tweets als Meta-Information in Nachrichten verwendet, um diese einem Thema nach zu gruppieren. Somit kann einfach nach dem Begriff, der nach dem Hashtag kommt, gesucht werden und es werden alle Tweets, die ihn enthalten, zurückgeliefert. Mehr als 90% aller Tweets enthalten Hashtags [14].

[1] https://twitter.com/

**Public Stream**

Im Public Stream von Twitter sind Nachrichten von Benutzern enthalten, die öffentlich zur Verfügung gestellt wurden. Diese können von registrierten Usern ausgelesen und verwendet werden [14]. Es wurden einige Twitter Datensets mit Daten des Public Streams zusammengestellt und zum Download angeboten. Allerdings ist dies nicht mehr erlaubt, da die Daten von Twitter nur mehr für den persönlichen Gebrauch verwendet werden dürfen. Es gab einige Änderungen in den „Developer Agreement & Policy" von Twitter. Unter Punkt „(I) Twitter API and Twitter Content" Unterkapitel „(B) License from Twitter" steht geschrieben, dass die Inhalte des Public Streams nur mehr in den eigenen Anwendungen verwendet und nicht mit anderen geteilt werden dürfen, siehe https://dev.twitter.com/overview/terms/agreement-and-policy.

### 3.2.2 Das Twitter-Datenset

Da, wie schon erwähnt, kein existierendes Datenset von Twitter verwendet werden darf, wurde nach einer anderen Möglichkeit gesucht, eine große Datensammlung mit positiv und negativ eingestuften Nachrichten auf schnellstem Wege automatisch zu generieren.

In diesem Falle wurde, wie in „Twitter as a Corpus for Sentiment Analysis and Opinion Mining" [26] beschrieben, ein solches Set automatisch erstellt, indem Tweets von dem Public Stream geladen werden, die Smileys enthalten [26]. Für die Datensammlung mit positiven Tweets wurde nach Nachrichten gesucht, die glückliche Smileys „:D", „:)" und „:-)" enthalten. Die negativen Nachrichten wurden durch negative Smileys „:(„ und „:'(„ charakterisiert.

### 3.2.3 Das Movie-Review-Datenset

Movie-Reviews im Allgemeinen haben eine andere Charakteristik als Tweets, denn sie enthalten bei weitem mehr Zeichen als eine Nachricht auf Twitter. Außerdem sind sie normalerweise in der Hochsprache formuliert, wo hingegen Tweets auch Jargon und Abkürzungen innehaben. Jede Kritik ist logischerweise offensichtlich positiv oder negativ auf den Film bezogen formuliert. Das Sentiment ist von Menschenhand einfach zu erkennen [5].

Um ein Set mit vielen Movie-Reviews zu bekommen, kann einfach eine bereits vorhandene Datensammlung mit Kritiken verwendet werden. So bietet sich beispielsweise das Movie-Review-Datenset des Natural Language Processing Toolkits (NLTK)[2] an, in dem die Reviews als positiv oder negativ gekennzeichnet sind.

[2] http://www.nltk.org/

**Tabelle 3.7:** Tabelle mit den am weitesten verbreiteten Datenbanksystemen mit deren Datenmodell und Abfragesprache (Query Language) [31]. Column = spaltenorientiert, Inhalte Spaltenweise gespeichert, Document = dokumentenorientiert, Inhalte Dokumentenweise gespeichert, Graph = graphenorientiert, Graphen speichern vernetzte Informationen, Key-Value = Schlüssel-Wert, Inhalte als Wert einem Schlüssel zugeordnet, Relational = tabellenorientiert, Inhalte in Tabellen gespeichert.

| Datenbanksystem | Datenmodell | Query Language |
|---|---|---|
| Cassandra | Column | Range, MapReduce |
| CouchDB | Document | MapReduce |
| HBase | Column | Range, MapReduce |
| MongoDB | Document | MapReduce |
| MySQL | Relational | SQL |
| Neo4j | Graph | SPARQL, internal |
| Redis | Key-Value | |
| Riak | Key-Value | MapReduce |

## 3.3 Datenspeicherung von „Big Data“

Big Data selbst repräsentiert große Datenmengen, die heutzutage schnell durch die Verwendung des Internets anfallen. Vor allem bei Daten aus sozialen Netzwerken ist die Wachstumskurve enorm. Diese riesigen Mengen abzuspeichern, erfordert Systeme, die schnell und effizient arbeiten. Jedes Datenbanksystem zur Speicherung von Inhalten hat sein eigenes Anwendungsgebiet, für welches es optimiert wurde. So können Systeme beispielsweise auf unzählige Speicherarbeiten („writes“) oder Lesearbeiten („reads“) spezialisiert sein. Einige weisen eine strukturierte Darstellung auf, andere sind unstrukturiert, dafür steht aber eine gute Performance im Vordergrund. In Tabelle 3.7 sind die am meisten verwendeten Datenbanksysteme mit deren Datenmodell und Abfragesprache („Query Language“) aufgelistet [31].

Diese in Tabelle 3.7 aufgelisteten acht Datenbanksysteme unterscheiden sich in der Datenspeicherung, deren Abfragesprachen und Datenverarbeitung. In Grafik 3.1 sind die Datenmodelle dieser Systeme nach deren Strukturiertheit und Datenskalierbarkeit geordnet dargestellt. Jeder Typ hat unterschiedliche Vor- und auch Nachteile, die sich je nach Nutzen anders darstellen. Bei der Speicherung von Nachrichten aus sozialen Netzwerken, insbesondere Tweets, ist es wichtig, dass diese zuverlässig und hochskaliert gespeichert werden. Dafür eignet sich laut Tabelle ein „Key-Value“ System, wobei sich Riak gut anbietet [3]. Mehr Informationen über Riak können aus Abschnitt 4.1 oder unter http://basho.com/riak/ entnommen werden.

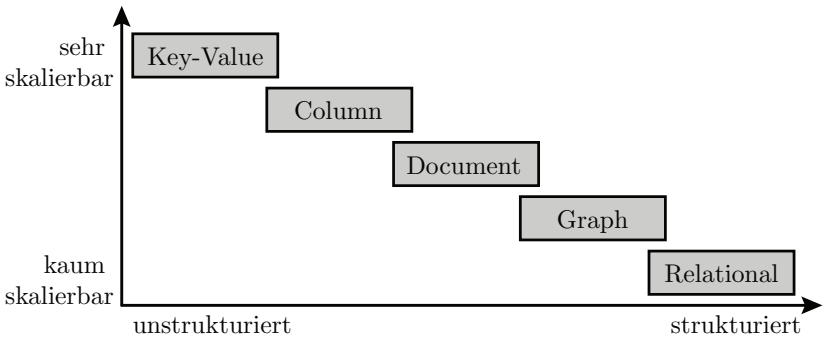



**Abbildung 3.1:** Diagramm mit allen Datenmodellen aus Tabelle 3.7 mit deren Strukturiertheit (x-Achse) und Datenskalierbarkeit (y-Achse) [31].

## 3.4 Textaufbereitung

Um die Inhalte der Trainings- und Testdaten vergleichen zu können, müssen sie auf gleiche Art und Weise bearbeitet und aufbereitet werden, da sonst die Daten nicht verglichen und somit keine Sentiment-Annahme (positiv oder negativ) getätigt werden kann. Um bessere Klassifikations-Ergebnisse zu erhalten, werden die Tweets und Movie-Reviews auf Sonder- und Satzzeichen untersucht und diese entfernt, da sie über die Positivität oder Negativität meist keine Aussage tätigen. Des Weiteren nehmen diese Symbole nur ungefähr 12% innerhalb eines Textes ein [1]. Nach diesem Textverarbeitungsprozess wird, je nach Applikation, Stemming angewendet oder nicht, es wird hier der Porter-Stemmer verwendet. Nun werden die Texte in Uni- und beziehungsweise oder Bigramme (je nach Applikation) aufgesplittert. Somit ergeben sich folgende Schritte:

1. Entfernung der Satz- und Sonderzeichen,
2. je nach Applikation Stemming mit dem Porter-Stemming-Verfahren und
3. Featuregenerierung.

Dieser Ablauf setzt voraus, dass die Inhalte alle „sauber“ aufgebaut und geschrieben sind, also nur aus Wörtern oder Zahlen bestehen. Die Movie-Reviews, die verwendet werden, sind bereits vorbearbeitet und sauber vorhanden, da sie von Bo Pang und Lillian Lee gesammelt, aufbereitet und klassifiziert wurden. Allerdings sind die Tweets nicht ganz so „sauber“, weil, vor allem in der Onlinewelt, die Benutzer gerne Abkürzungen und Jargon verwenden. Somit müssen die Nachrichten von Twitter noch aufbereitet werden.

### 3.4.1 Aufbereitung der Twitter Daten vor der Speicherung

Bei Tweets ist es üblich Meta-Informationen wie Retweets (erneut getweetete Tweets), Hashtags (#), Hyperlinks, Referenzen an User (@), E-Mail-Adressen und zusätzliche Leerzeichen oder Zeilenumbrüche zu verwenden. Diese Daten sind für eine Sentimentanalyse in deren eigentlicher Form nicht direkt oder gar nicht zu gebrauchen. Deshalb ist es notwendig, sie vor Speicherung der Tweets in die Datenbank zu lokalisieren und abzuändern oder zu entfernen.

Ein Retweet ist ein Tweet von einem User der von einem anderen Benutzer ein weiteres Mal der Öffentlichkeit zugänglich gemacht wurde wurde. Der Inhalt dieser Nachricht bleibt gleich und wäre somit möglicherweise ein weiteres Mal in dem Datenset zum Trainieren und Testen der Sentimentanalyse-Applikation enthalten. Doppelt vorhandene Inhalte würden die Ergebnisse einer Klassifikation nicht verbessern, deshalb werden solche Tweets nicht verwendet.

Für einen Hashtag gibt es zwei unterschiedliche Anwendungsgebiete. Einerseits wird er innerhalb der Nachricht als textuelles Bestandteil verwendet und der Hashtag an sich dient nur zur Unterstützung der Gruppierung. Andererseits wird er am Ende des Tweets verwendet, um den Inhalt der Nachricht verschiedenen Kategorien zuzuordnen. Die Tags sind somit auch wichtig für das Sentiment der Nachricht und können nicht einfach herausgelöscht werden. Deshalb wird beispielsweise aus dem Hashtag „#sentiment“ lediglich das Wort „sentiment“ geformt, also simpel einfach nur der Hash (#) entfernt.

Um den Tweet noch „sauberer“ zu bekommen, werden Hyperlinks herausgefiltert, da diese nicht für die Einstufung in positiv oder negativ hilfreich sind. Es wird somit nach allen Links innerhalb des Tweets gesucht und diese werden einfach rausgelöscht.

Ebenso ist eine Referenz an einen anderen User mit „@“ nicht für eine Sentimentanalyse relevant, somit werden auch diese Referenzen in den Tweets aufgespürt und komplett entfernt, da auch der Name des Benutzers nicht aussagekräftig ist.

E-Mail-Adressen sind wenig, eigentlich gar nicht hilfreich, um eine Nachricht einzustufen und werden deshalb, wie die Benutzerreferenzen, lokalisiert und anschließend gelöscht.

Zusätzliche Leerzeichen und Zeilenumbrüche behindern nur das schnelle Arbeiten mit den Nachrichten und sind für die inhaltliche Bedeutung des Tweets irrelevant, somit werden diese aufgespürt und aus den Tweets entfernt.

## 3.5 Der Klassifikations-Ablauf

Für eine Sentimentanalyse müssen einige Rechen- und Verarbeitungsschritte durchgeführt werden. Der genaue, interne Ablauf weißt folgende Schritte auf:

1. Auslesen der Trainings- und Testdaten
   (a) Erstellen einer Liste mit allen positiven Trainings- und Testdaten
   (b) Erstellen einer Liste mit allen negativen Trainings- und Testdaten
2. Aufbereitung und Strukturierung der ausgelesenen Daten auf gleiche Art und Weise
   (a) Verarbeiten der positiven Trainings- und Testdaten
      i. Stemming (je nach Applikation durchgeführt oder nicht)
      ii. Stoppwortentfernung (je nach Applikation durchgeführt oder nicht)
   (b) Verarbeiten der negativen Trainings- und Testdaten
      i. Stemming (je nach Applikation durchgeführt oder nicht)
      ii. Stoppwortentfernung (je nach Applikation durchgeführt oder nicht)
3. Erstellung von zwei Listen mit
   (a) allen positiven Wörtern
   (b) allen negativen Wörtern
4. Erstellung des Feature-Vektors aus den Trainings- und Testdaten
   (a) Trainings- und Testdaten zu Bigrammen formen (je nach Applikation durchgeführt oder nicht)
   (b) Trainings- und Testdaten zu Unigrammen formen (je nach Applikation durchgeführt oder nicht)
   (c) Limitieren der Bigramme und beziehungsweise oder Unigramme auf diejenigen mit den meisten Informationen (je nach Applikation durchgeführt oder nicht)
5. Erstellung des Klassifikations-Modells aus dem Trainings-Daten-Feature-Vektor
6. Klassifikation der Testdaten durch Übergabe des Test-Daten-Feature-Vektors an das Modell unter Anwendung des Naïve Bayes Verfahrens
7. Berechnung der Kennzahlen zur Performancemessung
   (a) Accuracy
   (b) Positive Precision
   (c) Negative Precision
   (d) Positive Recall
   (e) Negative Recall

Der eben beschriebene Ablauf ist in Abbildung 3.2 grafisch dargestellt. Es ist zu erkennen, dass die Daten gleich behandelt werden. Nach dem Stemming, werden sie in Trainings- und Testdaten-Sets geteilt und weiterverarbeitet.

## 3.6 Ähnliche Ansätze und Arbeiten

Es gibt wirklich viele andere Ansätze zur Sentimentanalyse mit Daten von Twitter. Ein sehr ausschlaggebender Ansatz für die in diesem Kapitel beschriebene Sentimentanalyse, wird noch genauer vorgestellt. Des Weiteren werden noch zwei Projekte vorgestellt, welche Inspiration und Ideen für die eigene Sentimentanalyse-Applikation lieferten.

### 3.6.1 Pak und Paroubeks Ansatz zur Sentimentanalyse mit Daten von Twitter

Twitter als beliebteste Microblogging-Plattform soll für eine Sentimentanalyse herangezogen werden. Pak und Paroubek zeigen, wie ein Korpus aus Twitter Daten für die Sentimentanalyse und auch das Opinion Mining automatisch gesammelt werden kann. Eine linguistische Analyse der Tweets wird durchgeführt und deren Charakteristika aufgezeigt. Durch die Verwendung des Korpus wird eine Sentimentanalyse-Applikation entwickelt, die in der Lage ist, positive, negative und neutrale Nachrichten zu identifizieren. Die Ergebnisse zeigen vor allem, dass die verwendeten Daten und Techniken zur Datenverarbeitung und Sentimentanalyse effizienter und besser als zuvor verwendete Methoden arbeiten. Es wird ausschließlich mit englischen Daten gearbeitet [26].

Das automatische Sammeln der Twitter Daten als Trainings- und Test-Sets wurde ebenso in der in dieser Arbeit entwickelte Sentimentanalyse-Applikation herangezogen. Pak und Paroubek verwendeten die Twitter API, um Tweets aus dem Public Stream zu laden. Es wurden drei verschiedene Klassen – positiv, negativ und objektiv – generiert. Um diese drei Klassen mit Daten zu füllen wurde nach:

- Tweets mit glücklichen Smileys „:-)“, „:)“, „=)“, „:D“ für positive,
- Tweets mit traurigen Smileys „:-(“, „:(“, „=(“, „;(“ für negative und
- Tweets von bekannten Zeitungen wie „New York Times“, „Washington Post“ für objektive Nachrichten gesucht [26].

Des Weiteren lieferte die Vorverarbeitung der Twitter Daten hilfreiche Ansätze, um die Tweets für eine Sentimentanalyse aufzubereiten. Die Applikation von Pak und Paroubek entfernt ebenfalls Hyperlinks, Benutzernamen und Retweets. Auch Satz- und Sonderzeichen werden außer Acht gelassen und nicht im Trainings- und Testdaten-Set verwendet [26].

Für eine genaue Beschreibung dieses Ansatzes und die Testergebnisse der Klassifikation, siehe „Twitter as a Corpus for Sentiment Analysis and

Opinion Mining“ [26]. Die beiden beschreiben dort ihren Ansatz und die Forschung auf diesem Gebiet sehr genau.

### 3.6.2 Microsoft Research Ansatz zur Sprachidentifikation

Die Sprachidentifikation an sich wird normalerweise von der Sentimentanalyse stark differenziert, aber als Klassifikations-Problem sind sich beide ähnlich. Bei der Sprachidentifikation wird lediglich die Nachricht einer Sprache zugewiesen, somit müssen verschiedene Klassen (Sprachen) gegeben sein, um die der Nachricht am ähnlichsten zuzuweisen. Bei der Sentimentanalyse ist dies im Prinzip genauso, nur sind die Klassen unterschiedlich, nämlich keine Sprache, sondern positiv oder negativ.

Die bisherige Forschung im Bereich der Sprachidentifikation hat sich auf „saubere“ Daten spezialisiert und nur eine bestimmte, kleine Anzahl an Sprachen zur Identifikation herangezogen. Zusätzlich wurden nur Dokumente mit sehr viel Inhalt verwendet, da diese eine Klassifikation vereinfachen. Diese Eigenschaften waren jedoch nicht bei Daten von Twitter gegeben und so konnten Nachrichten, die Abkürzungen und nicht in Hochsprache verfasst wurden, nicht klassifiziert werden [12].

Microsoft Research lieferte eine Applikation, welche die Sprache für Daten von Twitter erkennt. Dazu wurde darauf geachtet, dass einerseits sehr viele Sprachen (52) erkannt werden können, doch später wurden diese auf lediglich sechs limitiert, da dies die Performance der Klassifikation verbessert. Des Weiteren wurden verschiedene Algorithmen ausgetestet, um zu sehen, welcher die besten Ergebnisse liefert. Diese Algorithmen sind in deren Paper „Boot-Strapping Language Identifiers for Short Colloquial Postings“ genau erklärt [12].

Für die Identifikation werden zuerst öffentlich zugängliche Daten verwendet. Hier werden Artikel von Wikipedia herangezogen, da diese sehr „sauber“ geschrieben und in vielen Sprachen vorhanden sind. Danach wird das Wikipedia-Datenset mit Tweets kombiniert, um die Performance der Klassifikation zu verbessern. Um ein umfangreiches Twitter-Datenset zu erhalten, wurden Tweets ausgewählt, deren Klassifikationsergebnis des Wikipedia-Modells mit der Sprache der Meta-Daten des Tweets übereinstimmen. Die Idee der Kombination zwei verschiedener Datensammlungen wurde in dieser Arbeit und dem Projekt ebenfalls aufgegriffen, aber anstelle von Wikipedia-Daten wurden Movie-Reviews herangezogen, da diese „sauber“ geschrieben sind, mehr Inhalt aufweisen und einige Sammlungen öffentlich zugänglich sind.

### 3.6.3 Kaggle-Wettbewerb

Ein Wettbewerb auf Kaggle[3] mit dem Titel „Sentiment Analysis on Movie Reviews“ bei dem 861 Teams versuchten eine Sentimentanalyse-Applikation für Movie-Reviews zu erstellen, lieferte den Ansatz für die Verwendung von Reviews in Kombination mit Twitter Daten um Tweets in „positiv“ und „negativ“ einzuteilen.

Es wurde ebenso ein bereits vorhandenes Datenset – „Rotten Tomatoes movie review dataset“ – verwendet, welches von Bo Pang und Lillian Lee gesammelt und manuell klassifiziert wurde. Diese Sammlung weist allerdings fünf verschiedene Klassen auf, „negativ“, „eher negativ“, „neutral“, „eher positiv“ und „positiv“.[4]

[3] https://www.kaggle.com/

[4] Mehr Informationen zu dem Wettbewerb vom 28. Februar 2014 bis 28. Februar 2015, sind unter https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews zu finden.

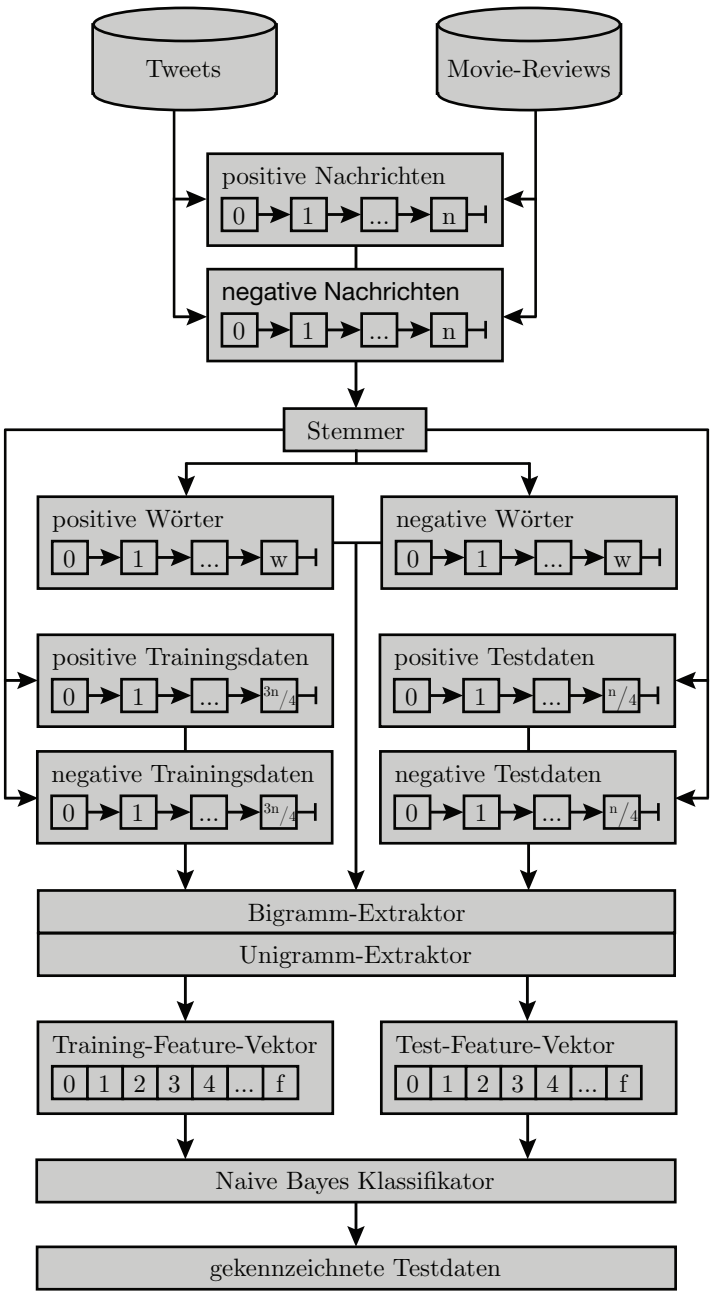



**Abbildung 3.2:** Ablauf einer Sentimentanalyse mit Stemming unter Verwendung der Uni- und Bigramme mit den meisten Informationen.

# Kapitel 4

# Technische Umsetzung des eigenen Ansatzes für die Verbesserung der Sentimentanalyse

Bis jetzt wurde die Sentimentanalyse und der Ansatz für die Verbesserung dieser nur in der Theorie besprochen, nun wird die Umsetzung im Vordergrund stehen und diese genau erklärt. Dies ist hilfreich, wenn der theoretische Hintergrund aus Kapitel 2 und der eigene Ansatz aus Kapitel 3 nicht ganz verstanden wurden.

## 4.1 Allgemeines zur Implementierung

Die komplette Sentimentanalyse-Applikation und somit alle Klassifikations-Modelle, die Datenaufbereitung und -verarbeitung sind mit dem Linux[1] Betriebssystem Ubuntu[2] entwickelt und getestet worden. Deshalb sind sie für die Verwendung unter Ubuntu optimiert.

Es wurde durchgehend darauf geachtet in einer einzigen Sprache zu entwickeln. Dafür wurde „Python 2.7[3]“ ausgewählt, da alle verwendeten Bibliotheken, die später noch in Abschnitt 4.2 genauer erklärt werden, mit Pyhton 2.7 kompatibel sind.

[1] https://www.linux.com/
[2] http://www.ubuntu.com/
[3] http://python.org/download/releases/2.7/

## 4.2 Verwendete Bibliotheken

Die Sentimentanalyse-Applikationen verwenden drei verschiedene Bibliotheken für unterschiedliche Zwecke:

- *NLTK (Natural Language Processing Toolkit)*[4] für die Sentimentanalyse und die Evaluierung der Klassifikations-Modelle,
- *RIAK*[5] als Datenbanksystem für die Speicherung der Twitter Daten und
- *Tweepy*[6] für die Verbindung zum Public Stream von Twitter und zum „Downloaden“ der Tweets.

### 4.2.1 Natural Language Processing Toolkit

Die Sentimentanalyse ist ein Problem bei dem natürliche Sprache verarbeitet werden muss. Unter natürlicher Sprache versteht man, die von Menschen kommunizierte Sprache wie Englisch, Deutsch, Spanisch und viele andere. Im Gegensatz dazu sind nicht natürliche Sprachen oder auch künstliche Sprachen wie Programmiersprachen oder mathematische Notationen durch explizite Regeln definiert. Diese Regeln vereinfachen die Verarbeitung von Texten. Viele Technologien basieren auf NLP (Natural Language Processing). Ein klassischer Anwendungsfall wäre, eine Suchmaschine, die das Gesuchte verstehen und in unstrukturierten Texten wiedererkennen muss [5].

Für die Verarbeitung von natürlicher Sprache werden bereits Bibliotheken wie NLTK angeboten. NLTK ist eine der führenden Plattformen für die Erstellung von Python-Programmen, die mit menschlicher Sprache arbeiten. Es ist eine einfach zu verwendende Schnittstelle, die außerdem mehr als 50 Korpora und eine Reihe von Textverarbeitungsbibliotheken für Klassifikation, Tokenisierung, Stemming, Tagging, Parsen und semantische Operationen anbietet. Bei der Klassifikation werden zusätzlich noch verschiedene Methoden zur Performancemessung angeboten, welche sehr hilfreich bei der Evaluierung sind.

Für eine gut strukturierte Einführung in die Verwendung dieser Bibliothek kann das Buch „Natural Language Proceccing with Python – Analyzing Text with the Natural Language Toolkit“ herangezogen werden [5]. Eine aktuelle Version des Buches ist immer auf der NLTK-Website (http://www.nltk.org/) zu finden oder unter http://www.nltk.org/book/.

Andere Bibliotheken sind beispielsweise „Apache OpenNLP[7]“ und „Stanford NLP.[8]“. Apache OpenNLP ist ein lernbasiertes Toolkit zur Verarbei-

[4] http://www.nltk.org/
[5] http://basho.com/posts/business/riak-0-10-is-full-of-great-stuff/
[6] http://www.tweepy.org/
[7] https://opennlp.apache.org/
[8] http://nlp.stanford.edu/

tung natürlicher Sprache, geschrieben in Java[9] Die Stanford Natural Language Processing Group bietet Teile ihrer NLP-Software, wie statistische Sprachverarbeitungsmethoden, maschinelles Lernen und regelbasierte NLP-Tools, zum Download an[10].

### 4.2.2 RIAK

Das Datenbanksystem Riak ist ein Key-Value Speichersystem, bei dem die einzelnen Datensätze nicht strukturiert in der Datenbank abgelegt werden. Vorteil eines solchen unstrukturierten Systems ist, dass es sehr schnell arbeitet, auch wenn eine riesige Anzahl an Daten verarbeitet werden muss. Jeder Datensatz wird als Key-Value-Paar gespeichert, das heißt, es ist ein Set mit einem Key (Schlüssel) und dem Inhalt (Value), welches folgende Strukturierung aufweist:

```
{
    "key" : "001",
    "value" : "Inhalt des Beispieldatensatzes"
}
```

Der Key kann automatisch vom System vergeben werden, oder ein Eigener beim Anlegen des Datensatzes übergeben werden. Bei der Speicherung von Tweets ist ein eigens definierter Key nicht nötig, deshalb wurde die automatisierte Speicherung vorgezogen [40].

Das System selber ist in Erlang[11] geschrieben, welches eine Programmiersprache und Plattform für hochskalierbare und hochparallelisierbare Systeme ist. Für die Suche nach einem Parameter und nicht nach dem Key wird MapReduce[12] verwendet [40]. Aktuell wird die Version 2.0.4 von Riak in den Sentimentanalyse-Applikationen unterstützt.

### 4.2.3 Twitter Bibliotheken

Twitter bietet eine Vielzahl von Bibliotheken in den verschiedensten Programmier- und Skriptsprachen, welche die Twitter API v1.1 unterstützen. Diese Bibliotheken werden nicht zwingend von Twitter getestet, außer die „hbc“, welche in Java[13] implementiert und selbst von Twitter entwickelt wurde. Eine umfangreiche Liste der Bibliotheken ist auf der Twitter-Website unter https://dev.twitter.com/overview/api/twitter-libraries zu finden.

Eine, die in dieser Arbeit beziehungsweise Projekt verwendeten Bibliothek ist „Tweepy“. Tweepy wird oft in Kombination mit Sentimentanalyse-Applikationen verwendet, um Tweets vom Public Stream zu sammeln [3, 14]. Es bietet eine Anbindung an Twitter RESTful und an die komplette

[9] Mehr dazu unter https://opennlp.apache.org/.
[10] Für mehr Informationen siehe http://nlp.stanford.edu/.
[11] http://www.erlang.org/
[12] http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
[13] https://www.java.com/de/

Streaming API, welche Methoden für die Authentifikation offeriert. Ohne dieser können vom Public Stream keine Nachrichten gelesen und auch keine Nachrichten auf die Timeline des Benutzers geschrieben werden [14].

## 4.3 Trainings- und Test-Korpus

Wie bereits in Abschnitt 3.2 erwähnt, sind verschiedene Datensets vorhanden, die auf gleiche Art und Weise behandelt und verarbeitet werden müssen. Die genaue Verarbeitung wird in Abschnitt 4.4 noch genau besprochen. Jedoch ist es zunächst wichtig, die genaue Datenstruktur, die Speicherung und das Auslesen der Einträge aus dem Twitter- und dem Movie-Review-Datenset zu verstehen.

### 4.3.1 Struktur der Daten

Bei der Datenstruktur muss zwischen drei verschiedenen Ausführungen unterschieden werden:

- die Struktur der Twitter Daten in der RIAK Datenbank,
- die Struktur der Movie-Review Daten im NLTK und
- die Struktur aller Datensets in der Sentimentanalyse-Applikation.

Da die Daten in der Datenbank und im Natural Language Processing Toolkit unterschiedlich abgelegt sind, müssen diese nach dem Auslesen aus den jeweiligen Speicherorten in eine einheitliche Form gebracht werden. Danach können sie erst für die Sentimentanalyse aufbereitet und verwendet werden.

#### Twitter-Daten in der Datenbank

In der Datenbank an sich hat jeder Eintrag eine bestimmte Struktur, welche in Abschnitt 4.2.2 schon erklärt wurde. Somit wird zu einem Key als Inhalt ein Tweet abgespeichert, der wiederum eine eigene Struktur aufweist. Für jeden Tweet ist ein Sentiment – positiv oder negativ – vorhanden, welches zusätzlich zu dem Inhalt des Tweets vermerkt wird. Somit ergibt sich folgende Struktur des Tweets in der RIAK Datenbank:

```
{
    "key" : "001",
    "value" : {
        "sentiment" : "positive",
        "text" : "Inhalt des Tweets"
    }
}
```

Um überhaupt eine solche Struktur zu erreichen, müssen zunächst die gestreamten Tweets aus dem Public Stream verarbeitet werden. Ein Public-Stream-Tweet beinhaltet nicht nur den Text, es gibt noch weitere unzählige Informationen zu der Nachricht selbst und zu dem User, der ihn verfasst

und publiziert hat. Somit werden Informationen wie „Erstellungsdatum“, Koordinaten des Ursprungsortes“, „als Favorit gekennzeichnet“, „Sprache“, „ID“, „Text (Inhalt der Nachricht)“, „Anzahl als Retweet publiziert“, „Zeitstempel“, „Source-Adresse“, „Benutzerinformationen“ und „Ort“ mit einem Tweet mitgeliefert.

In Programm 4.1 wurde ein Tweet aus dem Public Stream als Beispiel im JSON[14]-Format dargestellt. Hier wurden jedoch die Benutzerinformationen weggelassen, da diese zu umfangreich und für die Verarbeitung des Tweets und Verwendung bei einer Sentimentanalyse irrelevant sind. Für die Klassifikations-Applikationen sind lediglich die Sprache und der Inhalt der Nachricht von Bedeutung. Das Sentiment wird, wie schon in Abschnitt 3.2 erwähnt, durch Smileys bestimmt. Nicht jeder Tweet beinhaltet alle aufgezählten Informationen. Sind einzelne Parameter nicht bekannt, werden diese durch „None“ gekennzeichnet.

**Programm 4.1:** Struktur eines Tweets aus dem Public Stream im JSON-Format.

```
{
  'created_at': 'Wed Jul 01 09:56:39 +0000 2015',
  'coordinates': None,
  'favorited': False,
  'contributors': None,
  'in_reply_to_screen_name': None,
  'in_reply_to_user_id_str': None,
  'in_reply_to_status_id': None,
  'lang': 'de',
  'id_str': '616183628272414720',
  'in_reply_to_status_id_str': None,
  'text': 'Ein Beispiel für einen Tweet aus dem Public Stream',
  'retweet_count': 0,
  'timestamp_ms': '1435744599775',
  'possibly_sensitive': False,
  'source': '<a href="http://ifttt.com" rel="nofollow">IFTTT</a>',
  'favorite_count': 0,
  'user': { ... },
  'truncated': False,
  'filter_level': 'low',
  'id': 616183628272414720,
  'place': None,
  'in_reply_to_user_id': None,
  'geo': None,
  'entities': { ... },
  'retweeted': False
}
```

[14] JSON = JavaScript Object Notation, einfach zu lesendes Datenformat, mehr dazu ist unter http://json.org/ zu finden.

**Programm 4.2:** Movie-Review-Datensammlung als Beispiel im NLTK.

```
nltk_data
└── corpora
    └── movie_reviews
        ├── neg
        │   ├── CV000_[ID].txt
        │   ├── CV001_[ID].txt
        │   ├── CV002_[ID].txt
        │   ├── ...
        │   └── CV999_[ID].txt
        ├── pos
        │   ├── CV000_[ID].txt
        │   ├── CV001_[ID].txt
        │   ├── CV002_[ID].txt
        │   ├── ...
        │   └── CV999_[ID].txt
        └── README
```

### Movie-Review Daten im NLTK

Die Movie-Reviews sind im NLTK bereits vorhanden und können nach der Installation einfach verwendet werden. Die Daten sind in Dateien abgelegt. Die Datei-Bezeichnung ergibt sich aus einem „corss-validation-tag“ und dem ID der HTML-Datei von http://www.cs.cornell.edu/People/pabo/movie-review-data/. Mit der ID kann auf die Datei zugegriffen und alle Datensätze angesprochen und ausgelesen werden. Jeder Datensatz weist ein Sentiment und den Inhalt auf. Die Struktur der Dateien im Natural Language Processing Toolkit ist in Programm 4.2 dargestellt.

### Struktur in der Applikation

Da, wie bereits in den vorigen Abschnitten erklärt, die Daten von Twitter und die Movie-Reviews des Natural Language Processing Toolkits eine unterschiedliche Strukturierung aufweisen, müssen diese nach dem Auslesen in der Applikation aneinander angepasst werden. Zunächst wurden alle positiven und alle negativen Tweets beziehungsweise Movie-Reviews separat ausgelesen und in einer Liste abgelegt, es entstehen somit folgende Datensammlungen:

- posTweets,
- negTweets,
- posReviews und
- negReviews.

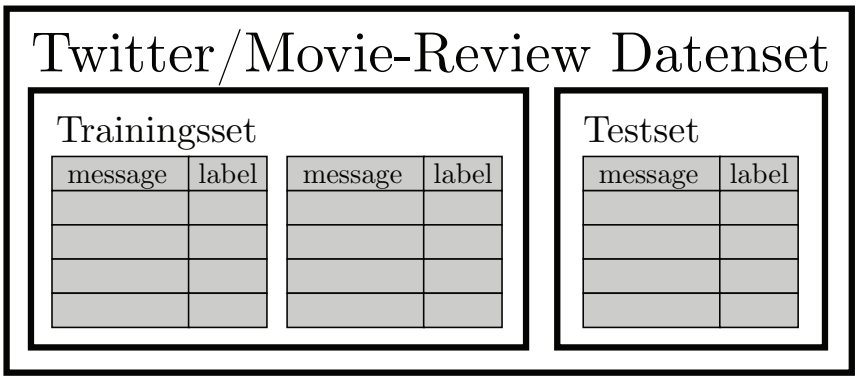



**Abbildung 4.1:** Struktur der Twitter- und Movie-Review-Datensets innerhalb der Applikation.

Die ausgelesen Daten von der Riak Datenbank (Twitter) und NLTK (Movie-Reviews) werden wiederum verarbeitet und daraus vier Sammlungen mit Features erstellt:

- posTweetFeats,
- negTweetFeats,
- posReviewFeats und
- negReviewFeats.

Nach dem Erstellen der Features können Sets mit den Trainings- und Testdaten erstellt werden. Die Trainingsdaten umfassen 3/4 der Twitter oder Movie-Review Daten und die Testdaten somit nur 1/4. Es entstehen aus den bereits erwähnten vier Datensammlungen folgende neue Sets mit positiven und negativen Daten:

- trainTweetFeats,
- testTweetFeats,
- trainReviewFeats und
- testReviewFeats.

Die Strukturierung ist für die Tweets und Movie-Reviews gleich, besteht aus Trainings- und Testdaten-Sets und ist in Abbildung 4.1 dargestellt.

Jede dieser Datensammlungen ist allgemein gleich aufgebaut, um die spätere Aufbereitung und Verwendung für die Klassifikation zu erleichtern. Der Aufbau bildet eine Liste mit Nachrichten, die wie folgt aussieht:

```
1 {
2     {'Das', 'ist', 'die', 'erste', 'Nachricht'},
3     {'Die', 'zweite', 'widerum', 'befindet', 'sich', 'hier'}
```

```
4 }
```

wobei jeder Eintrag eine Liste mit den Wörtern der Nachricht darstellt.

### 4.3.2 Datenspeicherung

Bei der Speicherung muss ebenso zwischen Tweets und Movie-Reviews unterschieden werden, da das Datenset mit den Reviews schon vorhanden ist, muss keine Speicherung in eine Datenbank mehr vorgenommen werden. Die Tweets hingegen müssen, wie schon erwähnt, erst vom Public Stream geladen, vorbearbeitet und danach in einer Datenbank abgelegt werden.

Der Weg der Tweets vom Public Stream bis in die Datenbank erfordert einige Arbeitsschritte, die durchgeführt werden müssen:

1. Verbindungsaufbau zu Twitter und Authentifikation
2. Stream-Listener erstellen, der solange die Daten vom Public Stream ausliest, bis die gewünschte Menge an Tweets vorhanden ist
3. Streaming Starten
   (a) Verbindung zur RIAK Datenbank herstellen
   (b) Geladenen Tweet vorbearbeiten
   (c) Tweet speichern

Die Verbindung zu Twitter kann ganz einfach mit Tweepy[15] hergestellt werden. Die Authentifikation ist auch mit dieser Bibliothek durchzuführen. Allerdings werden vier verschiedene Parameter benötigt, die auf https://dev.twitter.com/apps unter „OAuth Settings“ und „Your Access Token“ zu finden sind. Es werden der „Consumer Key“, „Consumer Secret“, „Access Token“ und „Access Token Secret“ benötigt. Ohne diese Schlüssel beziehungsweise Codes kann keine Verbindung zu Twitter und somit auch nicht zum Public Stream hergestellt werden.

Nach dem Verbindungsaufbau kann der Stream-Listener erstellt werden. Dieser liest solange die Daten vom Public Stream aus, bis eine bestimmte Anzahl an positiven und negativen Tweets vorhanden ist. Die Anzahl selbst kann dem Stream-Listener bei der Generierung übergeben werden. Es muss darauf geachtet werden, dass genauso viele positive wie negative Nachrichten abgespeichert werden, da sonst die Sentimentanalyse verfälschte Ergebnisse liefert. Um die positiven und negativen Tweets zu bestimmen, wird nach Smileys in der Nachrichten gesucht. Diese können während des Streamings herausgefiltert werden. Außerdem ist auch noch die Sprache wichtig, da die Tweets alle in englischer Sprache sein sollten. Der Filtervorgang kann einfach mit Tweepy durchgeführt werden. Das Starten des Streamings und die Filterung können aus Programm 4.3 herausgelesen werden.

Nachdem der Stream-Listener erstellt und mit dem Streaming begonnen wurde, kann nun die Vorbearbeitung und Speicherung vorgenommen

[15] Mehr Informationen sind in Abschnitt 4.2.3 zu finden.

**Programm 4.3:** Starten von Streamings und Filterung der Tweets für das positive Datenset in Python. „listener“ und „authentication“ sind zwei selbst erstellte Klassen, wobei beim „listener“ das Laden und die Verarbeitung der Tweets und in „authentication“ die Authentifikation durchgeführt wird.

```
1 # Generierung des Stream-Listeners für 2000 positive Tweets
2 streamL = listener.TwitterStreamListener(maxnum=2000, sentiment="pos")
3 # Starten des Streamings mit Übergabe von Authentifikation und Listener
4 stream = tweepy.streaming.Stream(authentication.auth, streamL)
5 # Filtern der Tweets aus dem Stream
6 stream.filter(track=[':D', ':)', ':-)'], languages=['en'])
```

werden. Zunächst wird der Tweet ins JSON-Format konvertiert und nochmals auf die Sprache mittels den Meta-Informationen überprüft. Ist dieser nicht in Englisch, dann wird er nicht verarbeitet sondern gleich der nächste Tweet herangezogen. Ist die Sprache der Nachricht Englisch, wird der Text des Tweets ausgelesen und dieser in Kleinbuchstaben umgeformt. Jetzt wird überprüft, ob der Tweet ein Retweet ist. Da ein solcher nicht benötigt wird, wird im Falle der Retweetkennzeichnung die Nachricht verworfen. In allen anderen Fällen wird mit der eigentlichen Vorverarbeitung begonnen. Alle Hyperlinks, Benutzernamen, E-Mail Adressen, zusätzliche Leerzeichen und Leerzeilen werden mittels Stringvergleiche und Regular Expressions[16] aufgespürt und gelöscht. Nun wird der Tweet in einzelne Wörter aufgesplittert und in einer Liste abgelegt. Jetzt muss überprüft werden, ob die Nachricht genau oder mehr als zwei unterschiedliche Wörter beinhaltet, da sonst die Erstellung von Bigrammen unmöglich ist. Entspricht der Tweet diesen Anforderungen, werden Satz- und Sonderzeichen entfernt und die Nachricht in der Datenbank abgelegt. Der Verbindungsaufbau zu RIAK und die Speicherung des Tweets kann in Programm 4.4 eingesehen werden.

### 4.3.3 Auslesen der Daten

Bei der Speicherung im vorigen Kapitel wurde erwähnt, dass zwischen den Tweets und Movie-Reviews unterschieden werden muss. Ebenso muss beim Auslesen dieser Daten eine Differenzierung vorgenommen werden, da es einen Unterschied macht, die Daten aus der RIAK Datenbank oder aus NLTK auszulesen.

#### Twitter Daten

Um die Tweets aus dem Speicherort auszulesen, muss zu Beginn die Verbindung zur Datenbank hergestellt werden. Der Verbindungsaufbau ist mit

[16] Textstring, der einen Suchausdruck beschreibt, mehr dazu ist unter http://www.regular-expressions.info/ zu finden.

**Programm 4.4:** Verbindungsaufbau zu RIAK und Speicherung eines Tweets in Python. „self.sentiment“ entspricht dem Sentiment, welches dem Listener übergeben wurde und „final_tweet“ ist der verarbeitete Inhalt der Nachricht.

```
1 # Verbindungsaufbau zur Datenbank
2 client = riak.RiakClient(port=8087, protocol='pbc')
3 # Erstellung eines Speicherortes für die Tweeets
4 twitterBucket = client.bucket('tweets-sentimentanalyse')
5 # Generierung des Datensatzes, "None" = automatischer Key
6 msg = twitterBucket.new(None, data={
7     'sentiment': self.sentiment,
8     'text': final_tweet,
9 })
10 # Speicherung des zuvor erstellten Datensatzes
11 msg.store()
```

dem der Speicherung gleichzusetzen und kann aus Programm 4.5 Zeile 2 entnommen werden. Danach muss ein „Query“ für die eigentliche Abfrage erstellt werden. Bei RIAK muss der Speicherort der Daten zum Query hinzugefügt werden, siehe Zeile 4 in Programm 4.5. Jetzt kann die Abfrage generiert werden. Da die Keys von den Tweets nicht bekannt sind und auch nicht bekannt sein müssen, wird MapReduce verwendet, um alle Datensätze mit dem Sentimentvermerk „positive“ und beziehungsweise oder „negative“ auszulesen. Hierfür wird eine JavaScript Funktion erstellt, siehe Programm 4.5. Abschließend wird die Abfrage mit „run()“ gestartet und alle Tweets in einer Liste abgelegt.

#### Movie-Review Daten

Im Gegensatz zum Auslesen der Tweets muss bei den Movie-Reviews keine Verbindung zu einer Datenbank hergestellt werden. Das Natural Language Processing Toolkit bietet einfache Funktionen, um die Datensätze effizient auszulesen. Zunächst müssen lediglich die IDs der Dateien, welche die positiven Reviews und jene die negative Reviews enthalten, in die Applikation geladen werden. Danach können alle Dateien durchlaufen, die einzelnen Kritiken ausgelesen und in einer Liste für die positiven und eine für die negativen Reviews abgespeichert werden. Dieser Ablauf wird in Programm 4.6 durchgeführt.

## 4.4 Aufbereitung der Trainings- und Testdaten für die Klassifikation

Die Aufbereitung der Daten erfolgt nach dem Auslesen aus der Datenbank (Tweets) und vom NLTK (Movie-Reviews). Zunächst werden die Daten

**Programm 4.5:** Verbindungsaufbau zur RIAK Datenbank und Auslesen der positiven Tweets in Python. Für die negativen Tweets kann der Code einfach abgeändert werden, indem das sentiment von „positive“ auf „negative“, und „posTweets“ in „negTweets“ angepasst wird.

```
# Verbindungsaufbau zur Datenbank
client = riak.RiakClient(port=8087, protocol='pbc')
# Bucket in dem die Tweets gespeichert wurden zum Query hinzufügen
query = client.add('tweets-sentimentanalyse')

# JavaScript Map-Funktion für das Auslesen der positiven Daten.
query.map("function(v) {
    var data = JSON.parse(v.values[0].data);
    if(data.sentiment == 'positive') {
        return [[v.key, data]];
    }
    return [];
}")
# Abfrage durchführen und ablegen der Tweets in "posTweets"
posTweets = query.run()
```

**Programm 4.6:** Auslesen der Movie-Reviews aus dem NLTK in Python.

```
# Liste mit allen Datei-IDs der negativen Reviews erstellen
negids = movie_reviews.fileids('neg')
# Liste mit allen Datei-IDs der positiven Reviews erstellen
posids = movie_reviews.fileids('pos')

# Alle negativen Reviews auslesen und zur Liste hinzufügen
for fileid in negids:
    negReviews.append(list(movie_reviews.words(fileids=[fileid])))

# Alle positiven Reviews auslesen und zur Liste hinzufügen
for fileid in negids:
    posReviews.append(list(movie_reviews.words(fileids=[fileid])))
```

in gleiche Struktur gebracht (siehe Abschnitt 4.3.1) und danach beginnt die eigentliche Aufbereitung. Dafür wurden verschiedene Normalisierungs-Funktionen erstellt, die je nach Applikation angewendet und für folgende Zwecke verwendet werden:

- Tokenisierung,
- Großbuchstaben in Kleinbuchstaben umformen,
- Stemming mit dem Porter-Stemming-Verfahren (je nach Applikation angewandt oder nicht) und
- Stoppwort-Entfernung (je nach Applikation angewandt oder nicht).

**Programm 4.7:** Normalisierung der Daten ohne Stemming und Stoppwort-Entfernung in Python.

```
1 def normalize(s):
2     words = tokenize.wordpunct_tokenize(s.lower().strip())
3     return ' '.join([w for w in words])
```

**Programm 4.8:** Normalisieren der Daten mit Stemming (Porter-Stemmer) in Python.

```
1 def normalize(s):
2     words = tokenize.wordpunct_tokenize(s.lower().strip())
3     stemmer = stem.PorterStemmer()
4     return ' '.join([stemmer.stem(w) for w in words])
```

**Programm 4.9:** Normalisierung der Daten mit Stoppwort-Entfernung in Python.

```
1 def normalize(s):
2     words = tokenize.wordpunct_tokenize(s.lower().strip())
3     s_words = [w for w in words if w not in stoprwords.words('english')]
4     return ' '.join([w for w in s_words])
```

Die Normalisierungs-Funktion, welche in den meisten Sentimentanalyse-Applikationen verwendet wird, ist in Programm 4.7 vermerkt. Es werden die einzelnen Wörter tokenisiert und anschließend auf Großbuchstaben überprüft. Vorhandene Großbuchstaben werden in Kleinbuchstaben umgeformt. Anschließend werden die tokenisierten Wörter wieder zu einem String zusammengeführt.

Als nächstes wurde Stemming[17] zur Normalisierung hinzugefügt, welches die Wörter in deren Basisformen umformt. In Programm 4.8 ist ein Beispiel für die Normalisierung mit Stemming unter Anwendung des Porter-Stemming-Verfahrens zu finden.

Die Stoppwort-Entfernung gehört auch zur Textaufbereitung, so wurde diese, wie das Stemming, auch in eine eigene Normalisierungs-Funktion gebracht und zwar für alle Sentimentanalyse-Applikationen, die Stoppwörter nicht in die Klassifikation mit einbeziehen. Diese Funktion ist in Programm 4.9 zu finden.

Zum Schluss wurde eine Normalisierung mit Stemming und Stoppwort-Entfernung durchgeführt, um zu sehen, welche Auswirkungen beide Textverarbeitungsmethoden hervorrufen. Als Beispiel in Programm 4.10 wurde

[17] Mehr dazu ist in Abschnitt 2.5.4 zu finden.

**Programm 4.10:** Normalisierung der Daten mit Stemming (Porter Stemmer) und Stoppwort-Entfernung in Python.

```
1 def normalize(s):
2     words = tokenize.wordpunct_tokenize(s.lower().strip())
3     stemmer = stem.PorterStemmer()
4     s_words = [w for w in words if w not in stoprwords.words('english')]
5     return ' '.join([stemmer.stem(w) for w in s_words])
```

wieder der Porter Stemmer angewendet.

Eine Übersicht über die erstellten Sentimentanalyse-Applikationen und deren spezifische Textaufbereitungsmethoden ist in Abschnitt 3.1 zu finden.

## 4.5 Klassifikation mittels Naïve Bayes Verfahren

Es ist wichtig, die allgemeinen Konventionen von Klassifikationsproblemen in dem Bereich der Sentimentanalyse zu kennen und zu wissen, welche Features für Sentimentanalyse-Aufgaben verwendet werden [27]. Die Erklärung der Sentimentanalyse sowie die verwendeten Techniken für das Klassifikationsproblem sind in Kapitel 2 erklärt.

Ein weiteres Problem ist der Grad der Positivität, ob es nur eine binäre Klassifikation oder eine Klassifikation über mehrere Klassen ist. Nicht in jedem Bereich ist die Positivität gleich eingestuft, es muss zwischen politischen Reden, Movie-Reviews und Produktbewertungen unterschieden werden [27]. Nun steht die binäre Klassifikation, deren genaue Anwendung bei der Sentimentanalyse von Tweets und deren Verständnis im Vordergrund. Zunächst wird auf die Extraktion der Features und die Klassifikatoren eingegangen. Anschließend spielt die Evaluierung und deren Methodik eine wichtige Rolle.

### 4.5.1 Feature Extractor

Zunächst sollte die Definition eines Features klar sein. Wenn nicht, kann diese in Abschnitt 2.5.2 nachgelesen werden. Um verschiedene Sentimentanalyse-Ansätze für die Verbesserung der Sentimentanalyse von Tweets zu testen, wurden sechs verschiedene Feature Extraktoren entwickelt und implementiert. Diese unterscheiden sich in der Verwendung der generierten Features wie folgt:

- Nutzung aller Unigramme,
- Nutzung aller Bigramme,
- Nutzung aller Uni- und Bigramme,
- Nutzung der Unigramme mit den meisten Informationen,
- Nutzung der Bigramme mit den meisten Informationen und

- Nutzung der Uni- und Bigramme mit den meisten Informationen.

Alle sechs verschiedenen Ansätze sind vom grundlegenden Aufbau her gleich, deshalb wird nur jener, der die Uni- und Bigramme mit den meisten Informationen verwendet, ausführlich erklärt. Es wurde hierfür eine Methode „extract_features(msg, bestWords)“ erstellt, welche die Extraktion startet. Der erste Übergabeparameter „msg“ ist die Nachricht, für welche die Uni- und Bigramme erstellt werden sollen. Der zweite Parameter („bestWords“) ist eine Liste mit den besten Wörtern aus allen Trainingsdaten. Die Extraktions-Methode bei einer Nutzung der Uni- und Bigramme mit den meisten Informationen weist folgende Schritte für die Generierung auf:

1. Erstellung eines Sets, welches alle Features aufnimmt
2. Erstellung einer Liste, welche alle Wörter der übergebenen Nachricht normalisiert zur Weiterverarbeitung bereitstellt.
3. Liste mit normalisierten Wörtern befüllen.
4. Die besten Bigramme extrahieren und in Set abspeichern.
5. Die besten Unigramme extrahieren und Set mit diesen updaten.

Die Schritte 4 und 5 ändern sich je nach Applikation, so könnte jeweils einer dieser Schritte weggelassen oder auch die Verwendung der besten Uni- und beziehungsweise oder Bigramme mit allen ausgetauscht werden.

Der erstellte Code der Methode „extract_features“ für die Generierung der Uni- und Bigramme kann aus Programm 4.11 entnommen werden. In Zeile 12 und 14 sind jeweils Methoden für die Extraktion der besten Uni- und Bigramme angewendet, auf welche in den folgenden Unterabschnitten genau eingegangen wird.

**Programm 4.11:** Feature Extraktor mit der Nutzung der Uni- und Bigramme mit den meisten Informationen in Python.

```
1  def extract_features(msg, bestWords):
2      features = {}
3      # Liste mit allen normalisierten Wörtern von "msg"
4      n_msg = []
5
6      # Normalisierung der Wörter von "msg" und Speicherung in "n_msg"
7      for token in msg:
8          n_msg.append(normalize(token))
9
10     # Extraktion der besten Bigramme und Speicherung in "features"
11     features = extract_best_bigrams(n_msg)
12     # Extraktion der besten Unigramme und Updaten von "features"
13     features.update(extract_best_unigrams(n_msg, bestWords))
14
15     return features
```

### Lokalisierung und Generierung der besten Bigramme

Für die Erstellung der Bigramme und deren Auffinden kann aus dem Natural Language Processing Toolkit die „Bigram Collocation Finder“-Klasse verwendet werden. Wird ein Objekt dieser Klasse erstellt, liefert es eine Liste mit allen Bigrammen der übergebenen Nachricht zurück. Anschließend kann mit der Lokalisierung der Bigramme mit den meisten Informationen begonnen werden. Hierfür wurde die Anzahl auf die 200 Besten beschränkt. Die „Bigram Assoc Measures“-Klasse aus dem Natural Language Processing Toolkit bietet eine Methode, die besten Bigramme aufzufinden, welche die „chi-square“-Methodik implementiert. Für eine genaue Erklärung der Chi-Square-Methode siehe Abschnitt 4.5.1 oder „Foundations of Statistical Natural Language Processing“ (Jänner 1999) von Christopher Manning und Hinrich Schütze Kapitel 5 Collocations, Unterkapitel 5.3.3 Pearson's chi-square test, welches unter http://nlp.stanford.edu/fsnlp/promo/colloc.pdf gefunden werden kann. Die Implementierung dieser im NLTK kann unter http://www.nltk.org/_modules/nltk/metrics/association.html eingesehen werden.

### Lokalisierung und Generierung der besten Unigramme

Unigramme sind lediglich einzelne Wörter einer Nachricht, wobei hier der Text durch die Leerzeichen geteilt wird. Anschließend wird bestimmt, ob die einzelnen Unigramme genug Informationen enthalten, um als Feature ausgewählt zu werden. Dies basiert auf einer Liste von „besten“ Wörtern, die zuvor generiert wird. Kommt das Unigramm in dieser Liste vor, ist es als Feature tauglich und wird in die Liste der Features aufgenommen. Die Aufstellung der Wörter mit den meisten Informationen beruht auf der Frequenz des Wortes in allen Trainingsdaten in Relation zur Frequenz des Wortes in den Trainingsdaten für ein bestimmtes Label.

Die Methode zur Generierung der Liste mit den besten Wörtern aus einer Liste mit allen Wörtern ist in Algorithmus 4.1 zu finden. Zunächst werden die Frequenzen der Wörter in allen positiven und negativen Daten bestimmt. Jedes Wort wird in einer Liste abgelegt und die Anzahl des Vorkommens in den Daten ermittelt. Danach werden die Frequenzen für jedes Wort pro Klasse (positiv und negativ) separat bestimmt. Des Weiteren werden die Anzahl der unterschiedlichen positiven und negativen Wörter festgestellt und diese auch zu einer totalen Wörteranzahl zusammengeführt. Daraus werden mittels der Chi-Square Methode die Scores berechnet und anschließend $10,000$ Wörter mit den höchsten Scores zur Ermittlung der Unigramme zurückgeliefert.

**Algorithmus 4.1:** Generierung einer Liste mit Wörtern mit den meisten Informationen aus einer Liste mit allen Wörtern einer Datensammlung, basierend auf der Frequenz des Wortes.

```
1:  getBestWords(positiveData, negativeData)
    Berechnet für jedes Unigramm einen Score der die „Informationsstärke“
    des Wortes definiert.
2:
3:      wordFreqDist ← Frequenz für jedes Wort
4:      labelWordFreqDist ← Frequenz für jedes Wort pro Label
5:
6:      for word in positiveData do          ▷ positive Wortliste durchlaufen
7:          nWord ← normalisierte Form von word
8:          wordFreqDist[nWord] += 1
9:          labelWordFreqDist['pos'][nWord] += 1
10:     end for
11:
12:     for word in negativeData do          ▷ negative Wortliste durchlaufen
13:         nWord ← normalisierte Form von word
14:         wordFreqDist[nWord] += 1
15:         labelWordFreqDist['neg'][nWord] += 1
16:     end for
17:
18:     posWordCount ← Anzahl der unterschiedlichen postiven Wörter
19:     negWordCount ← Anzahl der unterschiedlichen negativen Wörter
20:     totalWordCount = posWordCount + negWordCount
21:
22:     wordScores ← Set mit der Informationsstärke für jedes Wort
23:     for word, freq in wordFreqDist do  ▷ alle Frequenzen durchlaufen
24:         posScore ← chi-square(labelWordFreqDist['pos'][word], (freq,
    posWordCount), totalWordCount)
25:         negScore ← chi-square(labelWordFreqDist['neg'][word], (freq,
    negWordCount), totalWordCount)
26:         wordScores[word] = posScore + negScore
27:     end for
28:
29:     best ← Set mit 10,000 Einträgen der besten Scores aus wordScores
30:     words ← Set mit allen Wörtern die in best vorkommen
31: return words
32: end
```

**Chi-Square Methodik**

Aus der online Dokumentation vom NLTK, im Speziellen der Implementierung von Chi-Square unter http://www.nltk.org/_modules/nltk/metrics/association.html, ist folgende Berechnung zu entnehmen:

$$chi_{square} = n_{xx} \cdot phi_{square}(n_{ii}, n_{ix}, n_{xi}, n_{xx}). \tag{4.1}$$

Bei der Chi-Square Methode wird berechnet wie unabhängig zwei Wörter des Bigramms sind. Die Berechnung verwendet die Phi-Square Methodik

$$phi_{square} = \frac{(n_{ii} \cdot n_{xx} - n_{ix} \cdot n_{xi})^2}{(n_{ii} + n_{ix})(n_{ii} + n_{xi})(n_{xx} + n_{ix})(n_{xx} + n_{xi})}. \tag{4.2}$$

In den Berechnungen werden die Parameter „$n_{ii}$“, „$n_{ix}$“, „$n_{xi}$“, „$n_{xx}$“ verwendet, welche folgende Bedeutungen aufweisen:

- $n_{ii}$: Anzahl der Vorkommen des Bigramms im Datenset,
- $n_{ix}$: Anzahl der Vorkommen des ersten Wortes im Datenset,
- $n_{xi}$: Anzahl der Vorkommen des zweiten Wortes im Datenset und
- $n_{xx}$: Anzahl aller Bigramme.

### 4.5.2 Generierung des Klassifikator-Modells und Klassifikationsvorgang

Um ein Klassifikations-Modell zu erstellen, müssen zunächst die Features aus den Trainingsdaten generiert werden. Wie die Trainingsdaten strukturiert sind, ist in Abschnitt 4.3 zu finden. Der Feature Extraktionsvorgang ist in Abschnitt 4.5.1 noch genau beschrieben. Anschließend kann mit der eigentlichen Erstellung begonnen werden. Diese ist durch das Natural Language Processing Toolkit einfach durchzuführen, indem ein Naïve Bayes Klassifikator erstellt und mit der „train“-Methode für die Klassifikation bereitgestellt wird. Die Erstellung ist in Programm 4.12 vermerkt.

Nachdem das Klassifikations-Modell trainiert wurde, kann ein Klassifikations-Test durchgeführt werden. Hierfür kann einfach die „classify“-Methode des Natural Language Processing Toolkits der „NaiveBayesClassifier“-Klasse verwendet werden, siehe Programm 4.12 Zeile 2.

### 4.5.3 Evaluierungsmethodik

Um die erstellten Klassifikations-Modelle vergleichen zu können, müssen diese mit einer einheitlichen Evaluierungsmethodik getestet werden. Hierfür werden zunächst zwei Sets oder auch Dictionaries mittels der NLTK „Collections“-Klasse erstellt. Ein Set ist für die „richtige“ Klasse der Einträge des Testsets („Referenz-Set“) und das zweite repräsentiert die einem Testeintrag durch das Modell zugewiesene Klasse („Beobachtungs-Set“). Bei

**Programm 4.12:** Erstellung des Klassifikators und Klassifikation mit NLTK in Python. „NaiveBayesClassifier“ = Klasse des NLTK, „train“ = Methode zum Training, „trainFeats“ = Trainingsset mit den Features, „feats“ = Testset und „observed“ = durch Klassifikator ermittelte Klassen

```
1 classifierModel = NaiveBayesClassifier.train(trainFeats)
2 observed = classifier.classify(feats)
```

**Programm 4.13:** Berechnung der Evaluierungskennzahlen für ein Klassifikations-Modell und Ausgabe der 20 informativsten Features mit NLTK in Python.

```
1 # Accuracy Berechnung
2 nltk.classify.util.accuracy(classifier, testFeats)
3 # Positive Precision
4 nltk.metrics.precision(referenceSet['pos'], observedSet['pos'])
5 # Positive Recall
6 nltk.metrics.recall(referenceSet['pos'], observedSet['pos'])
7 # Negative Precision
8 nltk.metrics.precision(referenceSet['neg'], observedSet['neg'])
9 # Negative Recall
10 nltk.metrics.recall(referenceSet['neg'], observedSet['neg'])
11
12 # Wichtigste Features ausgeben
13 classifier.show_most_informative_features(10)
```

beispielsweise zehn Einträgen in der Testdatensammlung haben die beiden erstellten Sets jeweils zehn Einträge, wobei an der ersten Stelle bei beiden das eigentliche beziehungsweise das ermittelte Label des ersten Eintrages des Testsets steht.

Ist die Klassifikation abgeschlossen und alle ermittelten Klassen in dem Beobachtungs-Set eingetragen, kann mit der eigentlichen Evaluierung begonnen und Accuracy, positive Precision, positive Recall, negative Precision und negative Recall berechnet werden. Alle maßgebenden Berechnungen können mit dem Klassifikations-Modell, dem Testset, dem Referenz-Set, dem Beobachtungs-Set und NLTK durchgeführt werden, siehe Programm 4.13. Die berechneten Kennzahlen haben einen Wertebereich von $[0 \dots 1]$ und sollen die prozentuellen Werte darstellen. Die Bedeutung der einzelnen Kennzahlen und deren Berechnung können in Abschnitt 5.1 nachgelesen werden.

Des Weiteren können noch die wichtigsten Features ausgegeben werden, wobei die Anzahl bestimmt werden kann. Dies ist ebenso in Programm 4.13 Zeile 13 zu finden.

# Kapitel 5

# Evaluierung

Um eine Evaluierung für die einzeln erstellten Sentimentanalyse-Applikationen durchzuführen und um festzustellen, ob ein Klassifikations-Ansatz ausgezeichnet, gut oder eher nicht so gut funktioniert, können verschiedene Kennzahlen berechnet werden. Eine wichtige davon ist „Accuracy“, die schon bei der Verwendung von Unigrammen allein gute Ergebnisse liefert, was durch folgendes Zitat aus [27] ausgedrückt wird:

> Indeed, applying machine learning techniques based on unigram models can achieve over 80% in accuracy, which is much better than the performance based on hand-picked keywords reported above.

Doch was bedeutet Accuracy? Es ist lediglich eine Kennzahl, die für die Genauigkeit des Klassifikations-Modells steht. Aber eine einzelne Zahl wird nicht ausreichen, um eine Aussage über die Klassifikation an sich zu treffen. Bei einem Test sollte erkannt werden, welche Informationen interessant sind und welche nicht. Wie gut ein Sentimentanalyse-Modell funktioniert, ist im Zusammenhang mit vier Faktoren beziehungsweise auch „Fehlern“, die ein solches Modell macht, zu ermitteln:

- „falsch positive“,
- „falsch negative“,
- „richtig positive“ und
- „richtig negative“.

Im Zusammenhang mit dieser Arbeit und dem Projekt werden lediglich Accuracy (Genauigkeit), Precision (Präzision) und Recall (Rückruf) berechnet und die Ergebnisse der einzelnen Sentimentanalyse-Applikationen verglichen.

## 5.1 Kennzahlen zur Evaluierung

Um die Ergebnisse der berechneten Kennzahlen einschätzen zu können, müssen diese zunächst definiert und deren Bedeutung erklärt werden. Des Weiteren ist die Berechnung dieser für das Verständnis ausschlaggebend.

### 5.1.1 Accuracy

Die Accuracy oder auch Genauigkeit ist der Prozentsatz der richtig klassifizierten Daten in Relation zu allen, zu klassifizierenden Daten. Diese Messung wird als generelle Basis für das Datenset gesehen, wobei zu den richtig klassifizierten Nachrichten auch Nulltreffer zählen. Nulltreffer sind jene Nachrichten, die nicht klassifiziert werden können, da aus der Nachricht kein Feature im Feature-Vektor des Klassifikations-Modelles vorkommt. Die genaue Berechnung lautet wie folgt [42]:

$$Accuracy = \frac{\sum correctPositive + \sum correctNegative}{allInstances}. \tag{5.1}$$

Der Term *correctPositive* steht für alle positiven Daten des Testdatensets, die richtig klassifiziert wurden, *correctNegative* steht für alle negativen Daten, die richtig klassifiziert wurden. *allInstances* repräsentieren lediglich die Anzahl aller zu klassifizierenden Daten.

### 5.1.2 Precision

Precision oder Präzision wird als der Prozentsatz der korrekt klassifizierten Daten gegenüber aller klassifizierten Daten definiert. Diese Kennzahl wird für die positiven und negativen Daten separat berechnet [42]:

$$PositivePrecision = \frac{\sum correctPositive}{\sum correctPositive + \sum falsePositive}, \tag{5.2}$$

$$NegativePrecision = \frac{\sum correctNegative}{\sum correctNegative + \sum falseNegative}. \tag{5.3}$$

*correctPositive* steht für alle positiven Daten, die richtig klassifiziert wurden, wobei *falsePositive* alle positiven Daten, die falsch eingeteilt wurden, repräsentiert. *correctNegative* sind alle negativen Instanzen, die richtig klassifiziert wurden und *falseNegative* wird durch alle negativen Daten, die falsch eingestuft wurden, definiert.

### 5.1.3 Recall

Der Recall oder Rückruf wird als Prozentsatz der korrekt identifizierten Daten gegenüber der Daten in der jeweiligen Klasse bezeichnet. Diese Kennzahl

wird ebenso wie die Precision für die positiven und negativen Datensets ermittelt [42]:

$$PositiveRecall = \frac{\sum correctPositive}{\sum correctPositive + \sum falseNegative}, \tag{5.4}$$

$$NegativeRecall = \frac{\sum correctNegative}{\sum correctNegative + \sum falsePositive}. \tag{5.5}$$

Hier steht *correctPositive* ebenso für alle positiven Daten, die richtig klassifiziert wurden, *correctNegative* für alle negativen Daten, die korrekt identifiziert wurden und *falsePositive* beziehungsweise *falseNegative* für alle positiven beziehungsweise negativen Daten, die falsch erkannt wurden.

## 5.2 Ergebnisse des Klassifikators trainiert mit Daten von Twitter

Es wurden unterschiedliche Klassifikatoren mit dem Twitter-Datenset trainiert, die Daten wurden jedoch unterschiedlich bearbeitet. Zunächst soll auf die Klassifikatoren eingegangen werden, die ohne Stemming und Stoppwort-Entfernung arbeiten. Hier wird, wie bereits mehrmals in Kapitel 3 und Kapitel 4 beschrieben, bei der Verwendung der n-Gramme unterschieden. Des Weiteren wird später in diesem Abschnitt auf die Auswirkungen von Stemming und der Stoppwort-Entfernung eingegangen und dessen Klassifikations-Ergebnisse verglichen.

In Tabelle 5.1 sind die Ergebnisse aller Klassifikatoren, trainiert mit dem Twitter-Datenset ohne Stemming und Stoppwort-Entfernung, aufgelistet. Getestet wurden diese Modelle ebenso mit Tweets, da versucht werden sollte die Sentimentanalyse für Tweets zu verbessern. Der Tabelle ist zu entnehmen, dass die beiden Klassifikatoren mit der Verwendung von Uni- und Bigrammen bessere Ergebnisse liefern als jene, die nur Uni- oder Bigramme für die Analyse heranziehen. Allerdings liegt die Richtigkeit der Klassifikation zwischen 67% und 78%, die offensichtlich verbesserungswürdig ist.

Betrachtet man die Ergebnisse für jede Klasse im einzelnen (Recall), so ist zu erkennen, dass die Klassifikatoren, trainiert mit Unigrammen, die besten Klassifikationsergebnisse für die positiven Tweets liefern. Bei den negativen dominieren eindeutig wieder die Uni- und Bigramme in Kombination. Die Prozentsätze liegen hier zwischen 61% und 79%, wobei die schlechtesten Ergebnisse bei der Verwendung von Bigrammen zu Tage treten. Die Screenshots der Ergebnisse sind in Abbildungen A.1 bis A.3 zu finden.

Bei der Präzision der Klassifikatoren ist ebenso zwischen den positiven und negativen zu unterscheiden. Hier tauschen die Unigramme mit den kombinierten Klassifikatoren ihre Plätze, allerdings sind die Unterschiede nicht gravierend groß. Die Bigramme liefern hier ebenso kein nennenswertes Ergebnis.

**Tabelle 5.1:** Tabelle mit den Testergebnissen der Applikationen, trainiert und getestet mit Tweets ohne Stemming oder Stoppwort-Entfernung. Acc = Accuracy, pos = positiv, neg = negativ, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Acc | Precision | | Recall | |
|---|---|---|---|---|---|
| | | pos | neg | pos | neg |
| u_all | 0.7648 | 0.7524 | 0.7785 | 0.7900 | 0.7395 |
| u_sel | 0.7578 | 0.7471 | 0.7694 | 0.7800 | 0.7355 |
| b_all | 0.6817 | 0.6613 | 0.7080 | 0.7460 | 0.6172 |
| b_sel | 0.6747 | 0.6623 | 0.6891 | 0.7140 | 0.6353 |
| ub_all | 0.7748 | 0.7835 | 0.7665 | 0.7600 | 0.7896 |
| ub_sel | 0.7608 | 0.7647 | 0.7569 | 0.7540 | 0.7675 |

**Tabelle 5.2:** Tabelle mit den Testergebnissen der Applikationen, trainiert und getestet mit Tweets mit Stemming aber ohne Stoppwort-Entfernung. Acc = Accuracy, pos = positiv, neg = negativ, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Acc | Precision | | Recall | |
|---|---|---|---|---|---|
| | | pos | neg | pos | neg |
| u_all | 0.7678 | 0.7691 | 0.7665 | 0.7660 | 0.7695 |
| u_sel | 0.7748 | 0.7823 | 0.7676 | 0.7620 | 0.7876 |
| b_all | 0.6947 | 0.6750 | 0.7195 | 0.7520 | 0.6373 |
| b_sel | 0.6817 | 0.6770 | 0.6866 | 0.6960 | 0.6673 |
| ub_all | 0.7868 | 0.7899 | 0.7837 | 0.7820 | 0.7916 |
| ub_sel | 0.7678 | 0.7648 | 0.7708 | 0.7740 | 0.7615 |

### 5.2.1 Auswirkungen von Stemming auf das Klassifikations-Modell trainiert und getestet mit Tweets

Um die Auswirkungen von Stemming auf die Klassifikationsergebnisse zu ermitteln, müssen die berechneten Kennzahlen aus den Tabellen 5.1 und 5.2 verglichen werden. Es ist zu erkennen, dass Stemming die Klassifikation geringfügig verbessert, siehe auch Abbildungen A.13 bis A.15.

### 5.2.2 Auswirkungen der Stoppwort-Entfernung auf alle Applikationen trainiert und getestet mit Tweets

Die Stoppwort-Entfernung beeinflusst die Klassifikation bei den Trainings- und Testdaten in deren Performance. Um zu bestimmen, ob die Stoppwort-Entfernung eine Verbesserung der Klassifikation hervorruft, müssen die Ergebnisse aus der Tabelle der Klassifikatoren trainiert mit Tweets ohne Berücksichtigung von Stemming und Stoppwort-Entfernung (Tabelle 5.1) mit den Ergebnissen der Klassifikatoren, die eine Entfernung der Stoppwörter implementieren (Tabelle 5.3), verglichen werden. Dabei ist zu erkennen, dass die Stoppwort-Entfernung bei diesen Tests nicht wirklich eine Verbesserung mit sich bringt. Es arbeiten alle Klassifikatoren schlechter als ohne Stoppwort-Entfernung, außer jener, der die Uni- und Bigramme mit den meisten Informationen für die Klassifikation heranzieht. Allerdings ist die Verbesserung bei dem einen Klassifikations-Modell nicht gerade mit großer Bedeutung zu interpretieren, da sich die Werte nur geringfügig unterscheiden, siehe auch Abbildungen A.19 bis A.21.

**Tabelle 5.3:** Tabelle mit den Testergebnissen der Applikationen, trainiert und getestet mit Tweets mit Stoppwort-Entfernung aber ohne Stemming. Acc = Accuracy, pos = positiv, neg = negativ, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Acc | Precision | | Recall | |
|---|---|---|---|---|---|
| | | pos | neg | pos | neg |
| u_all | 0.7482 | 0.7435 | 0.7533 | 0.7681 | 0.7277 |
| u_sel | 0.7492 | 0.7596 | 0.7391 | 0.7391 | 0.7596 |
| b_all | 0.6201 | 0.5816 | 0.7547 | 0.8923 | 0.3404 |
| b_sel | 0.6254 | 0.5863 | 0.7534 | 0.8861 | 0.3574 |
| ub_all | 0.7314 | 0.7321 | 0.7306 | 0.7412 | 0.7213 |
| ub_sel | 0.7817 | 0.7680 | 0.7977 | 0.8157 | 0.7468 |

### 5.2.3 Auswirkungen von Stemming und der Stoppwort-Entfernung auf das Klassifikations-Modell trainiert und getestet mit Tweets

Die Anwendung von Stemming und Stoppwort-Entfernung können ebenso unterschiedliche Ergebnisse hervorrufen. Allerdings ist keine große Verbesserung zu erwarten, da die Ergebnisse mit Stemming und Stoppwort-Entfernung separat angewendet nur geringe Verbesserungen oder auch Ver-

**Tabelle 5.4:** Tabelle mit den Testergebnissen der Applikationen, trainiert und getestet mit Tweets mit Stemming und Stoppwort-Entfernung. Acc = Accuracy, pos = positiv, neg = negativ, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Acc | Precision | | Recall | |
|---|---|---|---|---|---|
| | | pos | neg | pos | neg |
| u_all | 0.7335 | 0.7223 | 0.7466 | 0.7702 | 0.6957 |
| u_sel | 0.7377 | 0.7335 | 0.7423 | 0.7578 | 0.7170 |
| b_all | 0.6254 | 0.5880 | 0.7384 | 0.8716 | 0.3723 |
| b_sel | 0.6285 | 0.5905 | 0.7417 | 0.8716 | 0.3787 |
| ub_all | 0.7618 | 0.7623 | 0.7613 | 0.7702 | 0.7532 |
| ub_sel | 0.7545 | 0.7495 | 0.7599 | 0.7743 | 0.7340 |

schlechterungen (vor allem bei Stoppwort-Entfernung) hervorgebracht haben. In Tabelle 5.4 wird die Performance der Klassifikatoren, trainiert und getestet mit Tweets unter Anwendung von Stemming und Stoppwort-Entfer-nung, gemessen (Abbildungen A.25 bis A.27). Wie schon zu erwarten, sind die Ergebnisse nicht besser, als die Anwendung von Stemming allein.

## 5.3 Ergebnisse mit dem Movie-Review-Datenset

Die Movie-Reviews als Daten für eine Klassifikation wurden zunächst selbst getestet, weshalb sechs Klassifikations-Modelle erstellt wurden, die mit den Movie-Reviews trainiert und getestet wurden. Dies war notwendig, um zu sehen, ob diese Daten für eine Klassifikation brauchbar sind. Um die Sentimentanalyse für Tweets zu verbessern, indem Twitter- und Movie-Review-Daten kombiniert werden, müssen die Modelle, die nur Movie-Reviews berücksichtigen, bessere Ergebnisse liefern als jene Klassifikatoren der Twitter-Daten.

Die Ergebnisse der Movie-Review Klassifikatoren sind in Tabelle 5.5 übersichtlich dargestellt und den Abbildungen A.4 bis A.6 zu entnehmen. Es ist zu erkennen, dass die Klassifikatoren, welche die n-Gramme mit den meisten Informationen für eine Klassifikation heranziehen, eine weitaus bessere Performance aufweisen, als die Klassifikatoren der Twitter-Daten. Das Klassifikations-Modell trainiert mit den Uni- und Bigrammen der Movie-Review-Daten kommt bei den Performance-Werten sogar über 90%, was schon wirklich ein sehr gutes Ergebnis darstellt. Die restlichen Werte sind, verglichen mit denen aus Tabelle 5.1 (Klassifikations-Modelle trainiert und getestet mit Tweets), auch ein wenig verbessert worden.

**Tabelle 5.5:** Tabelle mit den Testergebnissen der Applikationen, trainiert und getestet mit Movie-Reviews ohne Stemming oder Stoppwort-Entfernung. Acc = Accuracy, pos = positiv, neg = negativ, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Acc | Precision | | Recall | |
|---|---|---|---|---|---|
| | | pos | neg | pos | neg |
| u_all | 0.7240 | 0.6481 | 0.9590 | 0.9800 | 0.4680 |
| u_sel | 0.9280 | 0.8905 | 0.9735 | 0.9760 | 0.8800 |
| b_all | 0.7640 | 0.6886 | 0.9400 | 0.9640 | 0.564 |
| b_sel | 0.7860 | 0.8488 | 0.7424 | 0.6960 | 0.8760 |
| ub_all | 0.7480 | 0.6722 | 0.9429 | 0.9680 | 0.5280 |
| ub_sel | 0.9140 | 0.9091 | 0.9190 | 0.9200 | 0.9080 |

## 5.4 Ergebnisse mit beiden Datensets in Kombination

Nun wird auf die Kombination der Tweets und Reviews in Sachen Klassifikations-Performance eingegangen, welche eine Verbesserung der Klassifikation von Tweets hervorrufen soll und somit die wichtigsten Testergebnisse liefern.

Zunächst wird wieder die Klassifikation ohne Stemming und Stoppwort-Entfernung betrachtet. Hier werden wieder sechs Klassifikations-Modelle erstellt, die in der Verwendung der n-Gramme unterschiedlich arbeiten. Diese werden mit drei unterschiedlichen Datensets getestet:

- Twitter-Testdaten,
- Movie-Review-Testdaten und
- kombinierte Testdaten (Tweets und Movie-Reviews).

Dies soll zeigen, ob eine Kombination der Daten beim Training der Klassifikatoren für Tweets am optimalsten arbeitet, oder ob diese Methode für eine Klassifikation in positiv oder negativ bei anderen Daten besser funktioniert.

Die Testergebnisse in Sachen Performance der sechs getesteten Klassifikatoren mit drei unterschiedlichen Datensammlungen sind in Tabelle 5.6 aufgelistet oder in den Abbildungen A.7 bis A.12 zu finden. Pro Klassifikations-Modell sind in fünf Spalten jeweils drei Zeilen mit Performancewerten, wobei die erste Zeile die Tests mit den Tweets, die zweite mit den Movie-Reviews und die dritte mit den kombinierten Daten darstellen. Es ist zu entnehmen, dass die Testergebnisse deutlich besser ausfallen als jene zuvor. Es liegen alle bei einem Richtigkeitswert (Accuracy) von über 80%, ausgenommen je-

ner, der die Bigramme mit den meisten Informationen verwendet und mit Movie-Reviews getestet wurde. Alle weiteren Werte liegen über 70%, wobei es bei der Klassifikation von Movie-Reviews unter Verwendung von Bigrammen mit den meisten Informationen drei Ausreißer gibt. Tweets als Testdaten liefern bei allen Klassifikations-Modellen bessere Performance-Werte als die anderen beiden Testdaten-Gruppen, weshalb bewiesen wurde, dass eine Kombination von Movie-Reviews und Tweets zum Training sehr gut für die Klassifikation von Tweets arbeitet. Die Klassifikatoren, die nur Bigramme oder Uni- und Bigramme für die Klassifikation heranziehen, haben Performancewerte bei 98%.

**Tabelle 5.6:** Tabelle mit den Testergebnissen der Applikationen, trainiert mit dem kombinierten Datenset bestehend aus Tweets und Movie-Reviews ohne Stemming oder Stoppwort-Entfernung. Acc = Accuracy, pos = positiv, neg = negativ, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Acc | Precision | | Recall | |
|---|---|---|---|---|---|
| | | pos | neg | pos | neg |
| u_all | 0.8999 | 0.8968 | 0.9030 | 0.9040 | 0.8958 |
| | 0.8100 | 0.7818 | 0.8444 | 0.8600 | 0.7600 |
| | 0.8032 | 0.7705 | 0.8450 | 0.8640 | 0.7423 |
| u_sel | 0.8018 | 0.8069 | 0.7968 | 0.7940 | 0.8096 |
| | 0.9200 | 0.9200 | 0.9200 | 0.9200 | 0.9200 |
| | 0.9113 | 0.9130 | 0.9096 | 0.9093 | 0.9132 |
| b_all | 0.9860 | 0.9899 | 0.9821 | 0.9820 | 0.9899 |
| | 0.8140 | 0.7794 | 0.8584 | 0.8760 | 0.7520 |
| | 0.8025 | 0.7683 | 0.8468 | 0.8667 | 0.7383 |
| b_sel | 0.9890 | 0.9880 | 0.9899 | 0.9900 | 0.9880 |
| | 0.7100 | 0.7665 | 0.6733 | 0.6040 | 0.8160 |
| | 0.8105 | 0.7724 | 0.6714 | 0.5973 | 0.8238 |
| ub_all | 0.9820 | 0.9801 | 0.9839 | 0.9840 | 0.9800 |
| | 0.8360 | 0.8022 | 0.8784 | 0.8920 | 0.7800 |
| | 0.8205 | 0.7813 | 0.8727 | 0.8907 | 0.7503 |
| ub_sel | 0.9830 | 0.9860 | 0.9801 | 0.9800 | 0.9860 |
| | 0.8740 | 0.9119 | 0.8425 | 0.8280 | 0.9200 |
| | 0.8846 | 0.9104 | 0.8618 | 0.8533 | 0.9159 |

### 5.4.1 Auswirkungen von Stemming auf das Klassifikations-Modell trainiert mit dem kombinierten Datenset

Die Anwendung von Stemming auf die Trainings- und Testdaten kann die Performance der Klassifikatoren aus Tabelle 5.6 wiederum positiv oder negativ beeinflussen. Hier wurden allerdings nur die Tweets für die Tests herangezogen, da diese ja das wichtigste Testdaten-Set darstellen.

In Tabelle 5.7 oder in Abbildungen A.16 bis A.18 werden die Performance-Werte der Klassifikatoren mit Stemming übersichtlich dargestellt. Verglichen mit den vorherigen Werten aus Tabelle 5.6 sind keine gravierenden Unterschiede zu erkennen. Da alle Ergebnisse hervorragend sind, ist Stemming nicht unbedingt notwendig, um Tweets in positiv oder negativ einzustufen. Auch hier arbeiten die Klassifikatoren mit Verwendung von Uni- und Bigrammen am besten, allerdings sind die anderen nicht wesentlich schlechter, sondern ziemlich nah an der Performance der Besten dran.

**Tabelle 5.7:** Tabelle mit den Testergebnissen der Klassifikations-Modelle, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews mit Stemming aber ohne Stoppwort-Entfernung und getestet mit Tweets. Acc = Accuracy, pos = positiv, neg = negativ, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Acc | Precision | | Recall | |
|---|---|---|---|---|---|
| | | pos | neg | pos | neg |
| u_all | 0.8930 | 0.8938 | 0.8920 | 0.8920 | 0.8938 |
| u_sel | 0.8018 | 0.7949 | 0.8090 | 0.8140 | 0.7896 |
| b_all | 0.9790 | 0.9761 | 0.9819 | 0.9820 | 0.9760 |
| b_sel | 0.9820 | 0.9839 | 0.9800 | 0.9800 | 0.9840 |
| ub_all | 0.9750 | 0.9760 | 0.9740 | 0.9740 | 0.9760 |
| ub_sel | 0.9810 | 0.9898 | 0.9724 | 0.9720 | 0.9900 |

### 5.4.2 Auswirkungen der Stoppwort-Entfernung auf alle Applikationen trainiert mit dem kombinierten Datenset

Die Entfernung von Stoppwörtern kann die Performance der Klassifikation beeinflussen. Deshalb wurden hier die Klassifikations-Modelle mit Tweets getestet und deren Ergebnisse aufgelistet (Tabelle 5.8). Die ausführlichen Ergebnisse sind in Abbildungen A.22 bis A.24 zu finden. Auch diese Resultate sind mit denen aus Tabelle 5.6 ziemlich gleich auf. Die Verwendung von Unigrammen allein schneidet allerdings etwas schlechter als die anderen ab.

**Tabelle 5.8:** Tabelle mit den Testergebnissen der Klassifikations-Modelle, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews mit Stoppwort-Entfernung aber ohne Stemming und getestet mit Tweets. Acc = Accuracy, pos = positiv, neg = negativ, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Acc | Precision | | Recall | |
|---|---|---|---|---|---|
| | | pos | neg | pos | neg |
| u_all | 0.9213 | 0.9359 | 0.9072 | 0.9068 | 0.9362 |
| u_sel | 0.8090 | 0.8116 | 0.8064 | 0.8116 | 0.8064 |
| b_all | 0.9969 | 0.9979 | 0.9958 | 0.9959 | 0.9979 |
| b_sel | 0.9969 | 0.9938 | 1.0000 | 1.0000 | 0.9936 |
| ub_all | 0.9874 | 0.9836 | 0.9914 | 0.9917 | 0.9830 |
| ub_sel | 0.9927 | 0.9897 | 0.9957 | 0.9959 | 0.9894 |

### 5.4.3 Auswirkungen von Stemming und Stoppwort-Entfernung auf das Klassifikations-Modell trainiert mit dem kombinierten Datenset

Auch Stemming und Stoppwort-Entfernung in Kombination könnten andere Ergebnisse als die separate Anwendung liefern. Allerdings ist dies Aufgrund der vorherigen Tests nicht zu erwarten, da die Ergebnisse alle ziemlich gleichauf liegen. In Tabelle 5.9 sind die Testergebnisse der Klassifikations-Modelle mit Stemming und Stoppwort-Entfernung, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets, dargestellt. Die Screenshots dazu sind in Abbildungen A.28 bis A.30 zu finden.

Die Verwendung von Uni- und Bigrammen für die Klassifikation weisen Performance-Ergebnisse von 99% auf und liefern somit die bis dato besten Ergebnisse in Sachen Klassifikation von Tweets in dieser Arbeit. Die Verwendung von Bigrammen allein mit Stemming und Stoppwort-Entfernung liegt knapp dahinter, die Unigramme jedoch liefern eher schlechte Ergebnisse, wenn man die herausragenden anderen betrachtet.

**Tabelle 5.9:** Tabelle mit den Testergebnissen der Klassifikations-Modelle, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews mit Stemming und Stoppwort-Entfernung und getestet mit Tweets. Acc = Accuracy, pos = positiv, neg = negativ, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Acc | Precision | | Recall | |
|---|---|---|---|---|---|
| | | pos | neg | pos | neg |
| u_all | 0.8688 | 0.8792 | 0.8587 | 0.8592 | 0.8787 |
| u_sel | 0.8038 | 0.7996 | 0.8083 | 0.8178 | 0.7894 |
| b_all | 0.9958 | 0.9938 | 0.9979 | 0.9979 | 0.9936 |
| b_sel | 0.9927 | 0.9897 | 0.9957 | 0.9959 | 0.9894 |
| ub_all | 0.9947 | 0.9959 | 0.9936 | 0.9938 | 0.9957 |
| ub_sel | 0.9937 | 0.9918 | 0.9957 | 0.9959 | 0.9915 |

# Kapitel 6

# Zusammenfassende Diskussion

Eine Verbesserung der Sentimentanalyse von Tweets kann mittels der Kombination von Twitter- und Movie-Review-Daten hervorgebracht werden, wie in Kapitel 5 den Testergebnissen zu entnehmen ist. Zu Beginn war die Performance beziehungsweise die Richtigkeit der Klassifikations-Modelle zwischen 68% und 78%, wobei bei diesen nur Tweets für das Training herangezogen wurden. Die Verwendung des kombinierten Datensets aus Tweets und Movie-Reviews brachte Richtigkeitswerte von 80% bis 99% hervor, was eine deutliche Verbesserung darstellt. Diese Werte sind in den Tabellen 5.1 und 5.6 oder auch Tabelle 6.1 und 6.2 zu finden.

Wird nun Stemming zu den Klassifikations-Modellen hinzugefügt, ändern sich natürlich die Accuracy-Werte. Bei den Klassifikatoren, trainiert mit dem Twitter-Datenset, kann durch Hinzufügen von Stemming bei jedem eine Verbesserung hervorgerufen werden. Die Richtigkeit steigt um 0.3% bis 1.7% und durchschnittlich um 0.983%. Die Modelle, trainiert mit dem kombinierten Datenset, hingegen verschlechtern sich um 0.2% bis 0.7%, außer bei jenem, trainiert mit den Unigrammen mit den meisten Informationen, hier bleibt die Accuracy gleich. Die durchschnittliche Verschlechterung liegt bei 0.43%. Um diese Werte vergleichen beziehungsweise ermitteln zu können, sind die genauen Richtigkeits-Werte der Modelle mit und ohne Stemming in Tabelle 6.1 übersichtlich zusammengefasst. Stemming verbessert also nicht immer eine Klassifikation. In diesem Fall ist es für Klassifikatoren, welche ausschließlich mit Tweets trainiert wurden, ein Gewinn. Für das kombinierte Datenset hingegen keineswegs, obwohl die Verschlechterung der Accuracy-Werte nicht so gravierend ausfällt.

Auch die Entfernung von Stoppwörtern wurde in Kapitel 5 Evaluierung durchgeführt und auf die Änderung der Performance bewertet. In Tabelle 6.2 sind die Accuracy-Werte der Klassifikatoren, trainiert mit Tweets und dem kombinierten Datenset mit und ohne Stoppwort-Entfernung, zu fin-

**Tabelle 6.1:** Tabelle mit den Accuracy-Testergebnissen aller Klassifikations-Modelle, trainiert mit Tweets und dem kombinierten Datenset mit und ohne Stemming im Vergleich. Tweet = Klassifikations-Modelle trainiert mit Tweets, Tweet-Rev. = Klassifikations-Modelle trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews, Stemming = Anwendung von Stemming auf die Trainings- und Testdaten, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Tweet | Tweet-Rev. | Stemming | |
|---|---|---|---|---|
| | | | Tweet | Tweet-Rev. |
| u_all | 0.7648 | 0.8999 | 0.7678 | 0.8930 |
| u_sel | 0.7578 | 0.8018 | 0.7748 | 0.8018 |
| b_all | 0.6817 | 0.9860 | 0.6947 | 0.9790 |
| b_sel | 0.6747 | 0.9890 | 0.6817 | 0.9820 |
| ub_all | 0.7748 | 0.9820 | 0.7868 | 0.9750 |
| ub_sel | 0.7608 | 0.9830 | 0.7678 | 0.9810 |

den. Es kann herausgelesen werden, dass bei den Klassifikations-Modellen, trainiert mit Tweets, eine Verschlechterung von 0.8% bis 6.16% verursacht wurde, außer bei jenem Modell, welches die Uni- und Bigramme mit den meisten Informationen verwendet. Dieses wurde um 2.09% verbessert, somit ergibt sich ein durchschnittlicher Rückgang von 2.64%. Betrachtet man die Performance-Ergebnisse der Klassifikatoren, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews, ist zu erkennen, dass die Modelle, bei denen die Stoppwort-Entfernung angewandt wurde, eine Verbesserung von 0.54% bis 2.14% zu Tage legen, was einer durchschnittlichen Steigerung von 1.04% entspricht. Die Stoppwort-Entfernung rentiert sich also nur bei dem kombinierten Datenset, da sonst ein Rückgang der Performance zu erwarten ist. Die durchschnittliche Verbesserung ist zwar nicht sehr groß, aber dennoch vorhanden, deshalb sollte für jeden Anwendungsfall an sich entschieden werden, ob der zusätzliche Implementierungs- und auch Rechenaufwand die Sache wert ist, da auch die beiden Klassifikations-Modelle, trainiert mit dem kombinierten Datenset und der Verwendung von Uni- und Bigrammen, schon ziemlich nah an den Werten der Modelle mit Stoppwort-Entfernung liegen.

**Tabelle 6.2:** Tabelle mit den Accuracy-Ergebnissen aller Klassifikations-Modelle, trainiert mit Tweets und dem kombinierten Datenset aus Tweets und Reviews mit und ohne Stoppwort-Entfernung im Vergleich. Tweet = Klassifikations-Modelle trainiert mit Tweets, Tweet-Rev. = Klassifikations-Modelle trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews, Stoppwort = Anwendung von Stoppwort-Entfernung auf die Trainings- und Testdaten, u = Verwendung von Unigrammen, b = Verwendung von Bigrammen, ub = Verwendung von Uni- und Bigrammen, all = Verwendung von allen n-Grammen, sel = Verwendung der n-Gramme mit den meisten Informationen.

| Datei | Tweet | Tweet-Rev. | Stoppwort | |
|---|---|---|---|---|
| | | | Tweet | Tweet-Rev. |
| u_all | 0.7648 | 0.8999 | 0.7482 | 0.9213 |
| u_sel | 0.7578 | 0.8018 | 0.7492 | 0.8090 |
| b_all | 0.6817 | 0.9860 | 0.6201 | 0.9969 |
| b_sel | 0.6747 | 0.9890 | 0.6254 | 0.9969 |
| ub_all | 0.7748 | 0.9820 | 0.7314 | 0.9874 |
| ub_sel | 0.7608 | 0.9830 | 0.7817 | 0.9927 |

# Kapitel 7

# Schlussbemerkungen

Abschließend soll noch einmal kurz der eigene Ansatz zur Verbesserung der Sentimentanalyse dargebracht werden, um ein kurzes Fazit der Ergebnisse und der Aufgabenstellung ziehen zu können. Des Weiteren soll ein kurzer Ausblick auf eventuell mögliche Verbesserungen und Erweiterungen gegeben werden.

## 7.1 Ansatz

Um die Sentimentanalyse für Tweets zu verbessern, wurde ein eigener Ansatz entwickelt, der für die Klassifikation in positiv und negativ ein kombiniertes Datenset aus Tweets und Movie-Reviews zum Training des Klassifikations-Modelles verwendet. Für die Klassifikation an sich wurde das Naïve Bayes Verfahren verwendet, um die Wahrscheinlichkeit für die jeweilige Zugehörigkeit zu einer Klasse zu berechnen. Um eine optimale Kombination der Parameter einer Sentimentanalyse für Tweets zu ermitteln, wurden unterschiedliche Datenaufbereitungsmethoden ausprobiert. So wurden verschiedene n-Gramme aus den Trainingsdaten generiert und zur Erstellung der Klassifikatoren verwendet. Durch die Tests der unterschiedlichen Modelle wurde ermittelt, dass Uni- und Bigramme gemeinsam ein sehr gutes Ergebnis liefern. Die Uni- und Bigramme separat arbeiten ganz passabel, jedoch spielen hier die anderen Parameter, wie beispielsweise die Trainingsdaten (Tweets, Movie-Reviews oder beides), Stemming und Stoppwort-Entfernung eine Rolle.

Die Anwendung von Stemming und Stoppwort-Entfernung hat nicht so großen Einfluss auf die Ergebnisse. Je nach Klassifikator verbessern oder verschlechtern sich die Resultate. Der durchschnittliche Rückgang mit Stemming allein liegt bei 0.43%, welcher hauptsächlich durch die Verwendung von Bigrammen und Uni- und Bigrammen zusammen bei dem kombinierten Datenset zu Tage tritt und die Verbesserung beträgt durchschnittlich 0.98%, welche durch das Trainieren mit dem Twitter-Datenset hervorgerufen wird.

Diese Werte sind nicht gerade gravierend für die Ergebnisse selbst, wobei hingegen die Stoppwort-Entfernung schon mehr Einfluss nimmt. Hier tritt ein Rückgang von durchschnittlich 2.64% auf, wenn die Klassifikations-Modelle mit Tweets trainiert werden. Bei einem Training mit dem kombinierten Datenset kommt eine Verbesserung von durchschnittlich 1.04% vor.

Die performantesten Klassifikatoren wurden durch die Kombination der Tweets und Movie-Reviews als Trainingsdaten erreicht.

## 7.2 Fazit

Die Sentimentanalyse an sich ist nicht schwer zu verstehen, die genaue Arbeitsweise dahinter ist jedoch etwas komplexer. Vor allem das Verständnis des Naïve Bayes Verfahrens ist ein schwieriges Unterfangen und erfordert einiges an Zeit. Der Klassifikations-Vorgang selbst ist dann nicht mehr so schwierig, sofern Naïve Bayes keine Probleme macht, da die Klasse dem zu klassifizierenden Text zugewiesen wird, die am wahrscheinlichsten ist.

Sind die verwendeten Bibliotheken, wie beispielsweise NLTK, Tweepy oder Riak, erst einmal installiert beziehungsweise eingebunden, so steht der Verwendung eigentlich nicht viel im Wege, da es wirklich gute Dokumentationen für den Gebrauch dieser mit Python gibt. Allerdings muss Python selbst auch angewandt werden können, da sonst die Verwendung der Bibliotheken zu komplex wäre.

Der eigene Ansatz zur Verbesserung der Sentimentanalyse durch die Verwendung von zwei verschiedenen Datensammlungen, um die Klassifikation durchzuführen, lieferte, wie erhofft, gute Resultate, die im Abschnitt zuvor (Abschnitt 7.1) kurz zusammengefasst und in Kapitel 5 und Kapitel 6 ausführlich besprochen wurden. Die Movie-Review Datensammlung von Bo Pang und Lillian Lee ist wirklich gut für eine Sentimentanalyse von Tweets geeignet und in Kombination mit dem selbst erstellten Twitter-Datenset herausragend gut. Das eben erwähnte Datenset mit Tweets wurde mittels positiven und negativen Emoticons erstellt, was eine gute Grundlage darstellt und auch annehmbar funktioniert, wie in Abschnitt 3.6 durch die ähnlichen Ansätze verstärkt wird. Allerdings ist die Spracherkennung mittels des Twitter-Sprach-Codes in den Metadaten nicht wirklich nützlich, um eine Aussage über die Sprache machen zu können, da viele Einträge, die als Englisch vermerkt sind, dann nicht komplett in englischer Sprache verfasst sind, sondern andere Sprachen wie Spanisch beinhalten oder vollkommen in dieser Sprache formuliert wurden.

## 7.3 Ausblick

Die Sentimentanalyse selbst ist – vor allem für Unternehmen – ein sehr wichtiges Unterfangen. Tweets stellen eine heutzutage wichtige Kommuni-

kationsform dar. Deshalb ist es vorteilhaft, alle negativen Einträge schnell herauszufiltern und darauf reagieren zu können, da pro Tag Unmengen generiert werden.

Die API von Twitter bietet schon wirklich sehr viele Möglichkeiten, um an die Daten aus dem Public Stream zu kommen, aber die Metadaten dieser Einträge sind in Sachen Sprache nicht gerade gut gepflegt. Somit könnte eine separate Sprachidentifikation bei der Erstellung eines Datensets mit positiven und negativen Tweets eingebaut werden. Diese Identifikation ist wiederum ein eigenes Klassifikations-Problem, bei welchem eigene Klassifikatoren mit Unmengen an Trainingsdaten in den gewählten Sprachen erstellt werden müssen. Vor allem bei Daten von Twitter muss bei einer Spracherkennungs-Applikation darauf geachtet werden, dass die Trainingsdaten auch Abkürzungen und Slang innehaben, ansonsten würde die Qualität der Sprachidentifikation massiv leiden. Durch diese Gegebenheiten stellt die Identifikation der Sprache eines Tweets kein leichtes Unterfangen dar.

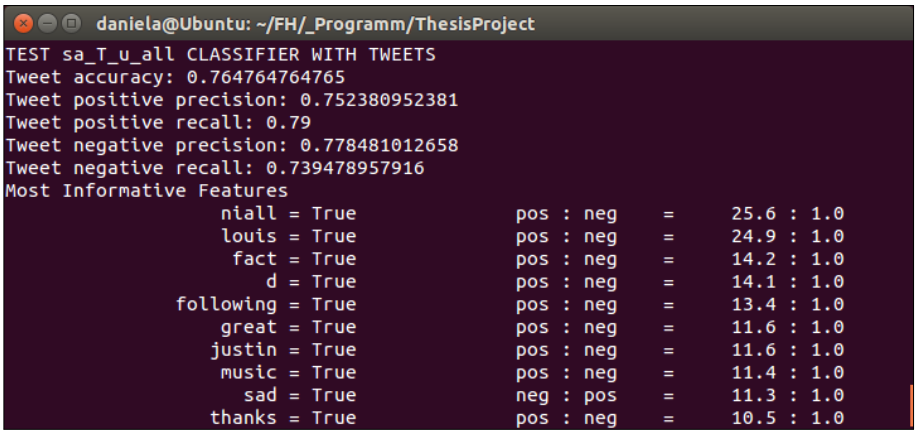
Des Weiteren wird bei Tweets auch oft Sarkasmus angewandt. Diesen zu erkennen ist ebenso ein separates Klassifikations-Problem, welches aufgrund der Komplexität der Aufgabe schwerwiegend ist, da sogar viele Menschen Probleme damit haben, diesen in Sprache oder Text zu erkennen.

Wenn all diese Klassifikations-Probleme vereint werden würden und bei den Einzelnen eine gute Performance erzielt werden kann, dann würde ein großer Schritt bei der Automatisierung des Erkennens von negativen Nachrichten, vor allem in sozialen Netzwerken, gemacht werden.

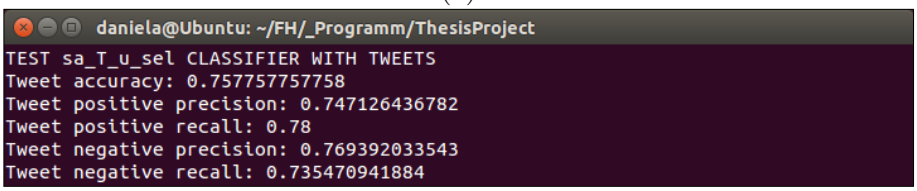
# Anhang A

# Testergebnisse

## A.1 Ergebnisse mit dem Tweet-Datenset

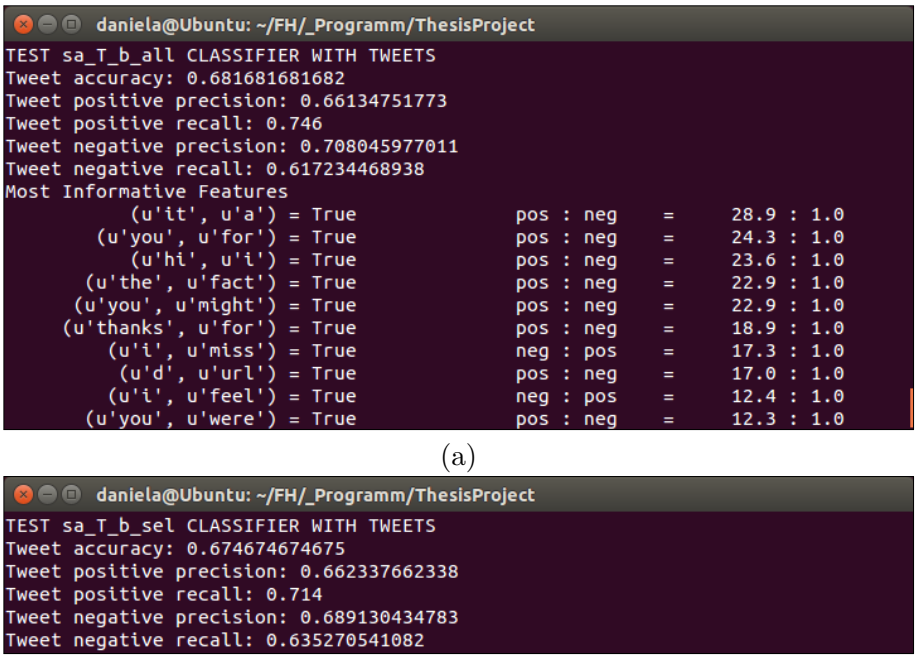



(a)




(b)

**Abbildung A.1:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Unigrammen (a) und den 10,000 Unigrammen mit den meisten Informationen (b).

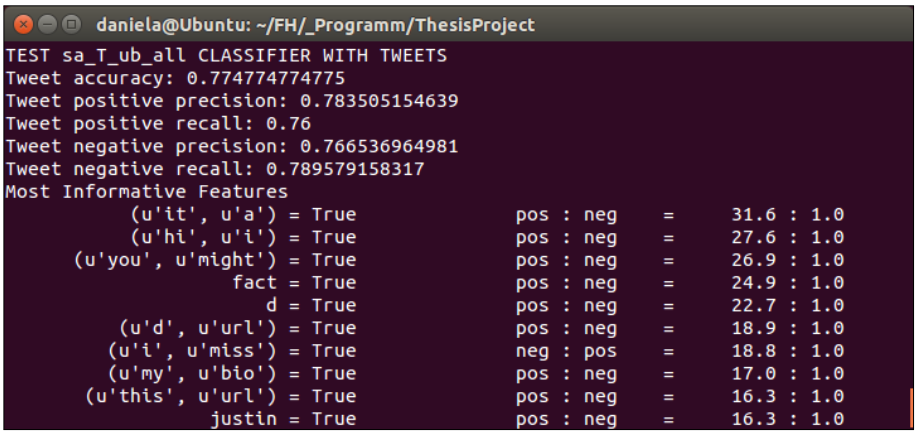



(a)

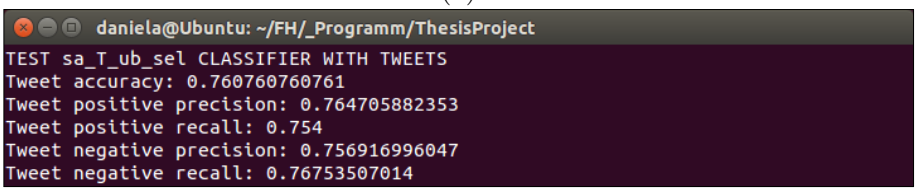



(b)

**Abbildung A.2:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Bigrammen (a) und den 200 Bigrammen mit den meisten Informationen (b).



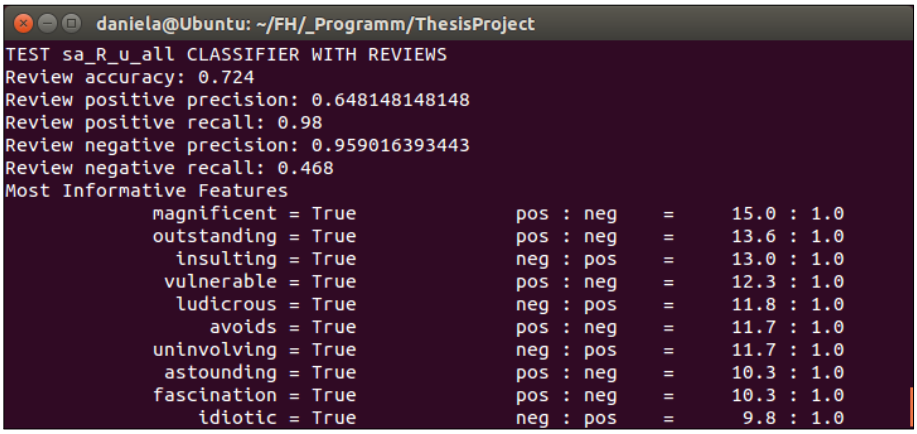

(a)



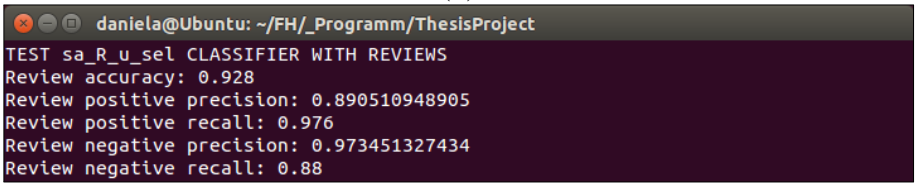

(b)

**Abbildung A.3:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Uni- und Bigrammen (a) und 10,000 Uni- und 200 Bigrammen mit den meisten Informationen (b).

## A.2 Ergebnisse mit dem Movie-Review-Datenset

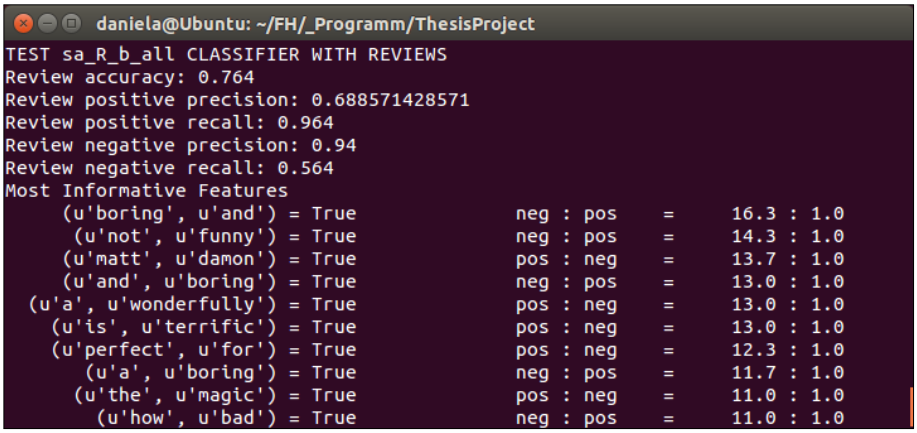



(a)

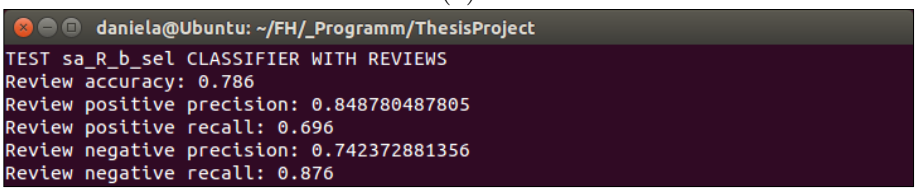



(b)

**Abbildung A.4:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Movie-Reviews unter Verwendung von allen Unigrammen (a) und den 10,000 Unigrammen mit den meisten Informationen (b).

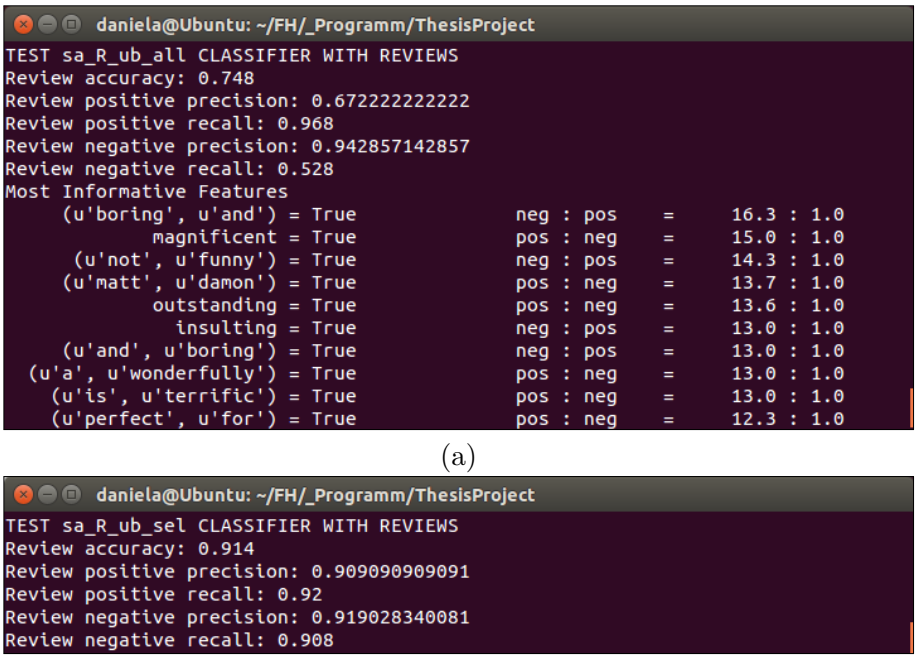



(a)




(b)

**Abbildung A.5:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Movie-Reviews unter Verwendung von allen Bigrammen (a) und den 200 Bigrammen mit den meisten Informationen (b).



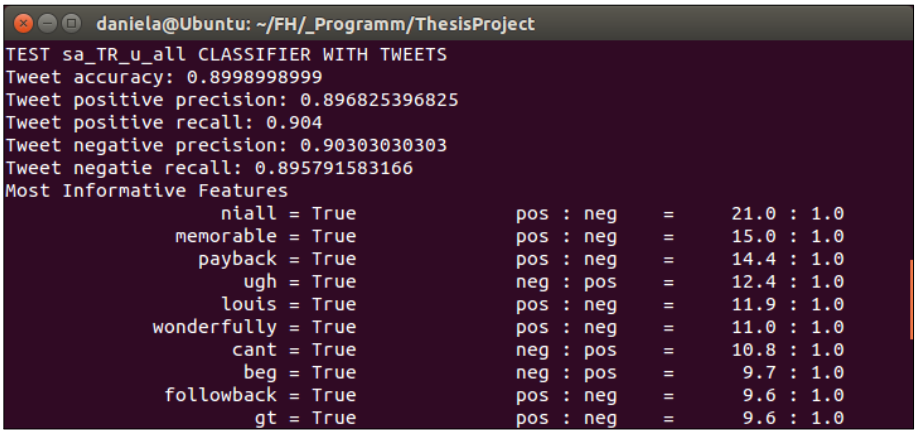

(a)



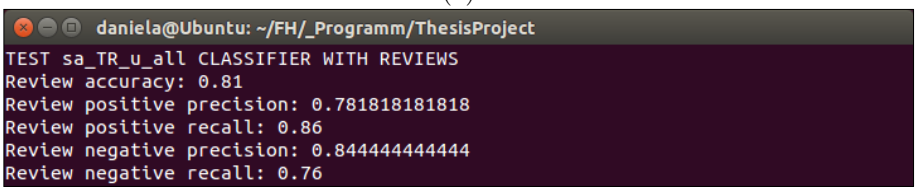

(b)

**Abbildung A.6:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Movie-Reviews unter Verwendung von allen Uni- und Bigrammen (a) und den 10,000 Uni und 200 Bigrammen mit den meisten Informationen (b).

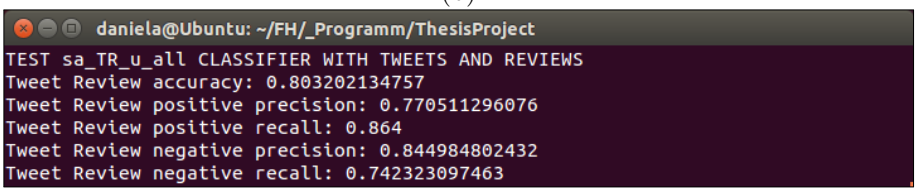
## A.3 Ergebnisse mit dem kombinierten Datenset




(a)

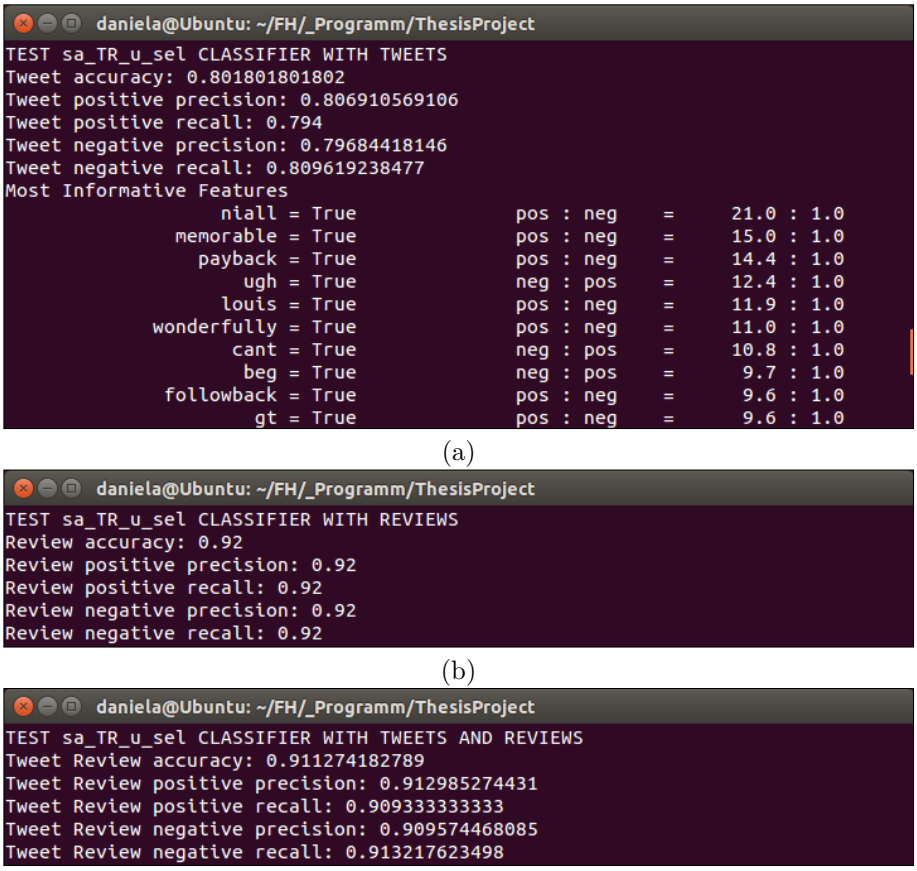



(b)




(c)

**Abbildung A.7:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets (a), Movie-Reviews (b) und beiden (c) unter Verwendung von allen Unigrammen.




(a)




(b)




(c)

**Abbildung A.8:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets (a), Movie-Reviews (b) und beiden (c) unter Verwendung von den 10,000 Unigrammen mit den meisten Informationen.

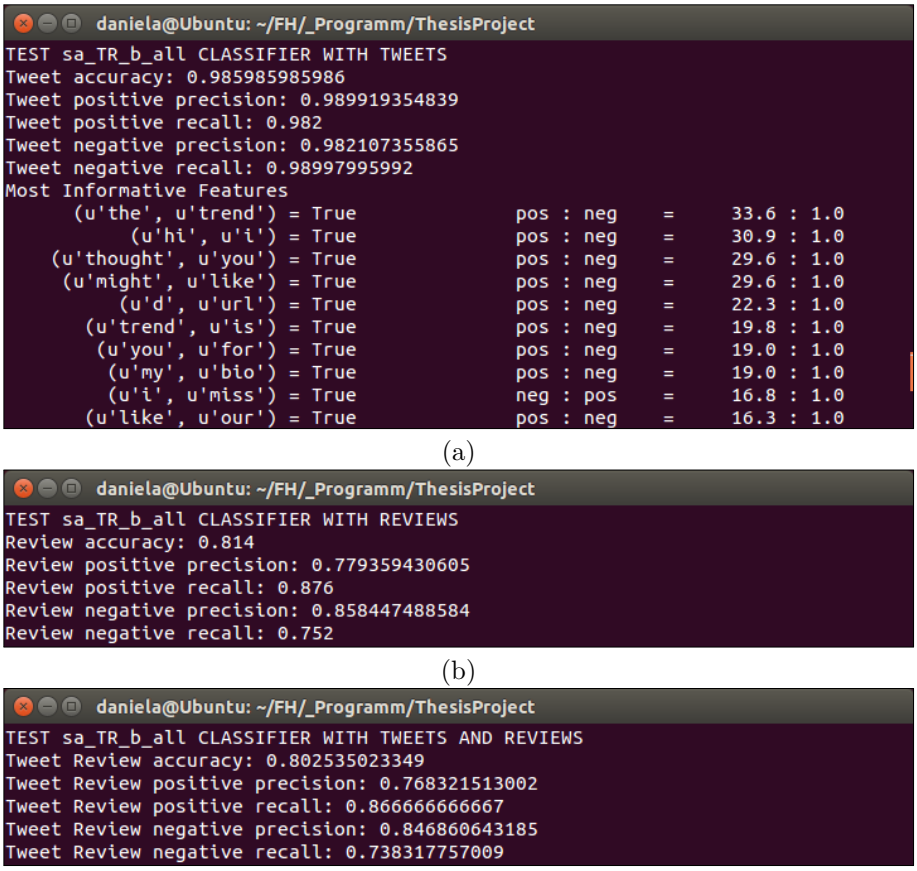
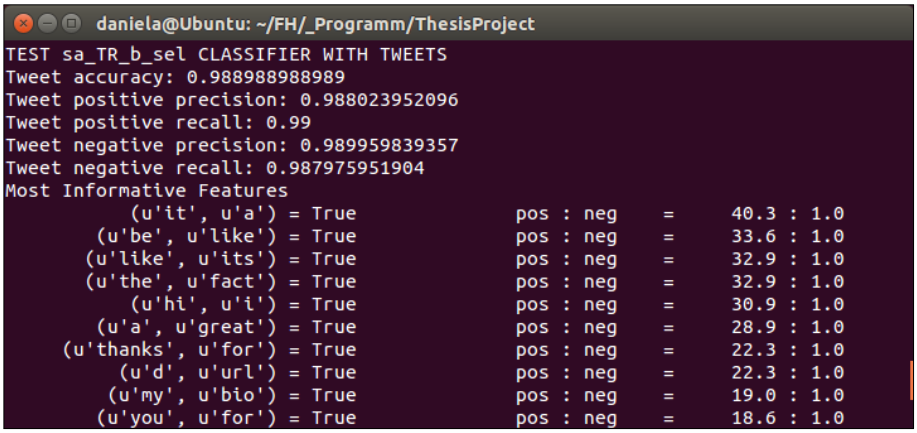
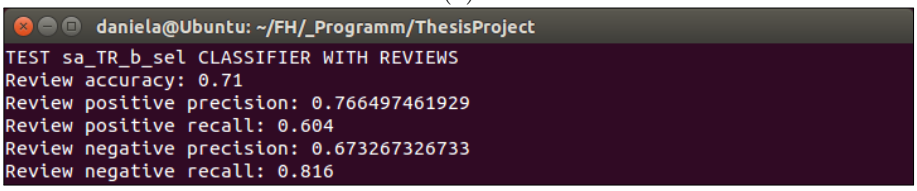


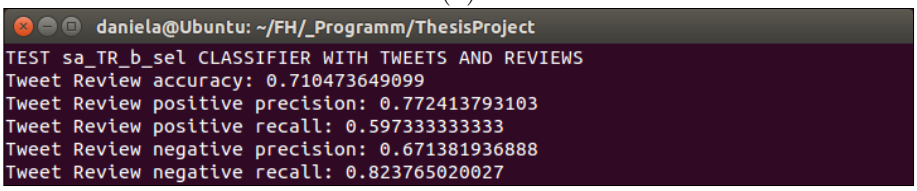

**Abbildung A.9:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets (a), Movie-Reviews (b) und beiden (c) unter Verwendung von allen Bigrammen.




(a)

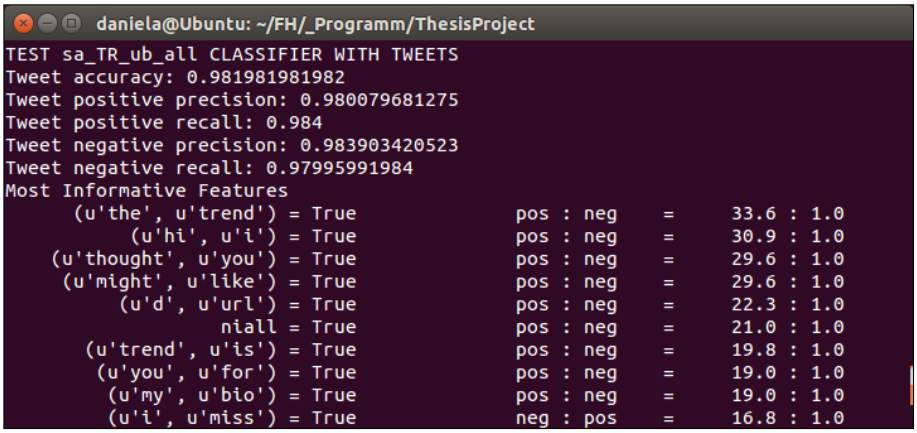



(b)

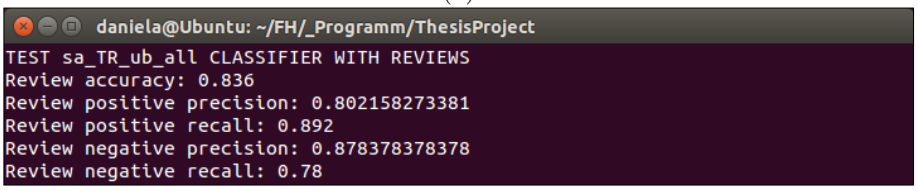



(c)

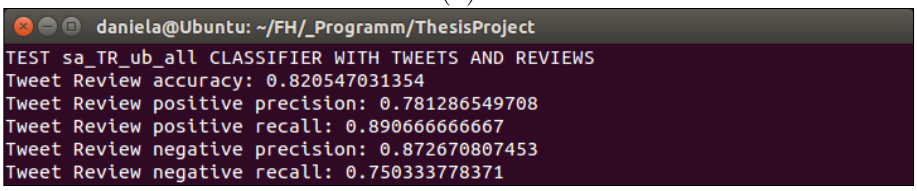
**Abbildung A.10:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets (a), Movie-Reviews (b) und beiden (c) unter Verwendung von den 200 Bigrammen mit den meisten Informationen.




(a)

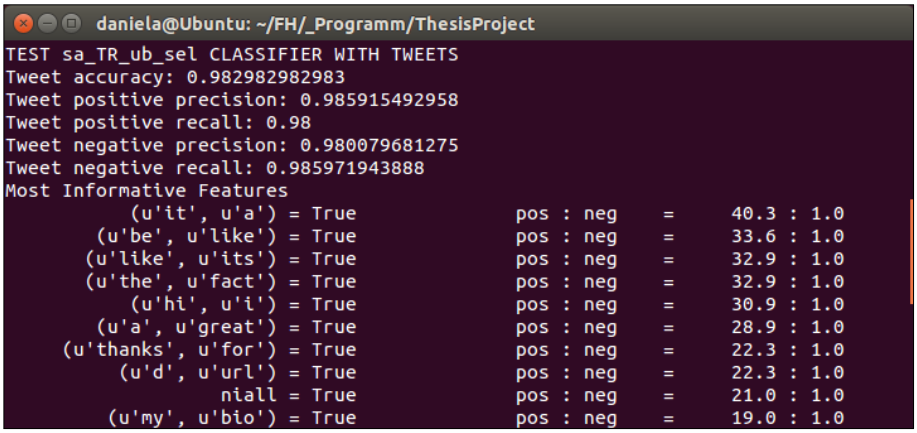



(b)

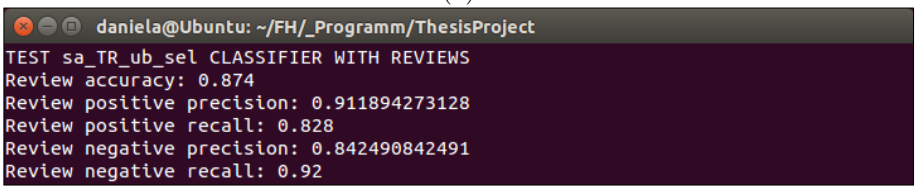



(c)

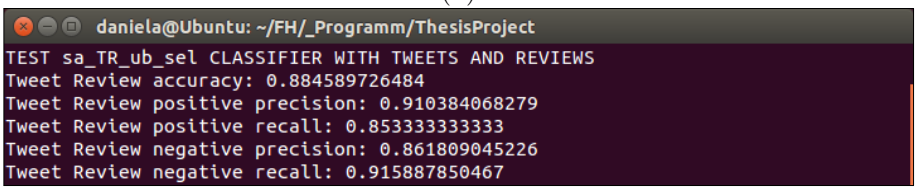
**Abbildung A.11:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets (a), Movie-Reviews (b) und beiden (c) unter Verwendung von allen Uni- und Bigrammen.




(a)



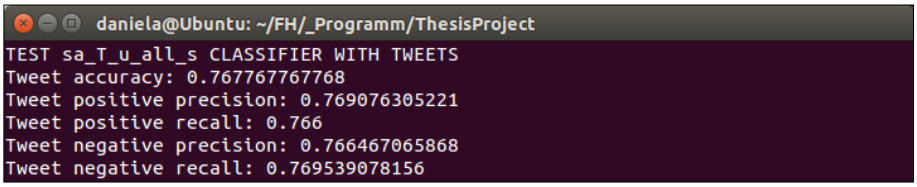

(b)



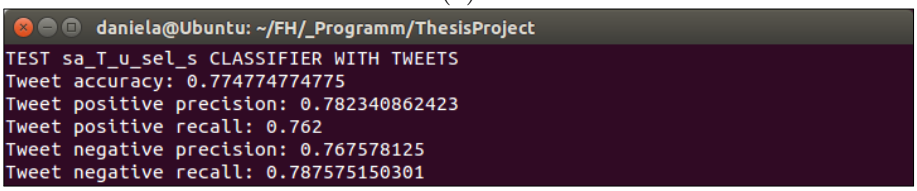

(c)

**Abbildung A.12:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets (a), Movie-Reviews (b) und beiden (c) unter Verwendung von den 10,000 Uni- und 200 Bigrammen mit den meisten Informationen.

## A.4 Ergebnisse mit dem Twitter- und dem kombinierten Datenset mit Stemming

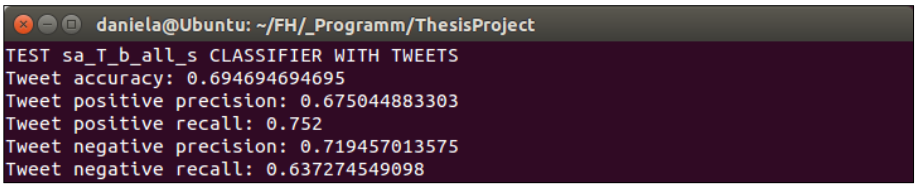



(a)

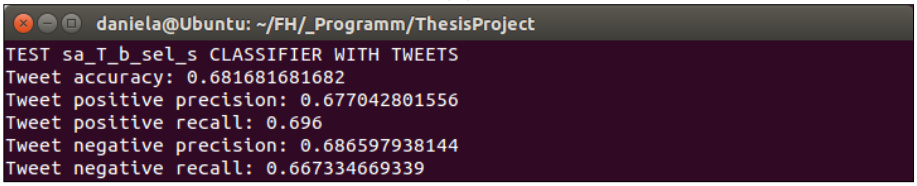



(b)

**Abbildung A.13:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Unigrammen (a) und den 10,000 Unigrammen mit den meisten Informationen (b) mit Stemming.

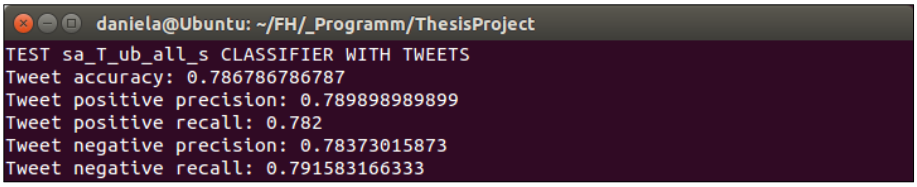



(a)

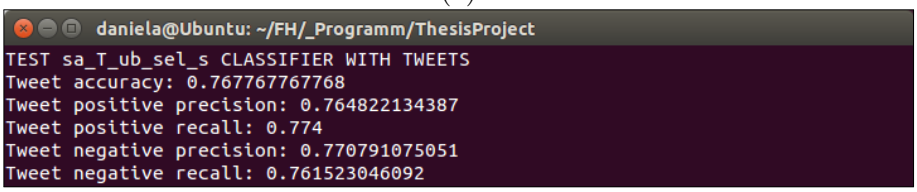



(b)

**Abbildung A.14:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Bigrammen (a) und den 200 Bigrammen mit den meisten Informationen (b) mit Stemming.

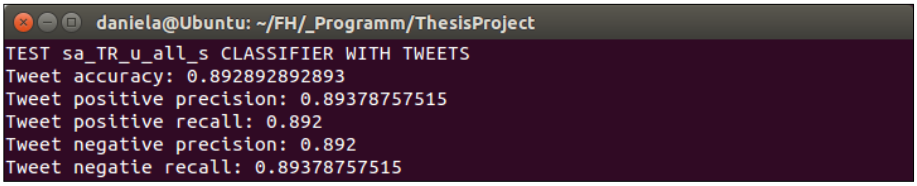



(a)

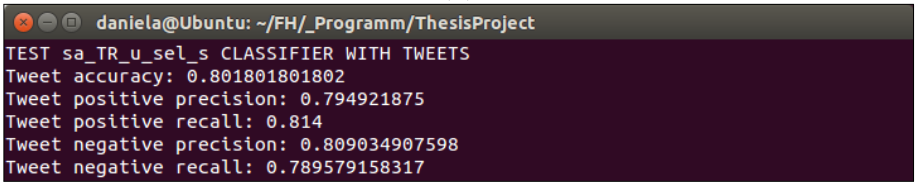



(b)

**Abbildung A.15:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Uni- und Bigrammen (a) und den 10,000 Uni- und 200 Bigrammen mit den meisten Informationen (b) mit Stemming.

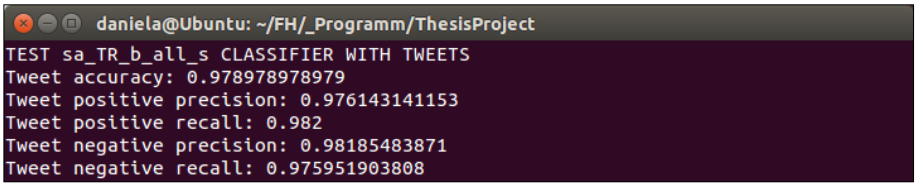



(a)

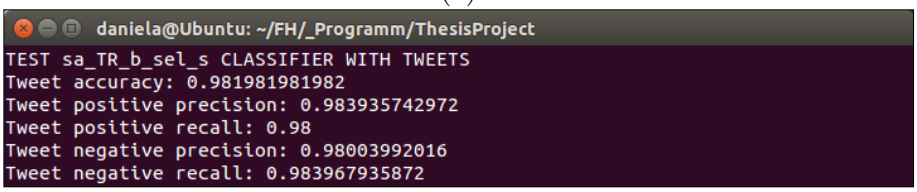



(b)

**Abbildung A.16:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets unter Verwendung von allen Unigrammen (a) und den 10,000 Unigrammen mit den meisten Informationen (b) mit Stemming.

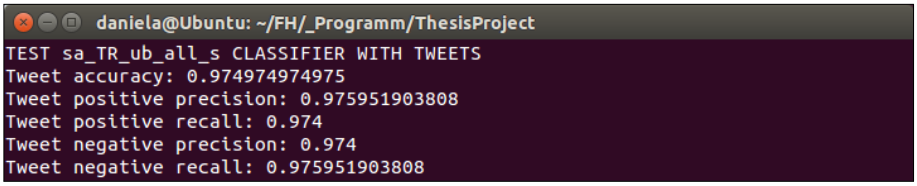



(a)

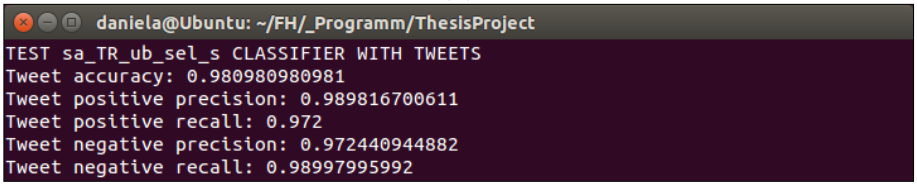



(b)

**Abbildung A.17:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets unter Verwendung von allen Bigrammen (a) und den 200 Bigrammen mit den meisten Informationen (b) mit Stemming.



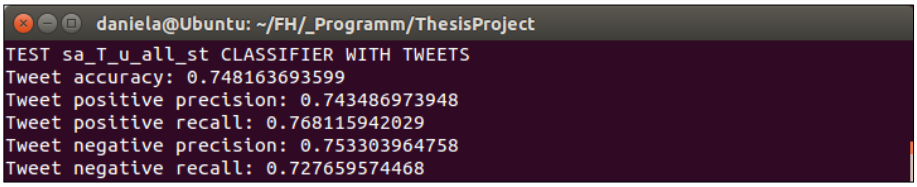

(a)



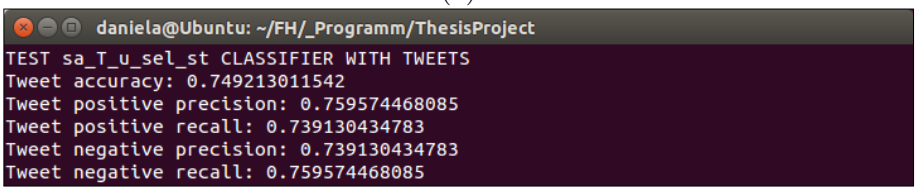

(b)

**Abbildung A.18:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets unter Verwendung von allen Uni- und Bigrammen (a) und den $10,000$ Uni- und 200 Bigrammen mit den meisten Informationen (b) mit Stemming.

## A.5 Ergebnisse mit dem Twitter- und kombinierten Datenset mit Stoppwort-Entfernung

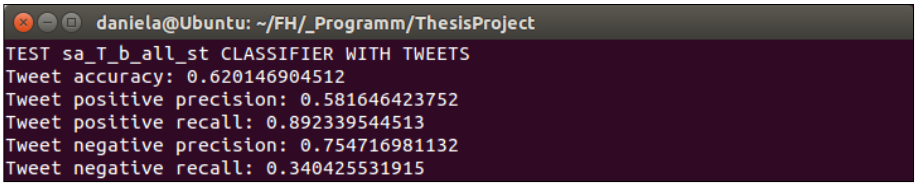



(a)

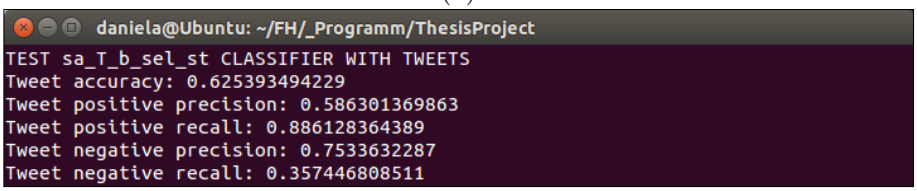



(b)

**Abbildung A.19:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Unigrammen (a) und den 10,000 Unigrammen mit den meisten Informationen (b) mit Stoppwort-Entfernung.

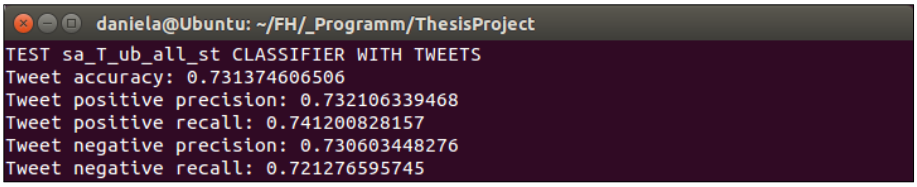



(a)

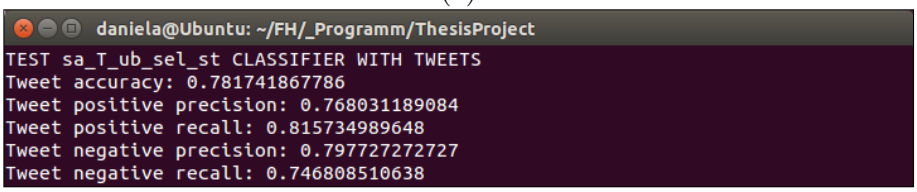



(b)

**Abbildung A.20:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Bigrammen (a) und den 200 Bigrammen mit den meisten Informationen (b) mit Stoppwort-Entfernung.

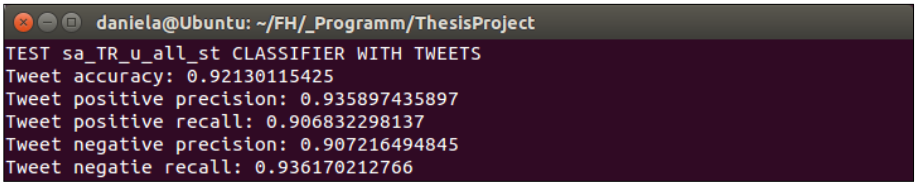



(a)

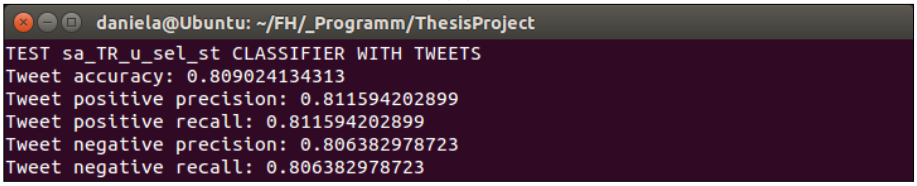



(b)

**Abbildung A.21:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Uni- und Bigrammen (a) und den 10,000 Uni- und 200 Bigrammen mit den meisten Informationen (b) mit Stoppwort-Entfernung.

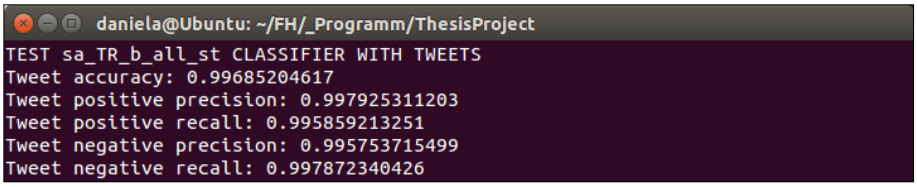



(a)

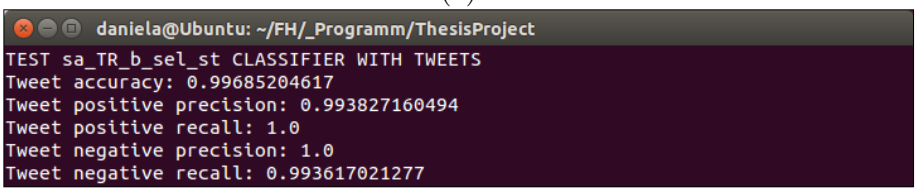



(b)

**Abbildung A.22:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets unter Verwendung von allen Unigrammen (a) und den 10,000 Unigrammen mit den meisten Informationen (b) mit Stoppwort-Entfernung.

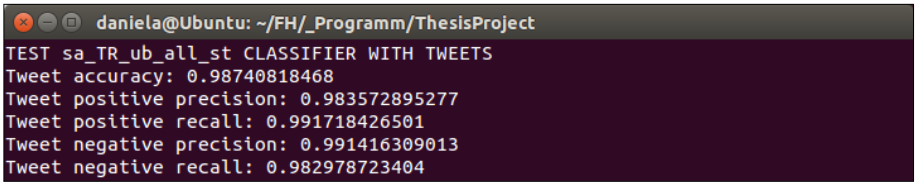



(a)

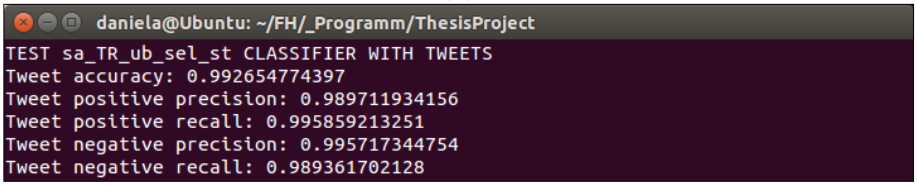



(b)

**Abbildung A.23:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets unter Verwendung von allen Bigrammen (a) und den 200 Bigrammen mit den meisten Informationen (b) mit Stoppwort-Entfernung.



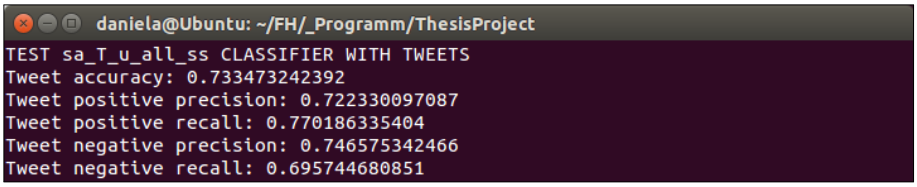

(a)



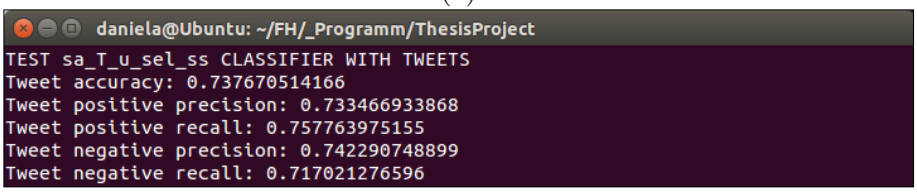

(b)

**Abbildung A.24:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets unter Verwendung von allen Uni- und Bigrammen (a) und den 10,000 Uni- und 200 Bigrammen mit den meisten Informationen (b) mit Stoppwort-Entfernung.

## A.6 Ergebnisse mit dem Twitter- und kombinierten Datenset mit Stemming und Stoppwort-Entfernung

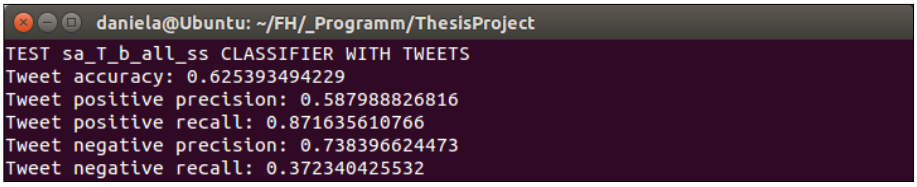



(a)

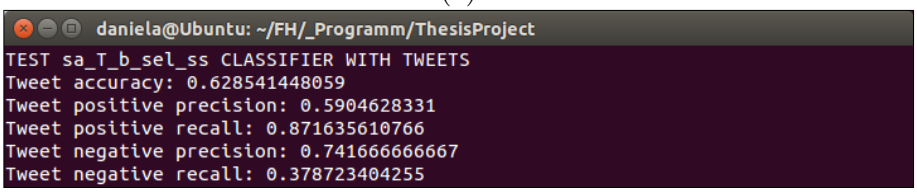



(b)

**Abbildung A.25:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Unigrammen (a) und den 10,000 Unigrammen mit den meisten Informationen (b) mit Stemming und Stoppwort-Entfernung.

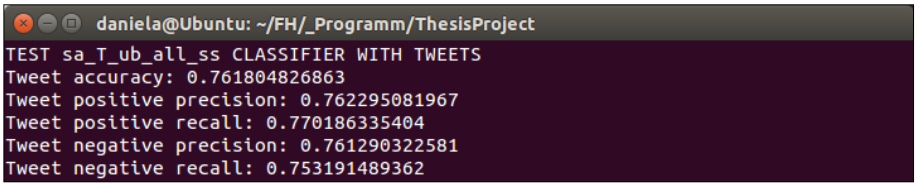



(a)

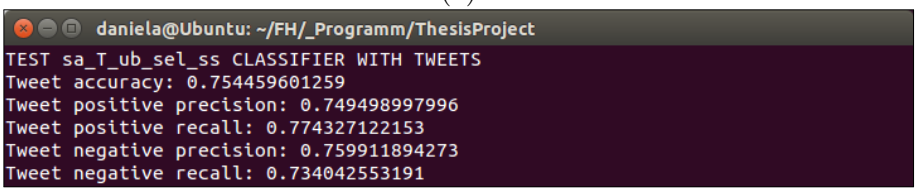



(b)

**Abbildung A.26:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Bigrammen (a) und den 200 Bigrammen mit den meisten Informationen (b) mit Stemming und Stoppwort-Entfernung.

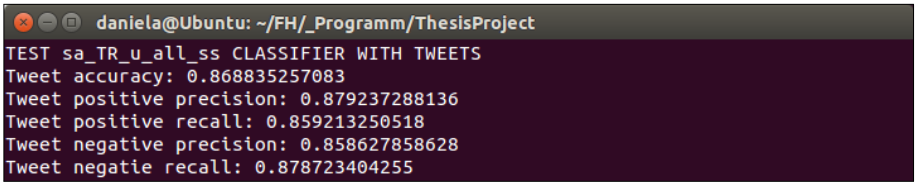



(a)

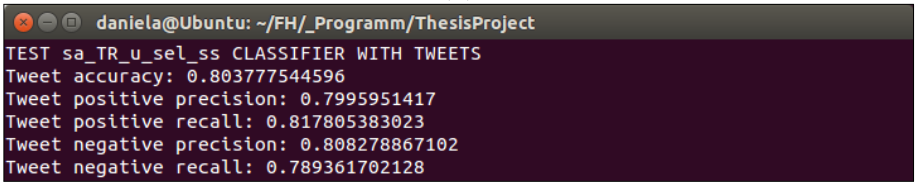



(b)

**Abbildung A.27:** Testergebnisse des Klassifikations-Modelles, trainiert und getestet mit Tweets unter Verwendung von allen Uni- und Bigrammen (a) und den 10,000 Uni- und 200 Bigrammen mit den meisten Informationen (b) mit Stemming und Stoppwort-Entfernung.

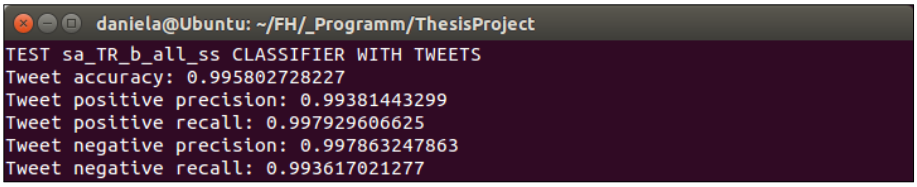



(a)

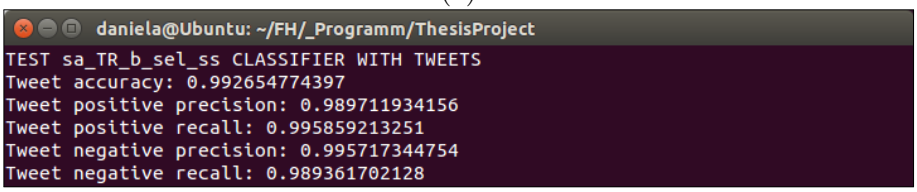



(b)

**Abbildung A.28:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets unter Verwendung von allen Unigrammen (a) und den 10,000 Unigrammen mit den meisten Informationen (b) mit Stemming und Stoppwort-Entfernung.

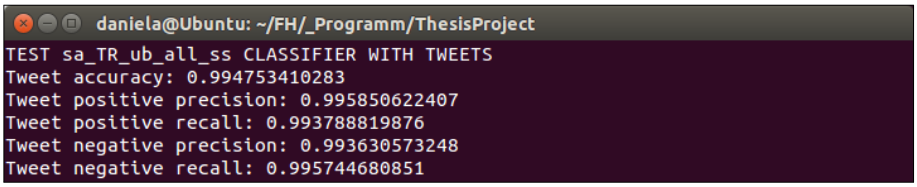



(a)

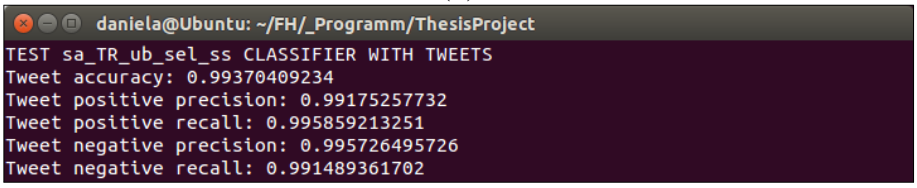



(b)

**Abbildung A.29:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets unter Verwendung von allen Bigrammen (a) und den 200 Bigrammen mit den meisten Informationen (b) mit Stemming und Stoppwort-Entfernung.




(a)




(b)

**Abbildung A.30:** Testergebnisse des Klassifikations-Modelles, trainiert mit dem kombinierten Datenset aus Tweets und Movie-Reviews und getestet mit Tweets unter Verwendung von allen Uni- und Bigrammen (a) und den 10,000 Uni- und 200 Bigrammen mit den meisten Informationen (b) mit Stemming und Stoppwort-Entfernung.

# Anhang B

# Inhalt der CD-ROM/DVD

## B.1 Struktur der CD

**Pfad:** /

| | |
|---|---|
| Bilder | Grafiken und Bilder der Arbeit |
| Latex | Ordner des LaTeX-Projekts |
| Quellen | Online Quellen und PDFs |
| Source | Implementierung des Projektes |
| ma_1310629017_Pillwein_Daniela.pdf | Masterarbeit mit Instruktionen (Gesamtdokument) |

## B.2 Bilder-Ordner

**Pfad:** /Bilder/

| | |
|---|---|
| test | Screenshots aller Tests |
| *.ai | Original Adobe Illustrator-Dateien aus denen die *.png-Dateien generiert wurden |
| dataset.png | Struktur der Twitter- und Movie-Review-Datensets innerhalb der Applikation |
| dbSystem.png | Diagramm mit Datenmodellen nach Strukturiertheit und Skalierbarkeit |
| klassifikationAblauf.png | Ablauf der Sentimentanalyse mit Stemming unter Verwendung der Uni- und Bigramme mit den meisten Informationen |
| tree-coin-general.png | Baumdiagramm für den Ablauf des Zufallsexperimentes (Wahl einer Münze und Werfen) |

## B.3 LaTeX-Ordner

**Pfad:** /Latex/

| | |
|---|---|
| _DaBa.tex | Masterarbeit (Hauptdokument) |
| vorwort.tex | Vorwort |
| kurzfassung.tex | Kurzfassung |
| abstract.tex | Abstract |
| ma_einleitung.tex | Kapitel 1 - Einleitung |
| ma_theorie.tex | Kapitel 2 - Theoretischer Hintergrund |
| ma_ansatz.tex | Kapitel 3 - Eigener Ansatz für die Verbesserung der Sentimentanalyse |
| ma_umsetzung.tex | Kapitel 4 - Technische Umsetzung des eigenen Ansatzes für die Verbesserung der Sentimentanalyse |
| ma_ergebnisse.tex | Kapitel 5 - Evaluierung |
| ma_diskussion.tex | Kapitel 6 - Zusammenfassende Diskussion |
| ma_zusammenfassung.tex | Kapitel 7 - Schlussbemerkungen |
| ma_anhang_a.tex | Screenshots der Testergebnisse |
| ma_anhang_b.tex | Inhalt der CD-ROM |
| messbox.tex | Messbox zur Druckkontrolle |
| literatur.bib | Literatur-Datenbank (BibTeX-Datei) |

## B.4 Quellen-Ordner

**Pfad:** /Quellen/

| | |
|---|---|
| *.pdf | Quellen als *.pdf-Dateien |
| *.HTML | Quellen als *.html-Dateien mit zugehörigem Ordner |

## B.5 Source-Ordner

**Pfad:** /Source/

| | |
|---|---|
| Libs | Ordner mit allen Abhängigkeiten und Bibliotheken |
| Sentimentanalyse | alle Sentimentanalyse Tests |
| TrainingData | Dateien zum Erstellen des Twitter-Datensets |
| Benutzerdokumentation.pdf | Infos zur Verwendung |
| Report.pdf | Projektbericht |
| Systemdokumentation.pdf | Infos zum System und der Implementierung |

### B.5.1 Libs-Ordner

**Pfad:** /Source/Libs/

| | |
|---|---|
| tweepy . . . . . | Bibliothek für die Verbindung zum Twitter Public Stream |
| erlang-solutions_1.0_all.deb | |
| riak_2.0.4-1_amd64.deb | Riak |

### B.5.2 Sentimentanalyse-Ordner

**Pfad:** /Source/Libs/

| | |
|---|---|
| sa_R_b_all.py . . . . . | Sentimentanalyse mit Movie-Reviews und allen Bigrammen |
| sa_R_b_sel.py . . . . . | Sentimentanalyse mit Movie-Reviews und den Bigrammen mit den meisten Infos |
| sa_R_u_all.py . . . . . | Sentimentanalyse mit Movie-Reviews und allen Unigrammen |
| sa_R_u_sel.py . . . . . | Sentimentanalyse mit Movie-Reviews und den Unigrammen mit den meisten Infos |
| sa_R_ub_all.py . . . . | Sentimentanalyse mit Movie-Reviews und allen Uni- und Bigrammen |
| sa_R_ub_sel.py . . . . | Sentimentanalyse mit Movie-Reviews und den Uni- und Bigrammen mit den meisten Infos |
| sa_T_b_all.py . . . . . | Sentimentanalyse mit Tweets und allen Bigrammen |
| sa_T_b_sel.py . . . . . | Sentimentanalyse mit Tweets und den Bigrammen mit den meisten Infos |
| sa_T_u_all.py . . . . . | Sentimentanalyse mit Tweets und allen Unigrammen |
| sa_T_u_sel.py . . . . . | Sentimentanalyse mit Tweets und den Unigrammen mit den meisten Infos |
| sa_T_ub_all.py . . . . | Sentimentanalyse mit Tweets und allen Uni- und Bigrammen |
| sa_T_ub_sel.py . . . . | Sentimentanalyse mit Tweets und den Uni- und Bigrammen mit den meisten Infos |
| sa_TR_b_all.py . . . . | Sentimentanalyse mit Tweets und Reviews und allen Bigrammen |
| sa_TR_b_sel.py . . . . | Sentimentanalyse mit Tweets und Reviews und den Bigrammen mit den meisten Infos |
| sa_TR_u_all.py . . . . | Sentimentanalyse mit Tweets und Reviews |

| | |
|---|---|
| | und allen Unigrammen |
| sa_TR_u_sel.py . . . . | Sentimentanalyse mit Tweets und Reviews und den Unigrammen mit den meisten Infos |
| sa_TR_ub_all.py . . . | Sentimentanalyse mit Tweets und Reviews und allen Uni- und Bigrammen |
| sa_TR_ub_sel.py . . . | Sentimentanalyse mit Tweets und Reviews und den Uni- und Bigrammen mit den meisten Infos |

### B.5.3 TrainingData-Ordner

**Pfad:** /Source/Libs/

| | |
|---|---|
| authentication.py . . . . | Authentifikation mit Twitter |
| initTrainingData.py . . | Starten des Authentifikations-Vorganges und des Listeners |
| Listener.py . . . . . | Verbindungsaufbau und Laden der Tweets vom Public Stream |

# Quellenverzeichnis

## Literatur


[1] Apoorv Agarwal u. a. „Sentiment Analysis of Twitter Data“. In: *Proceedings of the Workshop on Languages in Social Media*. (Portland, Oregon). LSM '11. Stroudsburg, PA, USA: Association for Computational Linguistics Stroudsburg, PA, USA, Juni 2011 (siehe S. 41).

[2] Charu C. Aggarwal und ChengXiang Zhai. *Mining Text Data*. New York, NY, USA: Springer Verlag, 2012 (siehe S. 22).

[3] Pavlo Baron. *Big Data für IT-Entscheider: Riesige Datenmengen und moderne Technologien gewinnbringend nutzen*. 1. Aufl. München: Carl Hanser Verlag, Apr. 2013 (siehe S. 40, 50).

[4] Adam L. Berger, Stephen Della Pietra und Vincent J. Della Pietra. „A Maximum Entropy Approach to Natural Language Processing“. In: *Computational Linguistics* 22.1 (März 1996), S. 39–71 (siehe S. 22, 23).

[5] Steven Bird, Ewan Klein und Edward Loper. *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit*. Sebastopol: O'Reilly Media, Inc., Juni 2009 (siehe S. 30, 39, 49).

[6] Julian Brooke und Graeme Hirst. „Native language detection with "cheap" learner corpora“. In: *Twenty Years of Learner Corpus Research: Looking Back, Moving Ahead. Proceedings of the First Learner Corpus Research Conference (LCR 2011)*. Hrsg. von S. Granger, G. Gilquin und F. Meunier. Paris, Frankreich: Presses universitaires de Louvain, 2013, S. 37–47 (siehe S. 26).

[7] Gert Cauwenberghs und Tomaso Poggio. „Incremental and Decremental Support Vector Machine Learning“. In: *Advances in Neural Information Processing Systems (NIPS*2000)*. Cambridge, MA, USA: MIT Press, 2001, S. 409–415 (siehe S. 23).

[8] Eleanor Clark und Kenji Araki. „Text Normalization in Social Media: Progress, Problems and Applications for a Pre-Processing System of Casual English“. In: *Procedia - Social and Behavioral Sciences* 27 (Dez. 2011), S. 2–11 (siehe S. 28).

[9] Sanjiv R. Das und Mike Y. Chen. „Yahoo! For Amazon: Sentiment Extraction from Small Talk on the Web“. In: *Management Science* 53.9 (Sep. 2007), S. 1375–1388 (siehe S. 29).

[11] George Forman. „An extensive empirical study of feature selection metrics for text classification“. In: *The Journal of Machine Learning Research* 3 (März 2003), S. 1289–1305 (siehe S. 26).

[12] Moises Goldszmidt, Marc Najork und Stelios Paparizos. „Boot-strapping Language Identifiers for Short Colloquial Postings“. In: *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2013)*. (Prag, Tschechische Republik). Heidelberg, Deutschland: Springer Verlag, Sep. 2013, S. 95–111 (siehe S. 45).

[13] Sylvaine Granger, Estelle Dagneaux und Fanny Meunier. *International Corpus of Learner English*. Paris, Frankreich: Presses universitaires de Louvain, 2002 (siehe S. 26).

[14] Apoorv Ingle u. a. „Sentiment Analysis: Sarcasm Detection of Tweets“. Bachelorarbeit. Nagpur, Maharashtra, Indien: Visvesvaraya National Institute of Technology, Department of Computer Science und Engineering, Mai 2014 (siehe S. 30, 38, 39, 50, 51).

[15] Youngjoong Ko und Jungyun Seo. „Automatic text categorization by unsupervised learning“. In: *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*. (Saarbrücken, Germany). COLING '00. Stroudsburg, PA, USA: Association for Computational Linguistics, Juli 2000, S. 453–459 (siehe S. 24, 25).

[10] Dave Kushal, Steve Lawrence und David M. Pennock. „Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews“. In: *Proceedings of the 12th International Conference on World Wide Web*. (Budapest, Hungary). WWW '03. New York, NY, USA: ACM, Mai 2003, S. 519–528 (siehe S. 27).

[16] David D. Lewis. „Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval“. In: *Proceedings of the 10th European Conference on Machine Learning*. (Chemnitz, Deutschland). ECML '98. London, UK: Springer Verlag, 1998, S. 4–15 (siehe S. 17).

[17] Bing Liu, Wynne Hsu und Yiming Ma. „Integrating Classification and Association Rule Mining“. In: *Proceedings of the 4th international conference on Knowledge Discovery and Data mining (KDD'98)*. New York, USA: AAAI Press, Aug. 1998, S. 80–86 (siehe S. 24).

[19] Marco Lui, Jey Han Lau und Timothy Baldwin. „Automatic Detection and Language Identification of Multilingual Documents“. In: *Transactions of the Association for Computational Linguistics* 2 (2014), S. 27–40 (siehe S. 26).

[20] Eric Mays, Fred J. Damerau und Robert L. Mercer. „Context Based Spelling Correction“. In: *Information Processing and Management: an International Journal* 27.5 (Sep. 1991), S. 517–522 (siehe S. 28).

[21] Sharon B. McGrayne. *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy*. New Haven, CT, USA: Yale University Press, Sep. 2012 (siehe S. 11, 16).

[22] Walaa Medhat, Ahmed Hassan und Hoda Korashy. „Sentiment analysis algorithms and applications: A survey“. In: *Ain Shams Engineering Journal* 5.4 (2014), S. 1093–1113 (siehe S. 16, 21–25).

[23] Donald Michie u. a. *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA: Ellis Horwood, Feb. 1994 (siehe S. 7, 17, 21).

[24] Tom M. Mitchell. *Machine Learning*. Columbus, OH, USA: WCB McGraw-Hill, März 1997 (siehe S. 7–10).

[44] Éva Mújdricza und Ganna Syrota. *Stemmingverfahren*. Techn. Ber. Heidelberg, Deutschland: Ruprecht-Karls-Universität Heidelberg, Feb. 2008. URL: http://www.cl.uni-heidelberg.de/~mujdricz/software/hunPort/IR_Referat_Stemming_MujdriczaSyrota.pdf (siehe S. 29).

[25] Kamal Nigam u. a. „Learning to Classify Text from Labeled and Unlabeled Documents“. In: *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. (Madison, Wisconsin, USA). AAAI '98/IAAI '98. Menlo Park, CA, USA: American Association for Artificial Intelligence, Juli 1998, S. 792–799 (siehe S. 24).

[26] Alexander Pak und Patrick Paroubek. „Twitter as a Corpus for Sentiment Analysis and Opinion Mining“. In: *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. (Valletta, Malta). Paris, France: European Languages Resources Association (ELRA), Mai 2010, S. 1320–1326 (siehe S. 39, 44, 45).

[27] Bo Pang und Lillian Lee. „Opinion Mining and Sentiment Analysis“. In: *Foundations and Trends in Information Retrieval* 2.1-2 (Jan. 2008), S. 1–135 (siehe S. 2, 6, 26, 27, 29, 60, 66).

[28] Bo Pang, Lillian Lee und Shivakumar Vaithyanathan. „Thumbs up?: sentiment classification using machine learning techniques“. In: *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA: Association for Computational Linguistics, Juli 2002, S. 79–86 (siehe S. 16, 22, 23, 26, 27).

[29] John A. Paulos. „The Mathematics of Changing Your Mind“. In: *The New York Times* (5. Aug. 2011). URL: http://www.nytimes.com/2011/08/07/books/review/the-theory-that-would-not-die-by-sharon-bertsch-mcgrayne-book-review.html?_r=0 (siehe S. 10).

[30] Beat Pfister und Tobias Kaufmann. *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Berlin, Heidelberg: Springer-Verlag, Juni 2008 (siehe S. 25).

[18] Marc Reznicek u. a. *Das Falko-Handbuch: Korpusaufbau und Annotationen*. Version 2.01. Sep. 2012. URL: https://www.linguistik.hu-berlin.de/de/institut/professuren/korpuslinguistik/forschung/falko/Falko-Handbuch_Korpusaufbau%20und%20Annotationen_v2.01 (siehe S. 26).

[31] Nicolas Ruflin, Helmar Burkhart und Sven Rizzotti. „Social-data Storage-systems“. In: *Databases and Social Networks*. (Athens, Greece). DBSocial '11. New York, NY, USA: ACM, Juni 2011, S. 7–12 (siehe S. 40, 41).

[46] Robert Schapire. *Theoretical Machine Learning*. Techn. Ber. Princeton, NJ, USA: Princeton University, Feb. 2008. URL: http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf (siehe S. 7).

[32] Richard Sproat u. a. „Normalization of Non-Standard Words“. In: *Computer Speech & Language* 15.3 (2001), S. 287–333 (siehe S. 27, 28).

[33] Martin Weigert. „Sentimentanalyse: Emotionen statt Fakten“. In: *Netzwertig* (Mai 2010). URL: http://netzwertig.com/2010/05/27/sentimentanalyse-emotionen-statt-fakten/ (siehe S. 6).

[34] Ian H. Witten und Frank Eibe. *Data Mining: Practical Machine Learning Tools and Techniques*. 2. Aufl. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005 (siehe S. 6).

[35] Helen Yannakoudakis, Ted Briscoe und Ben Medlock. „A New Dataset and Method for Automatically Grading ESOL Texts“. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. (Portland, Oregon). HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, Juni 2011, S. 180–189 (siehe S. 26).

## Online-Quellen

[40] Basho Technologies, Inc. *Riak*. 2015. URL: http://docs.basho.com/riak/latest/ (besucht am 18. 09. 2015) (siehe S. 50).

[41] Duden. URL: http://www.duden.de/ (besucht am 18. 09. 2015) (siehe S. 6, 7, 25, 26).

[43] GFB & Partner Marketing Services. *Wie funktioniert eine Sentimentanalyse*. 2015. URL: http://www.gfb-marketing.at/medien/factsheets/Social-Media-Monitoring.pdf (besucht am 18. 09. 2015) (siehe S. 5).

[36] Dan Jurafsky. *Introduction to N-grams*. URL: https://class.coursera.org/nlp/lecture/14 (besucht am 18. 09. 2015) (siehe S. 27).

[37] Dan Jurafsky. *Naïve Bayes: Learning*. URL: https://class.coursera.org/nlp/lecture/26 (besucht am 18. 09. 2015) (siehe S. 19).

[38] Dan Jurafsky. *The Noisy Channel Model of Spelling*. URL: https://class.coursera.org/nlp/lecture/22 (besucht am 18. 09. 2015) (siehe S. 28).

[39] Dan Jurafsky. *Word Normalization and Stemming*. URL: https://class.coursera.org/nlp/lecture/126 (besucht am 18. 09. 2015) (siehe S. 29).

[42] Semantria by Lexalytics. *Measurement Methodology - Accuracy, Recall and Precision*. 3. Mai 2013. URL: http://support.semantria.com/customer/portal/articles/973525-measurement-methodology---accuracy-recall-precision (besucht am 18. 09. 2015) (siehe S. 3, 67, 68).

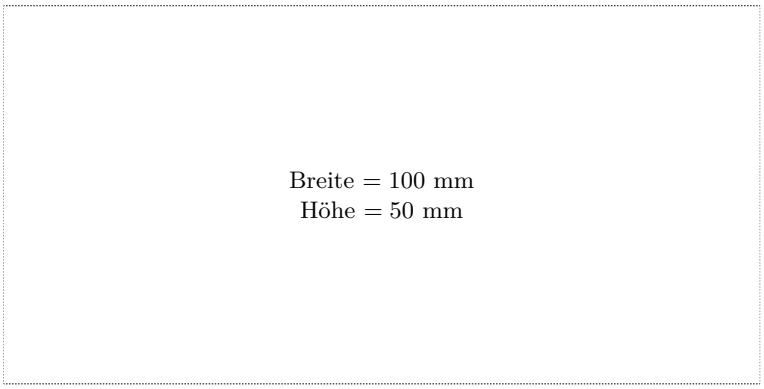
[45] Mathe Online. *Wahrscheinlichkeitsrechnung und Statistik*. Juni 2009. URL: http://www.mathe-online.at/mathint/wstat1/i.html#Wahrscheinlichkeit (besucht am 18. 09. 2015) (siehe S. 8–10).

[47] Wortwuchs. Juni 2015. URL: http://wortwuchs.net/stilmittel/sarkasmus/ (besucht am 18. 09. 2015) (siehe S. 30).

[48] Nina Zumel. *I don't think that means what you think it means; Statistics to English Translation, Part 1: Accuracy Measures*. Okt. 2009. URL: http://win-vector.com/dfiles/StatisticsToEnglishPart1_Accuracy.pdf (besucht am 18. 09. 2015) (siehe S. 3).

# Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —




— Diese Seite nach dem Druck entfernen! —