

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR STATISTIK



**Text Mining - Anwendung moderner Machine Learning Verfahren
zur Analyse von Kundenrezensionen**

BACHELORARBEIT
zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

Vorgelegt von:
Asmik Nalmpatian

Betreuung: Prof. Dr. Christian Heumann, Matthias Assenmacher
Datum : 9. Juli 2019

Abstract

In den vergangenen Jahren haben die Kundenrezensionen insbesondere durch das große Angebot im Internet an Bedeutung gewonnen. Da es jedoch aufwendig ist, einen Überblick über die Menge an Informationen zu behalten, ist eine Hilfestellung in Form von aufbereiteten, kategorisierten Daten sinnvoll. Als Grundlage der Analyse dieser Arbeit dienen die Kundenrezensionen des Onlineversandhändlers Amazon.com. Zur Klassifikation derer in positive bzw. negative Bewertungskategorien werden Machine Learning Verfahren, wie die logistische Regression mit Ridge- und Lasso-Penalisation, Naïve Bayes sowie Random Forests eingesetzt. Mit Hilfe von verschiedenen Methoden aus dem Bereich Natural Language Processing, wie Bag of Words, TF-IDF und Global Vectors Modellen, werden die Rezensionen numerisch repräsentiert. Die Prädiktionsgüte wird primär anhand des F1-Scores beurteilt. Gegenstand dieser Arbeit ist der Vergleich zwischen verschiedenen Techniken sowohl zur Merkmalsextrahierung, als auch zum Klassifikationsverfahren. Die Analysen werden zum größten Teil mithilfe des Programm Pakets Python und partiell mit R durchgeführt. Nach der Untersuchung aller Ergebnisse wird ein Vorteil für die logistische Regression mit beiden Penalisation Varianten gegenüber Naïve Bayes und Random Forests erkennbar sein. Wie es sich bei der Evaluation herausstellt, spielt das Gleichgewicht der Daten eine entscheidende Rolle beim Erlernen der Modelle. Des Weiteren zeigen die herkömmlichen Bag of Words basierten Modelle eine leicht höhere Prädiktionsgüte im Vergleich zu Global Vectors Modellen. Die Ergebnisse werden zudem durch die Kategorisierung von 3-Sterne-Bewertungen beeinflusst.

Inhaltsverzeichnis

1 Einleitung	3
1.1 Motivation	3
1.2 Problemstellung und Zielsetzung	3
1.3 Vorgehen	3
2 Datenbasis	4
3 Bildung von numerischen Dokumentrepräsentationen	7
3.1 Normalisierung	7
3.2 Bag of Words Modell	9
3.3 TF-IDF Modell	10
3.4 Global Vectors Modell	12
4 Textklassifikation: Machine Learning Verfahren	15
4.1 Logistische Regression	16
4.2 Naïve Bayes	19
4.3 Random Forests	22
4.4 Kreuzvalidierung	24
4.5 Auswertungskennzahlen	25
5 Anwendung	27
5.1 Implementierung	27
5.2 Diskussion der Ergebnisse	32
5.2.1 Umgang mit unbalancierten Daten	34
5.2.2 Kategorisierung von 3-Sterne-Bewertungen	38
6 Fazit und Ausblick	40
Literatur	44
Anhang	45
A Quellcode: Funktionen	45
B Wahl der Modellparameter	52
C Kreuzvalidierungs- und Generalisierungsfehler	56
Abbildungsverzeichnis	60
Tabellenverzeichnis	61
Eigenständigkeitserklärung	64

1 Einleitung

1.1 Motivation

Die zunehmende Auswahl im Netz erschwert den Entscheidungs- und Meinungsbildungsprozess des Käufers¹ und unterstreicht die Bedeutung der Online-Kundenrezensionen als Wegweiser und eine der wichtigsten Kaufhilfen. Laut einer Studie² von Bitkom e.V. (2017) nehmen Kundenbewertungen einen beachtlichen Stellenwert ein: etwa 65 % der Befragten nutzen Online-Kundenbewertungen als Entscheidungshilfe vor dem Kauf eines Produkts. Dabei steht oft die Glaubwürdigkeit der Meinungen und Erfahrungen von Menschen im Vordergrund, die sich mit dem Produkt, für das sie sich interessieren, in ihrem Alltag auseinandersetzt haben. Weniger Vertrauen in Werbeversprechen und auf Marketingtexten basierende Produktinformationen von Herstellern bewegen Online-Käufer oft dazu, die Kundenrezensionen zur Minimierung des vermeintlichen Risikos bei der Wahl des Produktes heranzuziehen. Neben Informationstransparenz und Orientierungsfunktion, die Käufer mit Hilfe von Kundenrezensionen erstreben, können diese auch für Unternehmen und Händler vorteilhaft sein. Nicht nur eine rasche und offensive Gegenwirkung auf negative Rezensionen, sondern auch Erkenntnisse über eigene Produkte können zu einem angepassten Umgang mit Kunden und einer Verbesserung vom Firmenimage führen.

1.2 Problemstellung und Zielsetzung

Das Sichten, Zusammentragen und Extrahieren der Informationen, und besonders deren Analyse nimmt manuell betrieben viel Zeit und gegebenenfalls Personalressourcen in Anspruch. Da es sich dabei oft um eine große Auswahl an vergleichbaren Produkten und eine jeweils große Anzahl an Kundenrezensionen in Form von teilweise unstrukturierten Texten handelt. Um diesen Prozess zu automatisieren, ist es hilfreich die Kundenrezensionen auf eine geeignete Weise, wie beispielsweise Emotionen oder Zahlen, abzukürzen, so dass die in der jeweiligen Rezension abgebildete Meinung wiedergegeben wird. Dadurch können sowohl potenzielle Kunden, als auch Händler den zeitlichen Aufwand deutlich reduzieren, indem sie sich zum Beispiel nur auf die prozentuale Häufigkeit der vorkommenden positiven Rezensionen oder ihre Analyse auf die für sie möglicherweise im Fokus stehenden negativen Meinungsäußerungen beschränken. Viele elektronische Marktplätze haben bereits numerische Bewertungsschemata, zum Beispiel ein 1 bis 5 Sterne-System, welches einen kompakteren Überblick über die vorherrschende Meinung zu einem Produkt ermöglicht, als der Durchgang des Inhalts von einzelnen Rezensionen. Jedoch ist dies nicht bei allen Plattformen gegeben, weshalb es von essentieller Bedeutung ist, den rohen textuellen Kundenrezensionen eine Bewertung automatisch zu vergeben, um somit den generellen Eindruck mit möglichst wenig Zeitaufwand als Beurteilungs- und Entscheidungsgrundlage nutzen zu können.

1.3 Vorgehen

Gegenstand dieser Arbeit ist nun der Vergleich verschiedener Methoden zur Vorhersage einer negativen oder positiven Bewertungskategorie auf Basis von Kundenrezensio-

¹Wenn hier die Begriffe „Kunde“ oder „Nutzer“ gebraucht werden, dient dies lediglich dem besseren Lesefluss.

²Es handelt sich um eine durch den Digitalverband Bitkom beauftragte und Bitkom Research durchgeführte repräsentative Umfrage. Dabei wurden 1.114 Online-Käufer ab 14 Jahren befragt.

nen. Hierzu werden zunächst die verwendeten Daten, Verfahren zur numerischen Repräsentation von Texten sowie ausgewählte Machine Learning Algorithmen vorgestellt. Anschließend folgt die Entwicklung und Anwendung der Klassifikationskonzepte, wobei in diesem Zusammenhang auch praktische Herausforderungen Erwähnung finden. Im Hauptteil werden dann die Ergebnisse der Analyse evaluiert, bevor die Anwendungsmethodik im Fazit einer finalen Beurteilung unterzogen wird, und ein Ausblick auf weitere Methoden die Arbeit beschließt.

2 Datenbasis

Der dieser Arbeit zugrundeliegende Datensatz enthält Kundenrezensionen zu diversen Produkten auf Amazon.com, die Plattform eines US-amerikanischen Onlineversandhändlers ist. McAuley, Julian (2015), ein Dozent an der Universität von Kalifornien, San Diego, hat diesen frei zugänglichen Datensatz veröffentlicht, der 142.8 Millionen Amazon-Kundenrezensionen in englischer Sprache vom Zeitraum zwischen Mai 1996 und Juli 2014 beinhaltet. Außerdem werden nach Warenarten kategorisierte kleinere Teildatensätze zur Verfügung gestellt, wobei zu jedem Artikel und von jedem Nutzer jeweils mindestens fünf Rezensionen enthalten sind. Für diese Arbeit wurden zwei dieser Teildatensätze verwendet: Während der Erste 231.780 zur Warenkategorie Videospiele gehörende Rezensionen enthält, ist der Zweite mit 198.502 Rezensionen in der Warenkategorie Kosmetik eingegliedert. Diese werden nachfolgend mit den Kurzbezeichnungen *Video Games* und *Beauty* umschrieben.

Variablenname	Beschreibung
reviewerID	Nutzerkennzahl
asin	Produktkennzahl
reviewerName	Benutzername
helpful	Nützlichkeit der Bewertung: Anteil Nutzer, die die Bewertung als hilfreich bezeichnet haben
reviewText	Rezension in Textform
overall	Bewertung: 1- bis 5-Sterne
summary	Zusammenfassung der Rezension in Textform
unixReviewTime	Veröffentlichungszeitpunkt in Unixzeit: Anzahl vergangener Sekunden seit Donnerstag, dem 1. Januar 1970, 00:00 Uhr UTC.
reviewTime	Veröffentlichungszeitpunkt der Rezension formatiert als Monat, Tag, Jahr

Tabelle 1: Übersicht der Variablen in den verwendeten Datensätzen

Tabelle 1 verschafft einen Überblick über die in beiden Datensätzen vorhandenen Variablen. Ein Beispiel aus dem Datensatz Video Games ist die folgende Beobachtung:

```
{
  "reviewerID": "A1XUKDQJYFFJZV",
  "asin": "B00000DMB3",
  "reviewerName": "lucas",
  "helpful": [0:3],
  "reviewText": "This game is fun, but it is very overrated. It has not aged very well, and the game feels sloppy, compared to how revolutionary it was when it came out.",
  "overall": 3.0,
  "summary": "Overrated",
  "unixReviewTime": 1389139200,
  "reviewTime": "01, 8, 2014"
}
```

Wie dieselbe Rezension auf Amazon.com in origineller Form aussieht, kann der Abbildung 1 entnommen werden.

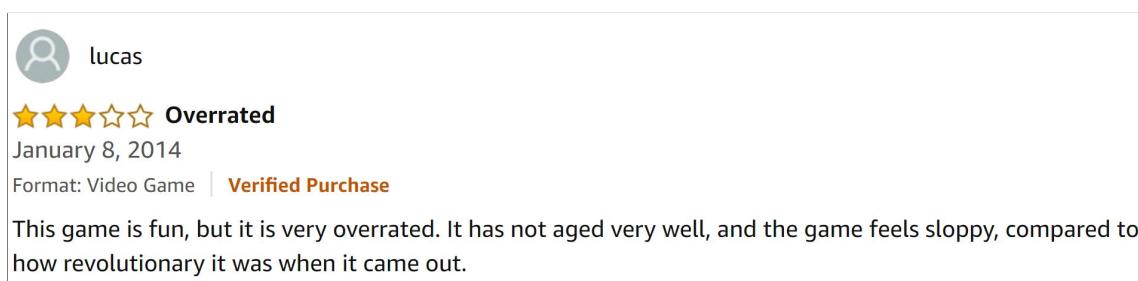


Abbildung 1: Beispiel für eine im Datensatz Video Games enthaltene Kundenrezension auf Amazon.com (2014)

Relevant für diese Arbeit sind zwei Variablen: `reviewText` und `overall`, die im Folgenden als *Rezensionstext* und *Bewertungssterne* bezeichnet werden. Die Bezeichnungen der beiden Warenkategorien kündigen bereits den Kontrast des verwendeten Vokabulars in den Rezensionstexten an. Während in Video Games oft die Rede von verschiedensten Spielen und deren Attributen ist, begegnet man in Beauty natürlich einem Kontext, der mit drogerietypischen Begriffen assoziiert ist. Auch in Abbildung 2 kommt dieser Unterschied zum Ausdruck. Hierbei werden die jeweils 20 häufigsten Wörter³ in Betracht gezogen, die in den Rezensionen der beiden Datensätze vorkommen: links Video Games, rechts Beauty, die Schriftgröße steht dabei proportional zur Häufigkeit des Wortes. Die durchschnittliche Anzahl der Wörter pro Rezension beträgt in Video Games ca. 210, in Beauty ca. 90. Bei den vom Rezensionstext abhängigen Bewertungssternen handelt es sich um insgesamt fünf Sterne, wobei 5 für die höchste und 1 für die niedrigste Bewertung steht. Die in Abbildung 3 visualisierten Häufigkeitsverteilungen der Bewertungssterne sind in beiden Datensätzen sehr ähnlich. Auffällig ist dabei die überwiegende Mehrheit der Rezensionen, die mit 5 Sternen versehen sind. Die mit diesem Ungleichgewicht verbundene Problematik und der technische Umgang werden im Abschnitt 5.2.1 detailliert thematisiert. Jedoch ist die Untersuchung potentiell dahinterstehender Gründe kein Bestandteil dieser Arbeit.

³Die Stoppwörter wurden dabei ausgenommen (vgl. Abschnitt 3.1).



Abbildung 2: Wordclouds: Die häufigsten Wörter in den Kundenrezensionen des jeweiligen Datensatzes

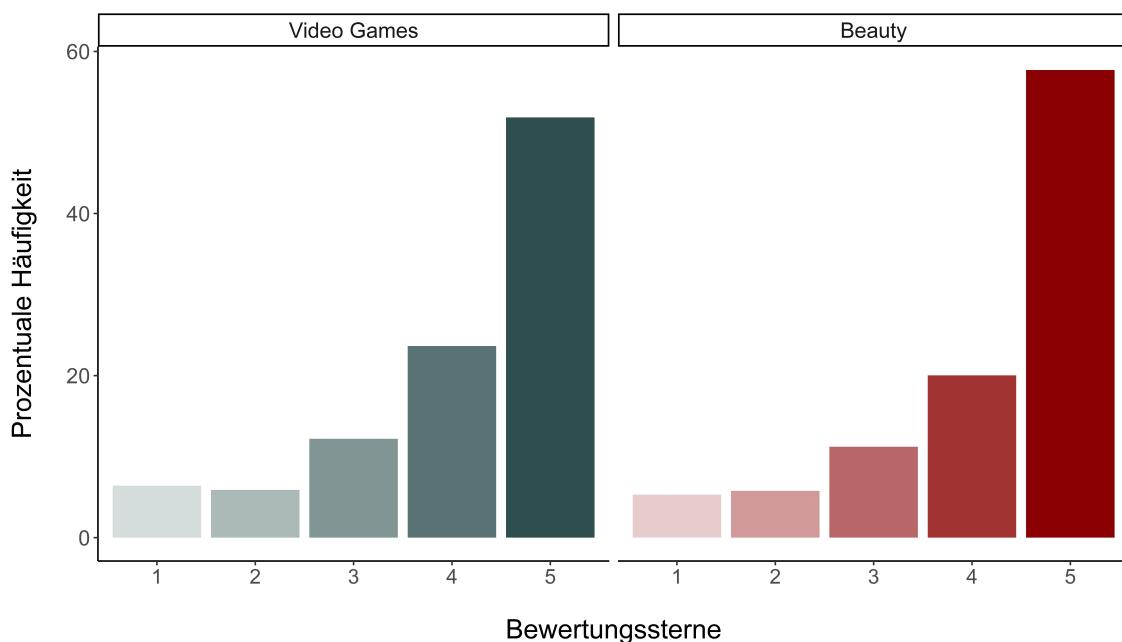


Abbildung 3: Prozentuale Häufigkeitsverteilung der Bewertungssterne

Wie in der Einleitung bereits erwähnt, ist das übergreifende Ziel dieser Arbeit, die Rezensionstexte in zwei Bewertungskategorien zu unterteilen: *negativ* und *positiv*. Daher werden die Bewertungssterne in eine binär kodierte Variable umgewandelt. Bei der Anwendung in Kapitel 5 wird näher auf die Datenaufbereitung eingegangen.

Zusammenfassend lässt sich also sagen, dass Video Games und Beauty eine ähnliche Häufigkeitsstruktur bezüglich der Bewertungssterne besitzen, wohingegen die Rezensionstexte sich hinsichtlich ihres Vokabulars stärker unterscheiden. Somit spielt Video Games für die weitere Analyse eine primäre Rolle und fungiert als Referenz-Datensatz, während Beauty die Untersuchung der Übertragbarkeit von den angewandten Auswer-

tungsmethodiken auf einen neuen Datensatz ermöglicht.

3 Bildung von numerischen Dokumentrepräsentationen

Es wird zum Ziel gesetzt, eine Kundenrezension aufgrund des gegebenen Rezensionstextes hinsichtlich der Kategorien als *positiv* oder *negativ* zu klassifizieren. Für die Prognose der Bewertungskategorien werden verschiedene Machine Learning Algorithmen angewendet, deren Funktionsweise in Kapitel 4 aufgeführt wird. Normalerweise arbeiten diese mit Einflussvariablen als Input, welche numerischer Natur sind. Textuelle Daten, wie die vorliegenden Rezensionstexte, können also nicht in deren Ursprungsform angewendet werden und müssen daher numerisch repräsentiert werden. Dazu stehen verschiedene Methoden und Techniken aus dem Bereich der Verarbeitung natürlicher Sprache bzw. *Natural Language Processing* (kurz: *NLP*) zur Verfügung. Die für diese Arbeit Relevanten werden in diesem Kapitel detailliert erläutert.

Bevor es aber zur eigentlichen Merkmalsextrahierung, dem sogenannten *feature extraction* kommt, müssen die Textdokumente - in diesem Fall die Rezensionstexte - normalisiert und aufbereitet werden, da diese anfangs oft sehr unstrukturiert sind. Nachfolgend werden die klassischen Schritte zur prototypischen Umsetzung erläutert, mit deren Hilfe die rohen Textdokumente in eine wohldefinierte, einheitlich standardisierte Form gebracht werden, um eine solide Basis für die darauffolgenden Analysephasen zu gewährleisten. Zu erwähnen ist, dass in diesem Kapitel die theoretischen Grundlagen eingeführt werden, während die genaue Implementierung im Abschnitt 5.1 dargelegt wird.

3.1 Normalisierung

Da die Aufbereitung der textuellen Daten einen wichtigen Part der Textanalyse darstellt, sollte diese keinesfalls unterschätzt werden. Insbesondere kann eine nicht angemessene Textaufbereitung zu ungewünschten oder irrelevanten Ergebnissen führen (Miner et al., 2012). Die Teilschritte werden hauptsächlich mithilfe des folgenden Satzbeispiels (Teil einer Amazon-Rezension) veranschaulicht, und die jeweils resultierenden Änderungen aufgezeigt: *love this gadget!! bought this to replace my non-electric model*. Da die vorliegenden Rezensionstexte in englischer Sprache geschrieben sind, werden auch sonstige Beispiele aus der englischen Grammatik aufgegriffen.

Tokenisierung: Bei der Tokenisierung handelt es sich um die Segmentierung von Texten. Diese werden dabei in vordefinierte Elemente bzw. *Tokens* aufgespalten, oftmals sind es Sätze oder Wörter. Der oben genannte Beispielsatz wird dann folgendermaßen zerlegt: *'love'*, *'this'*, *'gadget'*, *'!'*, *'!'*, *'bought'*, *'this'*, *'to'*, *'replace'*, *'my'*, *'non-electric'*, *'model'*, *'.'*. Die Tokenisierung in Wörter findet Anwendung in der späteren Auswertung und ist für die Normalisierungsprozesse relevant, die jedes Wort individuell behandeln.

Bereinigung: Zunächst sollen die Texte hinsichtlich der HTML-Formatierung bearbeitet und somit für weitere Schritte verständlicher dargestellt werden. Dies umfasst zum einen die Umkodierung von HTML-Entitäten, wodurch beispielsweise *&* zum speziellen Zeichen & wird (Meyer, Jens, 2002 - 2007). Zum anderen fällt die Entfernung von im Rahmen der vorliegenden Arbeit nutzlosen HTML-Zeichen darunter, wie zum Beispiel *< p >*, welche häufig am Satzanfang stehen, jedoch für die Textanalyse von keiner Bedeutung sind.

Ausschreibung von Abkürzungen: Anschließend sollen gängige Abkürzungen ausgeschrieben werden. Diese sind im Allgemeinen Kurzformen von Wörtern oder Silben

in der gesprochenen und geschriebenen Sprache. Insbesondere in der englischen Sprache finden die Kurzformen oft Verwendung unter Weglassung von einem der Vokale, beispielsweise wird *are not* mit *aren't* oder *you have* mit *you've* abgekürzt. Solche Abkürzungen werden in der formellen Sprache üblicherweise vermieden. Da es sich bei den vorliegenden Rezensionstexten nicht zwangsläufig um sachliche Texte handelt, gewinnt die Behandlung von Abkürzungen an Bedeutung. Problematisch wäre es, diese Kurzformen unbearbeitet in die weiteren Auswertungsschritte hineinzunehmen, unter anderem da die Tokenisierung des Apostrophs zu verzerrten Ergebnissen führen kann. Bei der Implementierung wird üblicherweise eine Liste mit gängigen Kurzformen und den zugehörigen Ausschreibungen erstellt (Sarkar, 2016). Beim Vorkommen von Abkürzungen in den Rezensionstexten werden diese entsprechend ausgeschrieben. Für diese Arbeit wurde das von Sarkar (2016) erstellte Abkürzungsvokabular verwendet.

Lemmatisierung: Nachdem einzelne Wörter in unterschiedlichen Zeit- und Zählformen vorkommen - wie beispielsweise *go*, *going*, *goes* - sollten diese für bestimmte Modelle, insbesondere das Bag of Words Modell oder das TF-IDF Modell (vgl. Abschnitt 3.2 und 3.3), in eine einheitliche Form gebracht werden. Das Ziel ist es, die Indexeinträge zu verringern. Bei der Lemmatisierung geht es insbesondere darum, Verben auf ihre Grundform und Nomen auf ihren Singular zu reduzieren (Manning et al., 2010). Es werden also zusätzlich die Informationen über die Wortart benötigt, welche mithilfe der von Sarkar (2016) detailliert beschriebenen *Parts of Speech Tags* gewonnen werden können. Dabei muss die resultierende Wortwurzel bzw. *das Lemma* im Wörterbuch existieren. Im Unterschied dazu wird beim Stemming die Endung des Wortes abgeschnitten, um dieses auf seinen Stamm zu bringen, ohne dabei seine lexikographische Korrektheit zu berücksichtigen. Während das Verb *studies* (in der dritten Person Singular) durch Lemmatisierung unter Berücksichtigung morphologischer Information auf *study* reduziert wird, bleibt beim Stemming das Wort *studi* übrig, welches in der englischen Sprache nicht existiert. In der weiteren Analyse kommt die Lemmatisierung zum Einsatz, die beim obigen Beispielsatz das Verb *bought* vom Perfekt zum Präsens bringen würde: *love this gadget!! buy this to replace my non-electric model.*

Entfernung von Sonderzeichen: Als nächstes sollen die Sonderzeichen entfernt werden, da diese für die Textanalyse meistens keine wichtige Rolle spielen (Sarkar, 2016). Unter Sonderzeichen sind gemäß Sarkar (2016) folgende Zeichen zu verstehen: ! “# \$% & ‘() * + , - . / : ; <= >? @ [] ~ _ { | } . Es ist sinnvoll, diesen Schritt nach der Ausschreibung von Abkürzungen einzusetzen, damit die Apostrophe nicht entfernt werden.

Rechtschreibkorrektur: Ein weiteres Problem stellen die Rechtschreibfehler dar, was beim Schreiben einer Rezension durchaus passieren kann. Aus einer Reihe von verschiedenen Algorithmen zur Korrektur solcher Fehler wird hier auf einen der bekanntesten Algorithmen eingegangen, der von Norvig, Peter (2007) entwickelt wurde. Dabei geht es grundsätzlich darum, gegeben einem Wort, eine Menge von kandidierenden Wörtern zu erstellen, die potenziell die richtige Schreibweise darstellen. Darauf basierend wird das ähnlichste Wort aus einem vordefinierten Wörterbuch gefunden, welches dann die korrekte Form des gegebenen Wortes sein soll. Dabei kommen verschiedene Ähnlichkeits- oder Distanzmaße in Frage. Diese Herangehensweise liefert jedoch nicht hundertprozentig korrekte Ergebnisse, da es davon abhängt, welche Wörter im Wörterbuch enthalten sind. Somit können varierende Verbesserungen resultieren. Außerdem ist dieser Schritt sehr zeitaufwendig, da jedes einzelne Wort einer relativ langen Prozedur bedarf. Um die späteren Auswertungen mit vertretbarem Zeitaufwand durchführen zu können, wird auf diesen Schritt, trotz der optionalen Implementierung, verzichtet (vgl. Funktion 1 im An-

hang A).

Entfernung von Stoppwörtern: Anschließend sollen die Rezensionen hinsichtlich der sogenannten *Stoppwörter* vorbereitet werden. Diese sind in der Regel sehr häufig auftretende Wörter, die meistens keine Relevanz für die Erfassung des Textinhalts besitzen und daher nicht betrachtet werden sollen. Beispiele für solche Wörter in der englischen Sprache sind: *me, and, do*. Die Entfernung geschieht anhand von einer vordefinierten Liste mit Stoppwörtern, die beliebig ergänzt werden kann (Sarkar, 2016). Der oben eingeführte Beispielsatz würde nach der Entfernung von Stoppwörtern folgendermaßen aussehen: *love gadget ! ! bought replace non-electric model*.

Letztlich wird der Text ausschließlich auf Buchstaben reduziert. Sodass Satzteile, die beispielsweise nur aus Zahlen bestehen, außer Acht bleiben, da durch diese meistens keine charakteristische Bedeutung gewonnen werden kann (Sarkar, 2016).

3.2 Bag of Words Modell

Wie bereits erwähnt, können gängige Machine Learning Algorithmen nicht auf Textdokumente in deren Ursprungsform angewendet werden, deshalb müssen die im vorherigen Abschnitt detaillierte Art und Weise normalisierten Rezensionstexte als Dokumentvektoren dargestellt werden (Sharafi et al., 2010). Als Datengrundlage dienen zum einen geordnete Mengen von Wörtern bzw. *Dokumente*, wie im vorliegenden Fall eine Rezension, und eine geordnete Menge von Dokumenten bzw. ein *Korpus*. Der Kernpunkt dieses Modells ist es, jedes Dokument bzw. jede Kundenrezension als einen numerischen Vektor darzustellen, wobei jeder Vektoreintrag der absoluten Häufigkeit eines bestimmten Wortes in der jeweiligen Rezension entspricht. Dabei kommt eine Document Term Matrix (kurz: *DTM*) zustande, die eine Dimension von $(d \times n)$ besitzt und deren Elemente reelle Zahlen sind. d stellt dabei die Anzahl der Rezensionen und n die Anzahl der Wörter im Vokabular dar, sodass jede Zeile eine Rezension und jede Spalte ein Wort repräsentiert.

$$DTM = \begin{pmatrix} f_{1,w_1} & \cdots & f_{1,w_n} \\ \vdots & \ddots & \vdots \\ f_{d,w_1} & \cdots & f_{d,w_n} \end{pmatrix} \in \mathbb{R}^{d \times n}$$

f_{k,w_j} : Absolute Häufigkeit des Wortes w_j in der k -ten Kundenrezension

$k : \{1, \dots, d\}$

$j : \{1, \dots, n\}$

Ausgehend vom folgenden fiktiven Beispielkorpus, der aus drei Sätzen besteht, resultieren die in Tabelle 2 dargestellten absoluten Häufigkeiten. Diese sind gleichzeitig die Einträge der DTM und somit die numerischen Dokumentrepräsentationen, die im Rahmen des Bag of Words (nachfolgend mit *BoW* abgekürzt) entstehen.

Dokument 1: he likes eating banana

Dokument 2: she likes eating cakes he likes drinking banana juice

Dokument 3: he likes drinking tomato juice

Dokument	he	likes	eating	banana	she	cakes	drinking	juice	tomato
1	1	1	1	1	0	0	0	0	0
2	1	2	1	1	1	1	1	1	0
3	1	1	0	0	0	0	1	1	1

Tabelle 2: Absolute Häufigkeitstabelle für jedes Wort, das in einem Dokument vorkommt. Jede Zeile stellt gleichzeitig die jeweilige Dokumentrepräsentation im BoW-Modell dar

Verallgemeinert wird der Text also zerlegt und auf Basis von jeweils $n \in N$ aufeinanderfolgenden Fragmenten die DTM gebildet. Es kommen zur Konstruktion von DTM nicht nur Unigrame, sprich einzelne Wörter (wie im vorherigen Beispiel), sondern auch sogenannte *N-Grame* in Frage, wobei N für die Anzahl der einbezogenen Wörter steht (Jurafsky, 2000). So wird der Satz *he likes eating banana* im Falle eines Bigrams in *he likes*, *likes eating*, *eating banana* aufgespalten. Dies führt also dazu, dass beispielsweise Negierungen (wie *not good*) als Wortkombinationen in die Modellbildung einfließen und nicht einzeln, wie im Unigram-Fall. Auch wenn N-Grame eine detaillierte Untersuchung der Wörter und deren Inhaltes erlauben, ist zu beachten, dass mit steigendem N die Häufigkeit der Fragmente im Korpus sinkt, was wiederum eine Gefahr von Overfitting darstellt. In Kapitel 5 wird auf die Anwendung von Unigrammen und Bigrammen näher eingegangen.

3.3 TF-IDF Modell

Da das BoW-Modell lediglich die absoluten Häufigkeiten der Wörter im Dokument als Grundlage für die Dokumentrepräsentationen hat, werden die Terme gleichermaßen behandelt. Dies kann zu eventuellen Problemen führen. So können Wörter, die in sehr vielen Dokumenten des Korpus vorkommen, aber zugleich keine essentielle Rolle für den Inhalt spielen, tendenziell einen zu hohen Stellenwert einnehmen. Dadurch werden die für die Identifizierung eigentlich relevanten Wörter unterdrückt, denn schließlich haben insbesondere häufig vorkommende Elemente oft keine oder wenig Aussagekraft hinsichtlich des Dokumentinhalts (Sarkar, 2016).

Daher ist es sinnvoll, aufbauend auf dem BoW-Modell, Gewichte einzuführen, die die genannte Problematik durch entsprechende Heruntergewichtung unter Kontrolle halten. Als Basis für das *TF-IDF Modell* dienen zwei Größen: die Termfrequenz (*TF*) und die inverse Dokumentenfrequenz (*IDF*). Das resultierende Modell nennt sich dann *TF-IDF Modell*. Während *TF* die absolute Häufigkeit analog zum BoW-Modell angibt (vgl. Abschnitt 3.2), wird *IDF* folgendermaßen berechnet (Sparck Jones, 1972):

$$idf_{w_j} = \log\left(\frac{d}{df_{w_j}}\right)$$

d : Anzahl aller Dokumente im Korpus

df_{w_j} : Anzahl der Dokumente, in denen das Wort w_j vorkommt

Je größer die Anzahl der Dokumente, in denen ein bestimmtes Wort vorkommt, desto kleiner ist seine inverse Dokumentenfrequenz, da für die Logarithmusfunktion gilt:

$\log\left(\frac{d}{df_{w_j}}\right) \xrightarrow{\frac{d}{df_{w_j}} \searrow 1} 0$. Die multiplikative Hinzunahme von idf_{w_j} führt dazu, dass Wörter, die zu häufig in Dokumenten auftreten, heruntergewichtet werden, um eine sinnvolle Ermittlung der Relevanz zu ermöglichen (Heyer et al., 2006).

Zu erwähnen ist an dieser Stelle, dass die IDF-Gewichte im Unterschied zur TF für jedes Wort dokumentenübergreifend berechnet werden. Die endgültigen TF-IDF Gewichte kommen durch die Multiplikation der beiden Größen wie folgt zustande:

$$tfidfk,w_j = tfk,w_j \cdot idf_{w_j}$$

tfk,w_j : Absolute Häufigkeit des Wortes w_j in der k -ten Kundenrezension

idf_{w_j} : Inverse Dokumentenfrequenz des Wortes w_j

$k : \{1, \dots, d\}$

$j : \{1, \dots, n\}$

Zurückgreifend auf die im Abschnitt 3.2 aufgeführten Beispielsätze werden in Tabelle 3 die zugehörigen IDF-Gewichte berechnet. Durch die darauffolgende Multiplikation mit den TF aus Tabelle 2 ergeben sich die endgültigen Dokumentvektoren des TF-IDF Modells.

Dokument 1: he likes eating banana

Dokument 2: she likes eating cakes he likes drinking banana juice

Dokument 3: he likes drinking tomato juice

	he	likes	eating	banana	she	cakes	drinking	juice	tomato
df_{w_j}	3	3	2	2	1	1	2	2	1
idf_{w_j}	0	0	0.18	0.18	0.48	0.48	0.18	0.18	0.48

Tabelle 3: Berechnung der Dokumentenfrequenz (DF) und der Inversen Dokumentenfrequenz (IDF) für jedes Wort w_j auf Basis von drei Beispielsätzen. Bemerkenswert ist, dass zum Beispiel das Wort *likes*, welches zur Identifikation der Sätze im vorliegenden Korpus-Beispiel nicht hinreichend ist, ein IDF-Gewicht von 0 bekommt. Wohingegen dem Wort *tomato* eine vergleichsweise hohe Relevanz verliehen wird

Dokument	he	likes	eating	banana	she	cakes	drinking	juice	tomato
1	0	0	0.18	0.18	0	0	0	0	0
2	0	0	0.18	0.18	0.48	0.48	0.18	0.18	0
3	0	0	0	0	0	0	0.18	0.18	0.48

Tabelle 4: Dokumentrepräsentationen im TF-IDF Modell für jeweils einen Beispielsatz. Dabei kommt jede Zeile durch die Multiplikation von der Inversen Dokumentenfrequenz (IDF) aus Tabelle 3 mit den entsprechenden Termfrequenzen (TF) aus Tabelle 2 zustande

Zusammenfassend lässt sich sagen, dass die BoW basierten Modelle trotz ihrer relativ einfachen Natur verhältnismäßig zufriedenstellende Ergebnisse liefern (Feldman and Sanger, 2007). Die Document Term Matrix, die auf der Vokabulargröße beruht, besitzt oft eine zu hohe Dimension. Zudem wird der semantische Zusammenhang zwischen den Wörtern bei dieser Methodik nicht berücksichtigt, und die Information über die Reihenfolge der Wörter wird vollkommen ignoriert. Dies führt zum Beispiel dazu, dass die hypothetischen Rezensionen *wonderful game with awful players* (sinngemäß: „Ein wunderbares Spiel mit schrecklichen Mitspielern“) und *awful game with wonderful players* (sinngemäß: „Ein schreckliches Spiel mit wunderbaren Mitspielern“) durch die exakt identischen Wortvektoren repräsentiert werden, obwohl diese gegensätzliche Meinungen darstellen. Im Folgenden wird eine der möglichen Techniken vorgestellt, wie dieses Problem umgangen werden kann.

3.4 Global Vectors Modell

Im Kontrast zu den in Abschnitten 3.2 und 3.3 eingeführten Möglichkeiten, Textdokumente bzw. Rezensionstexte durch numerische Vektoren zu repräsentieren, steht das Global Vectors Modell (kurz: *GloVe*). Dieses wurde 2014 vom Institut für Informatik an der Universität Stanford entwickelt mit dem Ziel, den semantischen Zusammenhang zwischen den Wörtern in den Bildungsprozess der Dokumentvektoren miteinzubeziehen (Pennington et al., 2014). Es fußt auf einem Algorithmus zur Generierung von Wortvektoren. Mit Hilfe derer sollen Dokumente so repräsentiert werden, dass der semantische und syntaktische Zusammenhang zwischen den Wörtern zur linearen Beziehung zwischen den Wortvektoren überführt wird. Das Modell soll also am Ende in der Lage sein, Gleichungen folgender Art herbeizuführen: $king - queen \approx male - female$.

Ein alphabetisch sortiertes Vokabular W mit der Größe n wird nachfolgend betrachtet, wobei w_1, w_2, \dots, w_n Wörter im Vokabular sind und v_1, v_2, \dots, v_n die Wortvektoren mit der festgelegten Dimension m . Primäres Ziel ist es also, jedem Wort $w_i \in W$ einen Wortvektor $v_i \in \mathbb{R}^m$ zuzuordnen.

Ausgangspunkt ist die symmetrische *co-occurrence Matrix* $C \in \mathbb{R}^{n \times n}$. Die Matrixeinträge C_{ij} zeigen dabei wie häufig ein Wort w_j im Kontext oder in der Nachbarschaft von einem Wort w_i auftritt. Die Richtung und der Umfang der Nachbarschaft müssen vorher definiert werden. Beispielsweise bei einer symmetrischen Nachbarschaft mit einer sogenannten *Fenstergröße* von $l = 2$ werden für das Wort w_i die folgenden benachbarten Wörter umfasst: $w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}$. Je weiter entfernt die Wörter voneinander sind, desto weniger beeinflussen sie sich in der Regel gegenseitig. Daher ist es sinnvoll, die absoluten Häufigkeiten zu gewichten. Für die spätere Analyse wird das Gewichtungsschema $g_s = 1/s$ verwendet, wobei s mit $s \in \{1, \dots, l\}$ die Entfernung (gemessen mit der Anzahl der dazwischen liegenden Wörter) zum Kontextwort w_i darstellt. Für das vorherige Beispiel mit $l = 2$ sehen die Gewichte für die entsprechenden Wörter also folgendermaßen aus:

$$\begin{aligned} w_{i-2} &\rightarrow g_2 = 1/2 \\ w_{i-1} &\rightarrow g_1 = 1 \\ w_{i+1} &\rightarrow g_1 = 1 \\ w_{i+2} &\rightarrow g_2 = 1/2 \end{aligned}$$

Dies kann anhand von einem der obigen Beispiele veranschaulicht werden: *Dokument 2: she likes eating cakes he likes drinking banana juice*. Zu betonen ist, dass an

dieser Stelle Dokument 2 der Einfachheit halber den gesamten Korpus darstellt. Der Tabelle 5 kann entnommen werden, dass beispielsweise das Wort *likes* (in der achten Spalte) im Kontext vom Wort *cakes* (in der ersten Zeile) 2 mal vorkommt, da die Fenstergröße von zwei für das Kontextwort *cakes* die folgenden Wörter in roter Farbe umfasst: *she likes eating cakes he likes drinking banana juice*. In der Tabelle 6 kann man erkennen, dass der Eintrag des Wortes *likes* von 2 zu 1 umgewandelt wird. Dies kommt durch die Gewichtung zustande, da der Abstand zwischen *cakes* und *likes* jeweils 2 beträgt und der zugehörige Matrixeintrag somit das Gewicht 1/2 bekommt.

	cakes	eating	drinking	banana	juice	he	she	likes
cakes	0	1	0	0	0	1	0	2
eating	1	0	0	0	0	1	1	1
drinking	0	0	0	1	1	1	0	1
banana	0	0	1	0	1	0	0	1
juice	0	0	1	1	0	0	0	0
he	1	1	1	0	0	0	0	1
she	0	1	0	0	0	0	0	1
likes	2	1	1	1	0	1	1	0

Tabelle 5: Einträge der ungewichteten co-occurrence Matrix in tabellarischer Form. Grundlage ist der Beispielsatz 2 als Korpus, Fenstergröße beträgt 2

	cakes	eating	drinking	banana	juice	he	she	likes
cakes	0	1	0	0	0	1	0	1
eating	1	0	0	0	0	0.5	0.5	1
drinking	0	0	0	1	0.5	0.5	0	1
banana	0	0	1	0	1	0	0	0.5
juice	0	0	0.5	1	0	0	0	0
he	1	0.5	0.5	0	0	0	0	1
she	0	0.5	0	0	0	0	0	1
likes	1	1	1	0.5	0	1	1	0

Tabelle 6: Einträge der mit dem inversen Abstand zum Kontextwort gewichteten co-occurrence Matrix in tabellarischer Form. Grundlage ist der Beispielsatz 2 als Korpus, Fenstergröße beträgt 2

Die co-occurrence Matrix liefert somit einen Grundbaustein für die weiteren Berechnungen. So entspricht $p_{ij} = \frac{C_{ij}}{C_i}$ für $i, j \in \{1, \dots, n\}$ der Wahrscheinlichkeit, dass das Wort w_j im Kontext vom Wort w_i vorkommt. Hierbei gibt $C_i = \sum_{j=1}^n C_{ij}$ die absolute Häufigkeit an, dass ein beliebiges Wort aus dem Vokabular W im Kontext von w_i vorkommt. Es ist gemäß Pennington et al. (2014) naheliegend, dass der semantische Zusammenhang zwischen zwei Wortvektoren mittels Chancen folgender Art berücksichtigt wird: p_{ik}/p_{jk} . Dabei geht es um die Verhältnisse der co-occurrence Wahrscheinlichkeiten, nämlich wie wahrscheinlich es ist, dass das Wort w_k im Kontext von w_i im Vergleich zu w_j auftritt. Einen ähnlichen, chancenbasierten Ansatz findet man später bei der logistischen Regression im Abschnitt 4.1 wieder.

Im Wortvektormodell, welches die Abbildung von linearen Beziehungen zwischen den Wörtern erzielt, soll die stetige Funktion $f(v_i, v_j, v_k)$ soweit wie möglich diesem

Verhältnis ähneln p_{ik}/p_{jk} . Wenn ein Wort w_i in einer engeren Beziehung zu w_k steht, als das Wort w_j , dann soll das obige Verhältnis derer Wahrscheinlichkeiten groß sein, aber auch große Differenzen der Wortvektoren v_i und v_j sollen entsprechend erwartet und durch die Funktion explizit berücksichtigt werden. Hingegen sind bei ähnlichen Wörtern kleinere Differenzen zu erwarten. Daher soll die Funktion f ihre Abhängigkeiten folgendermaßen anpassen, wobei der \tilde{v}_k Vektor in der Regel dem v_k Vektor entspricht mit der alleinigen Unterscheidung der Indexnotation (Pennington et al., 2014):

$$f((v_i - v_j), \tilde{v}_k) = p_{ik}/p_{jk} \quad (1)$$

Wie bereits erwähnt, sind v_i, v_j und \tilde{v}_k Wortvektoren gleicher Dimension und repräsentieren die Wörter w_i, w_j bzw. w_k . Jedoch zeigte Ciresan et al. (2012), dass das Trainieren mehrerer Vektoren und das Kombinieren von ihren Ergebnissen ein Überanpassungsproblem verhindern und zu besseren Endergebnissen führen kann, weshalb die endgültigen Wortvektoren in Form der folgenden Summe verwendet werden: $v + \tilde{v}$ (Pennington et al., 2014). Trotzdem müssen später beide Wortvektoren einzeln geschätzt werden. Bei der obigen Darstellung (1) ist zu berücksichtigen, dass die rechte Seite von der Gleichung einen Skalar darstellt, während die Argumente von f auf der linken Seite in einem Vektor resultieren. Um dieses Problem zu umgehen und die lineare Struktur aufrechtzuerhalten, sollen die Argumente mithilfe der Matrixmultiplikation so

$$f((v_i - v_j)^T \tilde{v}_k) = f(v_i^T \tilde{v}_k - v_j^T \tilde{v}_k) = \frac{p_{ik}}{p_{jk}} = \frac{C_{ik}/C_i}{C_{jk}/C_j} \quad (2)$$

umschrieben werden. Unter Ausnutzung der Einträge der co-occurrence Matrix C wird die Formulierung (2) ermöglicht. Für $f = \exp$ führt dies zu

$$v_i^T \tilde{v}_k + b_i + \tilde{b}_k \approx \log(C_{ik}), \quad (3)$$

wobei $b_i \approx \log(C_i)$ der Bias für v_i und \tilde{b}_k für \tilde{v}_k ist. $\log(C_{ik})$ bewirkt, dass den (in einem bestimmten Kontext) zu häufig auftretenden Wörtern ein nicht zu großer Einfluss zugeschrieben wird, in Relation zu weniger häufigen Wörtern. Bei der folgenden Darstellungsform (4) ist sichergestellt, dass das Modell gegenüber dem Vertauschen von v_i und \tilde{v}_i und dem Transponieren von C invariant ist (Pennington et al., 2014):

$$\begin{aligned} \log\left(\frac{p_{ik}}{p_{jk}}\right) &= \log(C_{ik}) - \log(C_i) - \log(C_{jk}) + \log(C_j) = \\ &\approx v_i^T \tilde{v}_k + b_i + \tilde{b}_k - \log(C_i) - v_j^T \tilde{v}_k - b_j - \tilde{b}_k + \log(C_j) \\ &\approx v_i^T \tilde{v}_k - v_j^T \tilde{v}_k \\ &\Leftrightarrow \frac{p_{ik}}{p_{jk}} \approx \exp(v_i^T \tilde{v}_k - v_j^T \tilde{v}_k) \end{aligned} \quad (4)$$

Die Schätzung der Wortvektoren wird durch die Minimierung der Summe der gewichteten quadratischen Abweichungen mit der Verlustfunktion J erreicht

$$J = \sum_{i,j=1}^n g(C_{ij})(v_i^T \tilde{v}_j + b_i + \tilde{b}_j - \log(C_{ij}))^2, \quad (5)$$

wobei die Gewichtsfunktion g gemäß den Autoren derart definiert wird:

$$g(C_{ij}) = \begin{cases} (C_{ij}/C_{max})^\alpha, & \text{wenn } C_{ij} < C_{max} \\ 1, & \text{sonst} \end{cases}$$

Der sogenannte Cut-Off C_{max} und der Parameter α müssen vorher definiert werden. Für die spätere Analyse werden $C_{max} = 100$ und $\alpha = 0.75$ festgesetzt, nachdem die Autoren eine empirische Motivation und Empfehlung für diese Wahl liefern. Im Allgemeinen erfüllt g folgende Eigenschaften: $g(0) = 0$, sodass die nie gemeinsam vorkommenden Wörter die Verlustfunktion nicht beeinflussen; $g(C_{ij})$ ist nicht fallend, damit zu selten vorkommende Wörter keine übermäßig große Gewichtung erhalten; $g(C_{ij})$ soll relativ klein sein, um eine Übergewichtung der zu großen co-occurrence Matrixeinträgen zu vermeiden. Abschließend werden die Parameter durch die Minimierung der Verlustfunktion J geschätzt (Pennington et al., 2014). Die Wortvektoren sind gemäß der Zielsetzung des GloVe-Modells für jedes Wort eindeutig, unabhängig von den Dokumenten, in denen sie vorkommen. Denn ein übergreifendes Ziel des GloVe-Modells ist die Schätzung der Wortvektoren, die eine feste und bestimmte Stelle im Vektorraum besitzen. Wie gut dies erreicht wird, hängt vor allem davon ab, inwiefern die oben vorgestellten Annahmen erfüllt werden, und welcher Korpus zugrundeliegt. Die Dimension der entstandenen Matrix von Wortvektoren, der sogenannten *embeddings*, lässt sich zum einen durch die anfangs festgelegte Dimension von Wortvektoren und zum anderen durch die beim Normalisierungsschritt determinierte Größe des Vokabulars festlegen. Es gibt verschiedene Möglichkeiten, um die geschätzten Wortvektoren zu Dokumentvektoren zusammenzufassen. Bei der gängigsten Variante wird das einfache arithmetische Mittel aller Vektoren der im jeweiligen Dokument vorhandenen Wörter herangezogen. Zudem kann das mit den TF-IDF-Einträgen multiplikativ gewichtete arithmetische Mittel verwendet werden (vgl. Abschnitt 5.1).

Zusammenfassend kann festgehalten werden, dass das GloVe-Modell - unter Berücksichtigung des semantischen Zusammenhangs zwischen den Wörtern in einem Korpus - eine lineare Beziehung derer im Vektorraum ermöglicht. Dabei werden zu häufig auftretende Wörter entsprechend heruntergewichtet. Auch wenn die Idee und die dahinterstehende Methodik relativ ersichtlich sind, hat das Modell auch seine Schattenseiten: Die Wortvektoren werden auf Basis eines bestimmten Korpus generiert, dessen Größe und Inhalt eine relevante Rolle spielen (Rezaeinia et al., 2017). So kann die Verwendung eines kleinen Korpus, dessen Wortschatz aus einem konkreten Fachbereich stammt, dazu führen, dass sich die Wortvektoren des Modells im Allgemeinen nicht korrekt verwenden lassen, sondern nur auf diesem Bereich spezifiziert sind. Dann kann möglicherweise ein Übertragbarkeitsproblem auftreten, sodass die auf solchem Korpus trainierten Wortvektoren sich bei einer anderen Datensituation nicht verwenden lassen. Andererseits kann ein zu allgemeiner und inhaltlich vielfältiger Korpus ebenfalls zu ungenauen Schätzungen führen, worauf in der weiteren Auswertung näher eingegangen wird. Zudem werden die Wortvektoren unter Verwendung der co-occurrence Matrix trainiert, was verhältnismäßig rechenaufwendig ist.

4 Textklassifikation: Machine Learning Verfahren

Im vorherigen Kapitel wurden Methoden zur Bildung von numerischen Dokumentrepräsentationen beschrieben, worauf basierend überwachte Machine Learning Algorithmen

Vorhersagen ermöglichen können. Wie bereits erwähnt, ist das grundlegende Ziel der vorliegenden Arbeit die Vorhersage, zu welcher der zwei Klassen (positiv oder negativ) ein Rezensionstext bzw. der zugehörige Dokumentvektor gehört. Es handelt sich also um ein Klassifikationsproblem. In diesem Kapitel werden verschiedene Klassifikationsverfahren vorgestellt, die bei der späteren Analyse eingesetzt werden. Logistische Regression, Naïve Bayes und Random Forests werden nachfolgend detailliert erläutert.

4.1 Logistische Regression

Im Allgemeinen verfolgen Regressionsansätze das Ziel, eine Zielvariable y durch die geeignete Kombination von $x = (x_1, x_2, \dots, x_n)$ zu beschreiben. Bei der binären Regression werden Modelle mit einer binären, durch 0 und 1 kodierten Zielgröße behandelt. Durch die Wahl der logistischen Responsefunktion ist das logistische Regressionsmodell gegeben, welches nachfolgend detailliert erläutert werden soll.

Ein weit verbreitetes Beispiel in der Literatur ist die Klassifikation von E-Mails in Spam und Nicht Spam. Logistische Regression kann also beispielsweise eingesetzt werden, um aufgrund des gegebenen Textes als Einflussvariable, eine eingehende E-Mail in den Spam-Ordner zu verschieben, falls diese als Spam klassifiziert wird. Im Falle der vorliegenden Arbeit geht es, wie bereits erwähnt, um die Klassifikation in zwei Bewertungskategorien auf Basis von numerischen Repräsentationen der Rezensionstexte, welche dabei als Einflussgrößen fungieren. Im Vordergrund steht also die Modellierung und Schätzung des Einflusses von Kovariablen auf die geschätzte Wahrscheinlichkeit. Diese kann naturgemäß Werte zwischen 0 und 1 annehmen (Fahrmeir et al., 2007).

In Abbildung 4 ist eine S-förmige Kurve bestehend aus Datenpunkten zu sehen. Grundlage dabei ist ein Teil des in R zur Verfügung gestellten `mtcars`-Datensatzes. Dieses Beispiel soll die Prognosen basierend auf dem logistischen Regressionsmodell veranschaulichen. Auf der x-Achse sind die Werte einer Einflussvariable und auf der y-Achse die geschätzten Wahrscheinlichkeiten abgetragen. Die Variablen namens `hp` bzw. `vs` haben im Kontext dieses Beispiels keine Relevanz und werden deshalb nicht näher erläutert. Beobachtungen mit bestimmten Werten fallen somit in eine der beiden Klassen, je nach dem, welche Wahrscheinlichkeit für die Klasse 1 prognostiziert wird: bei $\pi \leq 0.5$ wird die Klasse 0, bei $\pi > 0.5$ die Klasse 1 prognostiziert. Der Vollständigkeit halber sei erwähnt, dass die Schwelle von 0.5 theoretisch beliebig gewählt werden kann. Für die spätere Implementierung ist jedoch nur 0.5 als Schwellenwert von Bedeutung. Die 0-1-Kodierung der Zielvariable kann im Bezugsrahmen der Kundenrezensionen wie folgt verstanden werden

$$y = \begin{cases} 1, & \text{positive Rezension} \\ 0, & \text{negative Rezension} \end{cases}$$

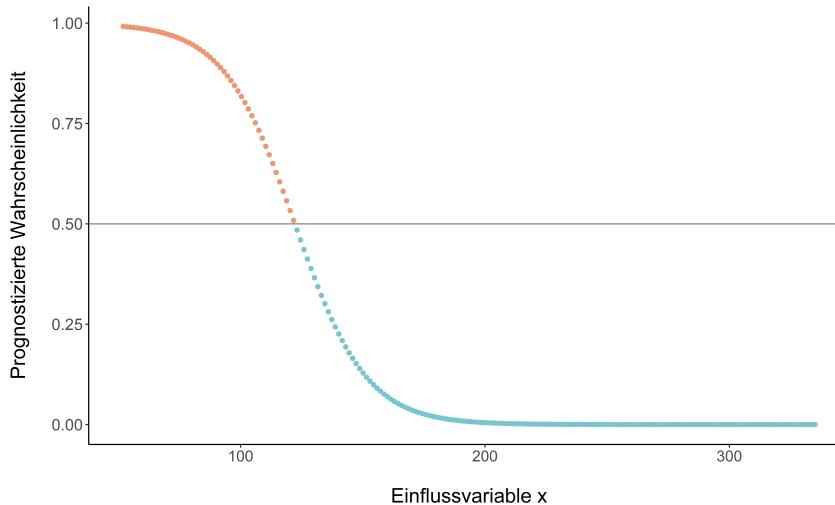


Abbildung 4: Logistisches Regressionsmodell basierend auf einem Beispieldatensatz. Aufgrund einer metrischen Einflussgröße auf der x-Achse, wird die Wahrscheinlichkeit geschätzt (y-Achse), mit der ein Datenpunkt in die fiktive Klasse 1 eingeordnet wird. Die horizontale Linie stellt dabei die Schwelle dar. Für die orangen Datenpunkte wird die Klassenzugehörigkeit 1, für die hellblauen Datenpunkte die Klasse 0 vorhergesagt

Die Logit-Responsefunktion h wird zur Verknüpfung vom linearen Prädiktor η mit der geschätzten Wahrscheinlichkeit π verwendet:

$$\pi = P(y = 1|x) = h(\eta) = \frac{\exp(\eta)}{1 + \exp(\eta)}$$

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

Generell stellen die Koeffizienten β_j mit $j = 1, \dots, n$ den Einfluss der Kovariablen auf die Zielgröße dar.

Die *Chancen* oder die *Odds* werden berechnet durch das Verhältnis der auf die Kovariablen bedingten Wahrscheinlichkeit für das Eintreten der Klasse 1 zur bedingten Wahrscheinlichkeit des Nichteintretens dieser:

$$Odds = \frac{\pi}{1 - \pi} = \exp(\beta_0) \cdot \exp(\beta_1 x_1) \cdot \exp(\beta_2 x_2) \cdot \dots \cdot \exp(\beta_n x_n)$$

Die Chance kann Werte zwischen 0 und ∞ annehmen, wobei ein Wert zwischen 0 und 1 der Klasse 0, und Werte ab 1 der Klasse 1 entsprechen. Diese Größe wird im Rahmen der logistischen Regression insbesondere dafür eingesetzt, um Interpretationen hinsichtlich der Einflüsse einzelner Kovariablen mithilfe der Chancenverhältnisse zu beschreiben. Da in dieser Arbeit der Fokus nicht auf der Untersuchung der einzelnen Einflüsse der Kovariablen, sondern allein auf der Vorhersage liegt, wird auf die Odds nicht weiter eingegangen.

Die Schätzung der Parameter β_j erfolgt üblicherweise durch die Maximierung der Likelihood basierend auf y und bedingt durch x . Die zugrundeliegende Verteilungsannahme der Zielgröße, die auf Kovariablen bedingt ist, ist die Bernoulli-Verteilung mit $y_k \sim B(1, \pi_k)$ für jede Beobachtung oder Rezension k mit $k = 1, \dots, d$. Die zugehörige Wahrscheinlichkeitsfunktion lässt sich wie folgt darstellen:

$$f(y_k|\pi_k) = \pi_k^{y_k} (1 - \pi_k)^{1-y_k}$$

Durch die oben ausgeführte Beziehung zwischen π und η hängt die Wahrscheinlichkeitsfunktion $f(y_k|\pi_k)$ von β_j ab und zwar bedingt auf ein bestimmtes x_k (Fahrmeir et al., 2007).

Unter der Annahme der bedingten Unabhängigkeit von der Zielvariable wird die Likelihood $L(\beta_j)$ folgendermaßen berechnet:

$$L(\beta_j) = \prod_{k=1}^d \pi_k^{y_k} (1 - \pi_k)^{1-y_k}$$

Die Intention hinter der Maximum-Likelihood-Schätzung für das logistische Modell ist es, die Koeffizienten β_j so zu schätzen, dass die geschätzte Wahrscheinlichkeit π für alle Rezensionen der Klasse 1 einen Wert nahe 1 und für die der Klasse 0 einen Wert nahe 0 einnimmt. Die Maximierung von $L(\beta_j)$ ergibt sich durch die Logarithmierung

$$l(\beta_j) = \sum_{k=1}^d \log(L(\beta_j)) = \sum_{k=1}^d (y_k(\eta) - \log(1 + \exp(\eta)))$$

und anschließende Ableitung, sodass die daraus resultierende Scorefunktion nach der Nullsetzung die unten stehende Maximum-Likelihood-Gleichung liefert:

$$\begin{aligned} s(\hat{\beta}_j) &= \sum_{k=1}^d x_k \left(y_k - \frac{\exp(\hat{\eta})}{1 + \exp(\hat{\eta})} \right) = 0 \\ \hat{\eta} &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \dots + \hat{\beta}_n x_n \end{aligned}$$

Die Maximum-Likelihood-Schätzer $\hat{\beta}_j$ gehen aus dieser Gleichung als sogenannte Nullstellen hervor. Zur Bestimmung derer kommen verschiedene iterative numerische Algorithmen in Frage. Eine detaillierte Vorstellung und Herleitung sowie Details zu den obigen Berechnungen kann Fahrmeir entnommen werden (Fahrmeir et al., 2007). Nachdem die Koeffizienten geschätzt sind, wird die Prognose für jede Rezension k berechnet.

Für hochdimensionale Datensituationen mit endlichem Stichprobenumfang ist nicht garantiert, dass der Maximum-Likelihood-Schätzer existiert. Dies gilt vor allem, wenn die Anzahl der Kovariablen viel größer, als der Stichprobenumfang ist, was zu instabiler Schätzung und zur Verschlechterung der Prognose führt (Tutz, 2011). Um dieses Problem unter Kontrolle zu halten, werden sogenannte Regularisierungsverfahren verwendet. Dabei wird je nach Verfahren zusätzlich ein bestimmter Penalisationsterm eingeführt. Dieser bestraft betragsmäßig zu große Werte der Parameter β_j und verkleinert somit die resultierende penalisierte Log-Likelihood, die nun anstelle der $l(\beta_j)$ maximiert werden soll (Friedman et al., 2000):

$$l_{pen}(\beta_j) = l(\beta_j) - \lambda \sum_{j=1}^n |\beta_j|^\gamma \quad \text{mit } \gamma > 0$$

Hierdurch wird versucht, das Gleichgewicht zwischen Datentreue und Modellanpassung aufrechtzuerhalten. Der Parameter λ bestimmt dabei, wie stark der Strafterm die Schätzung beeinflusst.

Relevant für diese Arbeit sind zwei Penalisierungsverfahren. Zunächst wird die *Ridge-Schätzung* eingeführt. Dieser liegt die folgende penalisierte Log-Likelihood zugrunde:

$$l_{pen}^{Ridge}(\beta_j) = l(\beta_j) - \lambda \sum_{j=1}^n |\beta_j|^2$$

mit $\gamma = 2$ als quadratischem Strafterm. $l_{pen}^{Ridge}(\beta_j)$ wird maximiert, um die Koeffizienten β_j zu schätzen.

Ein weiterer Spezialfall ist das *Lasso-Verfahren (Least Absolute Shrinkage and Selection Operator)*, welches die ursprüngliche Log-Likelihood $l(\beta_j)$ unter der Nebenbedingung $\sum_{j=1}^n |\beta_j| \leq \tau$ maximiert. Der Steuerungsparameter τ ist dabei positiv (Hastie et al., 2002). Dies ist identisch mit der folgenden Darstellungsform, wobei $\gamma = 1$:

$$l_{pen}^{Lasso}(\beta_j) = l(\beta_j) - \lambda \sum_{j=1}^n |\beta_j|$$

Bei beiden Verfahren werden die Koeffizienten geschrumpft und somit deren Einfluss Richtung Null gedrückt, sodass kleinere Koeffizienten β_j , als bei der Maximierung der ursprünglichen Log-Likelihood $l(\beta_j)$ geschätzt werden. Dabei gilt: Je größer λ oder je kleiner τ , desto stärker wird geschrumpft. Im Gegensatz zu Ridge wird der Einfluss der Koeffizienten bei Lasso nicht nur reduziert, sondern gegebenenfalls auch auf Null gesetzt. Somit inkludiert die Lasso-Penalisation ein Variablenelektionsverfahren.

Alles in allem ist die logistische Regression in ihrer Natur ein relativ einfaches Klassifikationsverfahren, welches keine sehr hohen Rechenleistungen erfordert und eingängige Interpretationen liefert. Jedoch ist es empfindlich gegen Ausreißer. Das Problem bezüglich des Kompromisses zwischen der Varianz und der Verzerrung der Schätzung kann durch die oben aufgeführten Penalisationsverfahren reguliert werden.

4.2 Naïve Bayes

Naïve Bayes ist ein wahrscheinlichkeitsbasierter Klassifikator. Eine Rezension k wird der Klasse y zugeordnet, für die die bedingte Klassifikationswahrscheinlichkeit $P(y|x_1, \dots, x_n)$ am größten ist. Deren Berechnung basiert auf dem Satz von Bayes (Fahrmeir et al., 2016):

$$P(y|x_1, \dots, x_n) = \frac{P(x_1, \dots, x_n|y)P(y)}{P(x_1, \dots, x_n)}$$

- $P(y|x_1, \dots, x_n)$: A-posteriori Wahrscheinlichkeit; Konditionale Wahrscheinlichkeit der Klasse y unter der Bedingung (x_1, \dots, x_n)
- $P(x_1, \dots, x_n|y)$: Konditionale Wahrscheinlichkeit des Ereignisses (x_1, \dots, x_n) bedingt auf die Klassenzugehörigkeit y
- $P(y)$: A-priori Wahrscheinlichkeit der Klasse y
- $P(x_1, \dots, x_n)$: Wahrscheinlichkeit der Kovariablen (x_1, \dots, x_n)

Die Annahme der stochastischen Unabhängigkeit zwischen den Kovariablen bedingt auf die Zielgröße, wird dabei als $P(x_j|y, x_1, \dots, x_{j-1}, x_j, x_{j+1}, \dots, x_n) = P(x_j|y)$ bezeichnet. Diese vereinfachende Annahme wird in der Literatur als *naiv* bezeichnet, da diese in

der Regel verletzt ist (Manning et al., 2010). Für die BoW basierten Modelle (vgl. Abschnitt 3.2) bedeutet dies, dass die (gewichteten) Häufigkeiten der Wörter unabhängig voneinander sind. Für die GloVe-Modelle setzt dies die Unabhängigkeit der einzelnen Koordinaten von Richtungsvektoren voraus, was aber in der Realität nicht der Fall ist (vgl. Abschnitt 3.4). Trotzdem ist der Naïve Bayes Algorithmus in der Lage, meist zufriedenstellende Ergebnisse zu liefern, vor allem im Bereich der Textklassifikation. Vielmehr wird Naïve Bayes als eines der effizientesten und effektivsten Machine Learning Verfahren bezeichnet (Lewis, 1998). Erlaubt wird nun eine Darstellung über das Produkt eindimensionaler Randwahrscheinlichkeiten:

$$P(y|x_1, \dots, x_n) = \frac{\prod_{j=1}^n P(x_j|y)P(y)}{P(x_1, \dots, x_n)}$$

Da $P(x_1, \dots, x_n)$ für alle Klassenzugehörigkeiten identisch und somit eine Konstante ist, kann diese für den Vergleich der Wahrscheinlichkeiten unberücksichtigt bleiben, so dass nur noch die Maximierung des Zählers im Vordergrund steht. Somit ergibt sich die folgende Schreibweise:

$$P(y|x_1, \dots, x_n) \propto \prod_{j=1}^n P(x_j|y)P(y)$$

Die jeweilige Rezension wird dann der Klasse mit der maximalen Wahrscheinlichkeit zugeordnet. Außerdem resultieren je nach Verteilungsannahme von $P(x_j|y)$ verschiedene Naïve Bayes Algorithmen, wobei relevant für die spätere Analyse zum einen die Gaußverteilung ist. Diese Verteilung ist für die stetigen Dokumentvektoren, wie die GloVe-Vektoren (vgl. Abschnitt 3.4), durch

$$P(x_j|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_j - \mu_y)^2}{2\sigma_y^2}\right)$$

gegeben, wobei der Mittelwert μ_y und die Varianz σ_y^2 für die jeweilige Klasse y mittels Maximum-Likelihood geschätzt werden (Chan et al., 1982). Zum anderen kommt für die diskreten Kovariablen die Multinomialverteilung in Frage. Diese ist nicht nur für die BoW-Modelle mit absoluten Häufigkeiten geeignet, sondern funktioniert auch für TF-IDF-Modelle in der Praxis relativ gut, obwohl hierbei nicht zwingend ganze Zahlen vorkommen (Kibriya et al., 2004). Die Multinomialverteilung ist gegeben durch

$$P(x_j|y) = \frac{f_{y,j} + \alpha}{f_y + \alpha n}$$

mit $f_{y,j}$ als absolute Häufigkeit einer bestimmten Ausprägung der Kovariablen x_j , f_y als Gesamtzahl aller Rezensionen der Klasse y , sowie n als Anzahl an Kovariablen (was gleichzusetzen ist mit der Dimension der Wortvektoren in GloVe und der Vokabulargröße in den BoW-basierten Modellen). Die Notwendigkeit der Einführung des sogenannten *Smoothing-Parameters* $\alpha \geq 0$ wird im Zuge des nachfolgenden Beispiels erläutert (Sarkar, 2016).

Anhand eines selbst erstellten Beispiels soll nun die Funktionsweise eines multinomialen Naïve Bayes-Vorhersagemodells veranschaulicht werden. Zu Grunde liegen die in

Tabelle 7 dargestellten Daten. Um die Stimmung eines Menschen in *negativ* oder *positiv* zu klassifizieren, werden zwei kategoriale Kovariablen *Wetter* (Wetter am Tag der fiktiven Befragung; sonnig, bewölkt, regnerisch) und *Beschäftigung* (Beschäftigungsstand des fiktiven Befragten; Arbeitslos, Student, Arbeitnehmer) in Betracht gezogen.

Wetter	Beschäftigung	Stimmung
Sonnig	Student	Positiv
Bewölkt	Arbeitslos	Negativ
Regnerisch	Arbeitnehmer	Positiv
Regnerisch	Student	Negativ
Regnerisch	Arbeitnehmer	Negativ
Sonnig	Arbeitslos	Negativ
Sonnig	Arbeitnehmer	Positiv
Bewölkt	Arbeitslos	Negativ
Sonnig	Student	Positiv
Bewölkt	Student	Positiv
Regnerisch	Arbeitslos	Negativ
Sonnig	Arbeitnehmer	Positiv
Sonnig	Arbeitnehmer	Positiv
Sonnig	Student	Positiv
Regnerisch	Student	Negativ

Tabelle 7: Daten bestehend aus 15 Beobachtungen zum Trainieren eines Naïve Bayes Modells. Zielvariable: Stimmung; Kovariablen: Wetter und Beschäftigung

Die Stimmung eines Studenten an einem sonnigen Tag lässt sich unter Zuhilfenahme der relativen Häufigkeitstabellen 8 und 9 prognostizieren. Es gilt: $P(y = \text{Negativ}) = 7/15$ und $P(y = \text{Positiv}) = 8/15$.

		Wetter		
		Sonnig	Bewölkt	Regnerisch
Stimmung	Positiv	3/4	1/8	1/8
	Negativ	1/7	2/7	4/7

Tabelle 8: Relative Häufigkeitstabelle $P(x_W|y)$. Datengrundlage dabei ist Tabelle 7

		Beschäftigung		
		Arbeitslos	Student	Arbeitnehmer
Stimmung	Positiv	0	1/2	1/2
	Negativ	4/7	2/7	1/7

Tabelle 9: Relative Häufigkeitstabelle $P(x_B|y)$. Datengrundlage dabei ist Tabelle 7

Mithilfe der obigen Formeln kann die Wahrscheinlichkeit, dass ein Student eine positive bzw. negative Stimmung bei den vorgegebenen Umständen hat, folgendermaßen approximiert werden:

$$\begin{aligned}
 P(y = \text{Negativ} | x_W = \text{Sonnig}, x_B = \text{Student}) &\propto P(x_W = \text{Sonnig} | y = \text{Negativ}) \\
 &\quad \cdot P(x_B = \text{Student} | y = \text{Negativ}) \\
 &\quad \cdot P(y = \text{Negativ}) \\
 &= 1/7 \cdot 2/7 \cdot 7/15 \approx 0.019
 \end{aligned}$$

$$\begin{aligned}
 P(y = \text{Positiv} | x_W = \text{Sonnig}, x_B = \text{Student}) &\propto P(x_W = \text{Sonnig} | y = \text{Positiv}) \\
 &\quad \cdot P(x_B = \text{Student} | y = \text{Positiv}) \\
 &\quad \cdot P(y = \text{Positiv}) \\
 &= 3/4 \cdot 1/2 \cdot 8/15 = 0.2
 \end{aligned}$$

Wegen $0.019 < 0.2$ wird die Stimmung des Studenten als positiv klassifiziert. Im Falle eines Arbeitslosen wäre die Berechnung mit $P(\text{Arbeitslos} | \text{Positiv}) = 0$ problematisch, da die a-posteriori Wahrscheinlichkeit für die positive Stimmung wegen der Multiplikation Null betragen würde. Um dies zu umgehen, wird zu den einzelnen Wahrscheinlichkeiten $P(x_j | y)$ (außer bei der Gaußverteilungsannahme mit stetigen Kovariablen) der Smoothing-Parameter α hinzugenommen. Dadurch entsteht zwar eine gewisse Verzerrung, das Problem beim Fehlen einer Kovariablenausprägung in einer der Klassen tritt jedoch nicht mehr auf (Mccallum and Nigam, 2001).

Abschließend lässt sich festhalten, dass das Naïve Bayes Verfahren neben der relativ hohen Effizienz, auch bei der Anwendung auf große Datensätze wenig rechenaufwendig ist. Dennoch ist die zugrundeliegende Annahme der unabhängigen Kovariablen in der Praxis kaum möglich.

4.3 Random Forests

Ein weiteres Klassifikationsverfahren, welches im Rahmen der vorliegenden Arbeit zum Einsatz kommt, ist das von Leo Breiman 2001 entwickelte Random Forests Verfahren. Die Grundidee ist, dass die Zufallswälder (oder Random Forests) durch die zufallsabhängige Bildung von Klassifikationsbäumen jedem Objekt in eindeutiger Weise eine Klasse zuordnen. Dabei werden durch zufälliges Ziehen mit Zurücklegen aus den Trainingsdaten (vgl. Abschnitt 4.4) $B \in \mathbb{N}$ Teildatensätze generiert, was als *Bootstrapping* bezeichnet wird. Darauf basierend wird mithilfe des Classification and Regression Trees (kurz: *CART*) Algorithmus ein Ensemble aus Klassifikationsbäumen konstruiert. Durch die Aggregation der Ergebnisse von B Bäumen ergibt sich die endgültige Klassifikation. Dieses Konzept nennt sich im Allgemeinen *Bagging* (Bootstrap Aggregating). Für eine in den Teildatensätzen nicht enthaltene Kundenrezension wird also durch jeden Klassifikationsbaum eine Klasse prognostiziert. Die endgültige Klassenzugehörigkeit bestimmt die Bewertungskategorie, die am häufigsten von den einzelnen Bäumen prognostiziert wurde (Breiman, 2001). Im Folgenden wird zunächst näher darauf eingegangen, wie die Klassifikationsbäume Zustandekommen. Daraufhin wird die Notwendigkeit des Bootstrapping-Verfahrens detailliert beschrieben.

Ein übergreifendes Ziel solcher Bäume ist die Erstellung von geeigneten Zuordnungsregeln. Ein Baum wird durch die rekursive Partitionierung der Daten in möglichst homogene Partien bezüglich der Zielvariable konstruiert. So wird der Merkmalsraum durch Splitten in Teilmengen aufgeteilt. Während der Wurzelknoten den gesamten Merkmalsraum enthält, sind die inneren Knoten durch die Aufteilung eines vorher gebildeten Knotens entstanden. Für jeden Split steht eine ausgewählte Menge $m \in \mathbb{N}$ an Kovariablen

zur Auswahl. Die zustande gekommenen Teilmengen werden aus allen n Kovariablen zufällig gezogen. Die Bestimmung von m erfolgt oft über $m \approx \sqrt{n}$ oder $m \approx \log(n)$. Der Hauptvorteil der Ziehung von m Kovariablen ist die Verringerung der Korrelation zwischen einzelnen Bäumen, was eine stärkere Stabilisierung der Varianz erlaubt (Hastie et al., 2002).

Die Abbildung 5 zeigt beispielhaft, wie ein Klassifikationsbaum als Abfolge von binären Partitionen einer Stichprobe resultiert. Ausgehend von einem BoW Modell (vgl. Abschnitt 3.2) entsteht hierbei die folgende Klassifikationsregel: Gruppe 1: enthält das Wort 1 häufiger als ein Mal \rightarrow positive Klasse, Gruppe 2: enthält das Wort 1 höchstens 1 Mal und das Wort 11 höchstens 19 Mal \rightarrow positive Klasse, Gruppe 3: enthält das Wort 1 höchstens 1 Mal und das Wort 11 mindestens 20 Mal \rightarrow negative Klasse. Die drei Endknoten bilden also eine disjunkte Partition des Merkmalsraums.

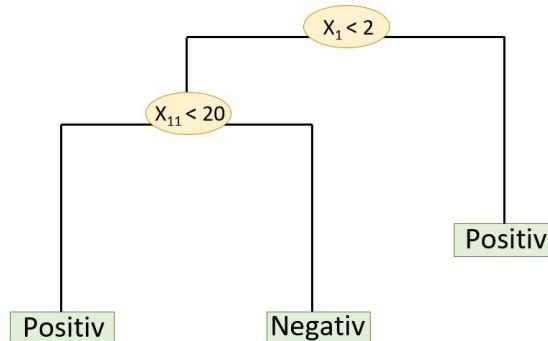


Abbildung 5: Beispiel für einen Klassifikationsbaum. Der Wurzel- und der innere Knoten werden als gelbe Kreise, und die Endknoten als grüne Quadrate dargestellt

Wichtig bei der Aufteilung ist, dass die Gruppen 1-3 möglichst homogen bezüglich der Zielvariablen sind. Die Rezensionen in jedem Knoten werden der Klasse mit geschätzten maximalen Wahrscheinlichkeit $\hat{p}_{t,y}$ zugeordnet. Das kann jedoch bei unbalancierten Daten zu Problemen führen (Chen et al., 2004). Ein Unreinheitsmaß, welches in dieser Arbeit verwendet wird, ist der Gini-Index. Dieser nimmt einen Wert nahe 0 an, falls der jeweilige Knoten „rein“ ist, sprich die Rezensionen einer einzigen Klasse angehören. Das Gini-Unreinheitsmaß wird für jeden Knoten t wie folgt definiert:

$$G_t = \sum_{y \in \{\text{positiv, negativ}\}} \hat{p}_{t,y} (1 - \hat{p}_{t,y})$$

$\hat{p}_{t,y}$: Anteil an Rezensionen der Klasse y im Knoten t

Die schrittweise optimale binäre Trennung der Kovariablen erfolgt durch die Minimierung dieser Kennzahl. Für weitere Details zum CART-Algorithmus wird auf Breiman et al. (1984) verwiesen. Im Allgemeinen weist ein Klassifikationsbaum das Problem der hohen Varianz und Instabilität auf. Um das zu beheben, ist es naheliegend, ein Ensemble aus mehreren Klassifikationsbäumen zu generieren: Ausgehend von B unabhängigen Beobachtungen mit jeweils einer Varianz σ^2 , beträgt die Varianz des arithmetischen Mittels σ^2/B und wird somit kleiner (Hastie et al., 2002). Um einen solchen Effekt der Varianzstabilisierung auch in Bezug auf die Prädiktionen der einzelnen Bäume zu erzielen,

werden B Bootstrap-Stichproben erzeugt und abschließend, nach der Aufstellung von jeweils einem Klassifikationsbaum, das arithmetische Mittel bzw. das häufigste Ergebnis, im Falle einer binären oder kategorialen Zielvariable, berechnet. Im Falle unbalancierter Daten ist es sehr wahrscheinlich, dass sehr wenige oder keine Beobachtungen der Minderheitsklasse in die einzelnen Bootstrap-Stichproben gelangen (Chan et al., 1982).

Die Anzahl der Klassifikationsbäume und somit der Bootstrapstichproben B , sowie die maximale Anzahl an Stufen der Bäume, sollen im Vorfeld festgelegt werden. Zu erwähnen ist, dass die Erhöhung von B durch die Natur des Bootstrappings nicht zu Überanpassungsproblemen (Overfitting) führt (James et al., 2014). Jedoch ist ab einer bestimmten Anzahl kaum oder wenig Verbesserung zu erwarten. Durch die zufällige Auswahl der m Kovariablen, aber auch der einzelnen Bootstrap-Teildatensätze weist der Random Forest Algorithmus meist eine hohe Vorhersagegenauigkeit und Robustheit auf (Strobl et al., 2009). Zudem liefert dieser im Falle hochdimensionaler Daten oft zufriedenstellende Ergebnisse (Strobl et al., 2009). Im Übrigen kann der Random Forests Algorithmus nicht nur bei Klassifikationsproblemen, sondern auch bei einer Regression herangezogen werden. Da beim Bootstrap-Verfahren durchschnittlich $2/3$ der Trainingsdaten zur Baumkonstruktion verwendet werden, können die restlichen sogenannten *out-of-bag* Daten zur Evaluierung der Vorhersagegenauigkeit in Betracht gezogen werden (vgl. Abschnitt 4.4). Nachteilig ist bei Random Forests, dass die Rechenzeit durch die multiplen Bäume und die integrierten Prozesse relativ lange dauert. Durch das Ensemble der Klassifikationsbäume wird zwar eine gewisse Varianzstabilisierung gewonnen, jedoch verlieren Random Forests dadurch die Eigenschaft der einfachen Interpretierbarkeit.

4.4 Kreuzvalidierung

Nach dem Einsatz verschiedener Klassifikationsverfahren und dem Erhalt von Prädiktionen, ist es von besonderem Interesse und essenzieller Bedeutung, eine verlässliche Einschätzung der Vorhersagefähigkeit des Modells vorzunehmen. Dazu gibt es im Falle einer binären Zielvariable verschiedene Qualitätsmaße, die im Abschnitt 4.5 ausführlich vorgestellt werden. Entscheidend ist, dass bei der Berechnung der Kennzahlen keinesfalls die zum Erlernen des Klassifikators verwendeten Lerndaten (im Folgenden als *Trainingsdaten* umschrieben) zur Grundlage genommen werden. Denn ansonsten wird eine zu optimistische Beurteilung vorgenommen. Stattdessen müssen die sogenannten *Testdaten* vor der Modellbildung zum eben genannten Zweck herausgelassen werden. Diese sind in der Regel ein Teil des anfangs zur Verfügung gestellten Datensatzes. Für die vorliegende Arbeit werden dazu 20% des Video Games-Datensatzes als Teststichprobe verwendet, welche mittels einer Zufallsziehung ohne Zurücklegen zustande gekommen ist. Der Prozess der Modellbildung findet auf Basis der restlichen 80% des Datensatzes, sprich auf der Trainingsstichprobe, statt.

Bevor der Prädiktions- oder Generalisierungsfehler auf Basis der Testdaten evaluiert werden kann, sollen die Parameter des Modells optimiert werden. Dies erfolgt mit Hilfe der Validierungsstichprobe, indem die oben beschriebene Trainingsstichprobe weiter in einen kleineren Trainings- und einen Validierungsdatensatz untergliedert wird. Die Parameter, mit deren Einstellung das Modell die geringsten Fehler aufweist, wird als optimal gekennzeichnet und dient als endgültige Grundlage zur Modellbildung (die verschiedenen Definitionen vom Begriff *Fehler* werden im nachfolgenden Abschnitt erläutert). Visuell soll also der ursprüngliche Datensatz idealerweise wie in Abbildung 6 in eine Trainings-, Test- und Validierungsstichprobe aufgeteilt werden, und zwar typischerweise annähernd

mit den abgebildeten Proportionalitäten (Hastie et al., 2002).

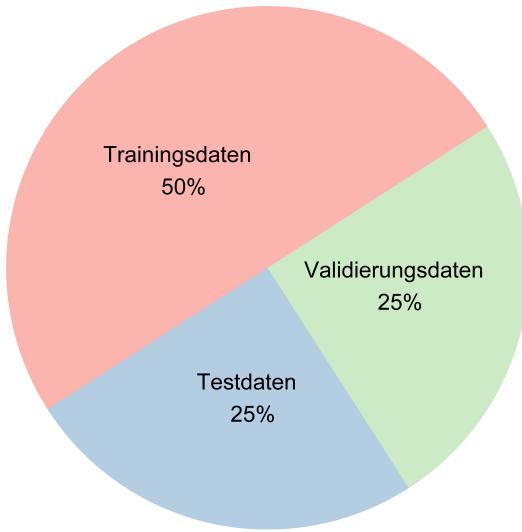


Abbildung 6: Qualitative Darstellung der disjunkten Aufteilung vom Gesamtdatensatz zur Validierung und Beurteilung der Modellgüte

Um noch verlässlichere Aussagen über die Modellvalidität zu erzielen, kann der Validierungsprozess mittels der sogenannten *Kreuzvalidierung* durchgeführt werden. Dabei wird der Trainingsdatensatz per Zufallsziehung in $k \in \mathbb{N}$ möglichst gleich große disjunkte Mengen aufgeteilt. Daraufhin erfolgen k Testdurchläufe, wobei die jeweils i -te Datenmenge als Testdaten zur Beurteilung der Modellgüte, und die restlichen $k - 1$ Datenmengen als Trainingsdaten fungieren. Der Validierungsfehler errechnet sich dann als arithmetisches Mittel aus den Einzelfehlern der k Durchläufe. In dieser Arbeit wird $k = 5$ gewählt. Somit kann beispielsweise das Naïve Bayes Modell mit dem optimalen Smoothing-Parameter λ gefunden werden, nachdem bei der Kreuzvalidierung eine Menge mehrerer Parameter-Kandidaten vorgegeben wurde.

Wie in Kapitel 2 erwähnt, sind die der vorliegenden Arbeit zugrundeliegenden Datensätze bezüglich der Bewertungskategorien nicht ausgewogen. Deshalb ist es in diesem Fall besonders wichtig sicherzustellen, dass bei der Kreuzvalidierung in jedem der k Teildatensätze die Verteilung des Gesamtdatensatzes bezüglich der Zielvariable annähernd erhalten bleibt. Dies kann mithilfe der sogenannten *stratifizierten Kreuzvalidierung* gewährleistet werden. Dadurch soll vermieden werden, dass die Testdatensätze nur die überrepräsentierte Klasse enthalten und somit nur diese als Grundlage für die darauf basierenden Prozedere haben.

4.5 Auswertungskennzahlen

Zur Beurteilung der Prädiktionsgüte eines Klassifikationsmodells oder zur Interpretation der Veränderungen mehrerer Modelle, kommen verschiedene Auswertungskennzahlen in Frage. Diese lassen sich durch die Einträge der sogenannten Konfusionsmatrix herleiten, die eine Gegenüberstellung zwischen den tatsächlichen und prädictierten Klassenzu-

gehörigkeiten zeigt. Dabei werden die absoluten Häufigkeiten f der vier möglichen Fälle in Tabelle 11 eingezzeichnet:

- richtig positiv** (kurz: RP) - die Rezension gehört der positiven Klasse an und wird durch das Modell auch als solche klassifiziert
- falsch positiv** (kurz: FP) - die Rezension gehört der negativen Klasse an, wird aber durch das Modell als positiv klassifiziert
- richtig negativ** (kurz: RN) - die Rezension gehört der negativen Klasse an und wird durch das Modell als solche klassifiziert
- falsch negativ** (kurz: FN) - die Rezension gehört der positiven Klasse an, wird aber durch das Modell als negativ klassifiziert

Tabelle 10: Definition der vier möglichen Fälle bei einer binären Klassifikation

		Prädiktiert	
		Positiv	Negativ
Wahr	Positiv	f_{RP}	f_{FN}
	Negativ	f_{FP}	f_{RN}

Tabelle 11: Die Konfusionsmatrix als Gegenüberstellung zwischen den tatsächlichen und prädiktierten Klassenzugehörigkeiten. Die zugehörigen Abkürzungen werden in Tabelle 10 definiert

Eine der weit verbreiteten Kennzahlen ist das Akkurateitsmaß, welches den Anteil richtig spezifizierter Rezensionen wiedergibt. Dieses ist jedoch in vielen Anwendungen nicht geeignet, da dabei keine Differenzierung zwischen den verschiedenen Klassen vorliegt. Insbesondere im Falle unbalancierter Daten kann dies zu Problemen bezüglich einer zu optimistischen Beurteilung der Modellgüte führen. Der Fokus bei der vorliegenden Arbeit wird daher auf die folgenden drei Beurteilungskennzahlen eines binären Klassifikators gelegt (Van Rijsbergen, 1979):

Recall wird auch als Sensitivität bezeichnet. Berechnet wird dabei der Anteil der richtig spezifizierten positiven Rezensionen unter allen tatsächlich positiven Rezensionen. Es handelt sich also um eine Art Trefferquote:

$$\text{Recall} = \frac{f_{RP}}{f_{RP} + f_{FN}}$$

Precision ist eine weitere Kennzahl, die den Anteil richtig spezifizierter positiver Rezensionen unter allen als positiv klassifizierten Rezensionen berücksichtigt. Je höher der Precision-Wert, desto kleiner ist die Anzahl fälschlicherweise als positiv klassifizierter Rezensionen. Precision lässt sich mit folgender Formel berechnen:

$$\text{Precision} = \frac{f_{RP}}{f_{RP} + f_{FP}}$$

Der **F1 Score** fasst die Recall- und Precision-Maße auf die nachfolgend dargestellte Weise zusammen. Dieser ist ein harmonisches Mittel der beiden letztgenannten Größen und kann als durchschnittliche Messgenauigkeit des Modells interpretiert werden:

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

Diese Kenngrößen können Werte zwischen 0 und 1 annehmen, wobei je höher der entsprechende Wert, desto zufriedenstellender ist die Modellgüte.

5 Anwendung

In diesem Kapitel werden das finale Analyseschema sowie die Implementierung erläutert und abschließend die Ergebnisse diskutiert. Die in den vorherigen Abschnitten eingeführten theoretischen Grundlagen erlauben eine Vielzahl an Modellvariationen im Hinblick auf die Entwicklung eines Verfahrens zur Textklassifikation auf Basis von Kundenrezensionen, welches ihm unbekannte Rezensionen automatisch in die richtige Bewertungskategorie einordnet. Bei der anschließenden Evaluation steht neben der Vorstellung der Ergebnisse, vor allem der Vergleich zwischen verschiedenen Modellierungsvarianten hinsichtlich der Merkmalsextrahierung im Fokus. Die Implementierung und Auswertung erfolgt nahezu ausschließlich in Python (Version 2.7.15), insbesondere mithilfe des Pakets `sklearn` (Pedregosa et al., 2011). Allein die Bildung von GloVe-Wortvektoren wurde in R (Version 3.5.0) durchgeführt, da es sich in Python (aus technischen Gründen) nicht installieren ließ. Außerdem sei an dieser Stelle erwähnt, dass die in dieser Arbeit inkludierten grafischen Visualisierungen in R programmiert sind (vgl. Abb. 2-4, 6, 7). Der Quellcode zu ausgewählten Teilschritten ist im Anhang platziert.

5.1 Implementierung

Zunächst werden die Rezensionen vom Referenz-Datensatz Video Games vorbereitet, indem die Bewertungssterne 1-2 in die positive und 3-5 in die negative Bewertungskategorie eingeteilt werden. Die daraus resultierende prozentuale Häufigkeitsverteilung ist in Abbildung 7 in Form von jeweils einem Balkendiagramm dargestellt. Die Anzahl an negativen bzw. positiven Rezensionen beträgt im Video Games-Datensatz 56.791 bzw. 174.989 und im Beauty-Datensatz 44.230 bzw. 154.272.

Die 3-Sterne-Bewertungen lassen sich aufgrund der ungradzahligen Skala zwar nicht eindeutig als negativ oder positiv bezeichnen, werden jedoch zunächst zur negativen Kategorie zugeordnet. Eine genauere Diskussion dieser Angelegenheit findet sich im Abschnitt 5.2.2.

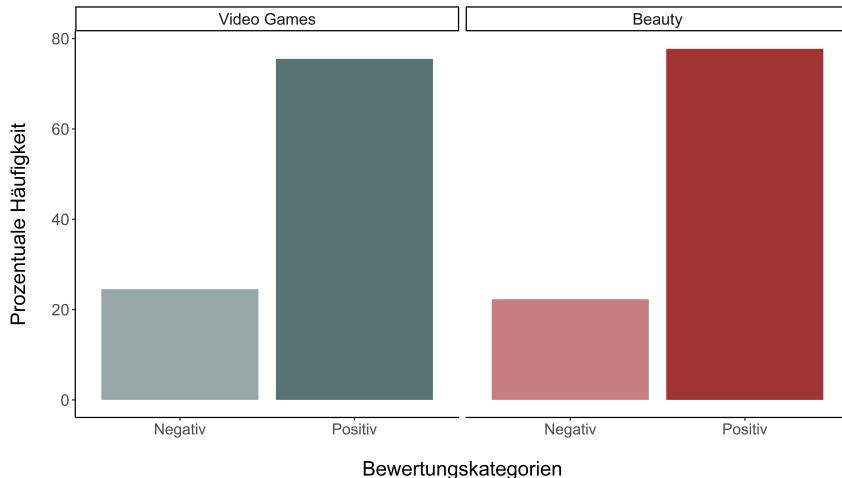


Abbildung 7: Prozentuale Häufigkeitsverteilung der Bewertungskategorien. Hierbei werden 1-, 2-, 3-Sterne-Bewertungen zur negativen und 4-, 5- zur positiven Kategorie zusammengefasst

Anschließend werden die Rezensionstexte normalisiert. Für die BoW-basierten Modelle kommen die Normalisierungsschritte in folgender Reihenfolge zum Einsatz (vgl. Abschnitt 3.1):

- 1) Bereinigung um HTML-Zeichen und -Entitäten
- 2) Ausschreibung von Abkürzungen
- 3) Lemmatisierung
- 4) Entfernung von Sonderzeichen
- 5) Entfernung von Stoppwörtern
- 6) Beschränkung auf Wörter als Satzbestandteile

Für GloVe-basierte Modelle wird bei der Normalisierung grundsätzlich auf die Lemmatisierung sowie die Bereinigung von Stoppwörtern verzichtet. Diese Schritte sind für die BoW-basierten Modelle insbesondere zur Reduktion von Dimensionen der Vektorrepräsentationen sinnvoll. Deren Nützlichkeit hinsichtlich des Trainierens und der Platzierung von Wörtern im Vektorraummodell ist umstritten, da die Dimension hierbei konstant bleibt und zudem dadurch möglicherweise für die Ermittlung des Inhalts relevante Feinheiten unberücksichtigt bleiben. Für weitere Diskussionsdetails sei an dieser Stelle auf Camacho-Collados and Pilehvar (2017) verwiesen. Jedenfalls wird bei der Evaluation der Ergebnisse im Abschnitt 5.2.1 auf diesen Aspekt zurückgegriffen.

Die bei der Programmierung verwendete Funktion `normalize_corpus()` basiert größtenteils auf die gleichnamige Funktion von Sarkar (2016) und wurde hinsichtlich einer Option zur Rechtschreibkorrektur mit geeignetem Umgang bezüglich der Zeichenhäufungen ergänzt (vgl. Abschnitt 3.1). Auch die zugrundeliegende Liste von englischen Abkürzungen `CONTRACTION_MAP`, die von Sarkar (2016) erstellt wurde, wird übernommen. Die Normalisierungsfunktion ist im Anhang zu finden.

Um die Rezensionstexte in Machine Learning Verfahren verwenden zu können, müssen diese durch numerische Vektoren repräsentiert werden. Wie in Kapitel 3 ausführlich beschrieben, erfolgt dies grundsätzlich auf zwei Wegen, die weiter in ihren Charakteristiken variiert werden können: Im Gegensatz zu den BoW-basierten Modellen, berücksichtigen

die GloVe-Modelle auch den semantischen Zusammenhang zwischen den Wörtern. Bei der erstenen Variante kommen die folgenden vier Merkmalsextrahierungsvarianten zum Einsatz: BoW und TF-IDF Modelle mit jeweils Uni- und Bigram Betrachtung (vgl. Abschnitt 3.2 und 3.3). Die dafür verwendete Funktion `build_feature_matrix()` lehnt sich ebenso an Sarkar (2016) an. Diese wurde dahingehend modifiziert, dass in den Rezensionstexten vorkommende Satzteile, die nur aus einem einzigen Buchstaben bestehen, nicht ignoriert werden (vgl. Anhang A). Zu erwähnen ist an dieser Stelle, dass bei der Implementierung von TF-IDF-Modellen, die IDF-Gewichte - aufbauend auf der Darstellung im Abschnitt 3.3 - wie folgt gebildet werden:

$$idf_{w_j} = 1 + \log\left(\frac{1 + d}{1 + df_{w_j}}\right)$$

d : Anzahl aller Dokumente im Korpus

df_{w_j} : Anzahl der Dokumente, in denen das Wort w_j vorkommt

Das zusätzliche Addieren von 1 zur Dokumentfrequenz df_{w_j} (bzw. zu d) kann als künstliche Hinzunahme von einer Rezension, die alle Wörter im Vokabular enthält, verstanden werden. Dies stellt sicher, dass zum einen keine Fehler durch die Division durch 0 entstehen, und zum anderen bewirkt es einen Glättungseffekt für idf_{w_j} . Zudem wird zum Gesamtausdruck eine 1 addiert, damit die Wörter, die tatsächlich eine IDF von 0 haben, sprich in allen Rezensionen vorkommen, nicht vollkommen unberücksichtigt bleiben.

Die GloVe basierten Modelle haben verschiedene Variationsmöglichkeiten zum einen hinsichtlich des zugrundeliegenden Korpus zur Schätzung von Wortvektoren und zum anderen im Hinblick auf die Art der Zusammenfassung von Wortvektoren innerhalb einer Rezension.

Beim ersten GloVe-Modell werden die öffentlich zur Verfügung gestellten vortrainierten 400.000 Wortvektoren mit der Dimension 200 verwendet (Pennington, Jeffrey and Socher, Richard and Manning, Christopher, 2014). Als Grundlage zur Wortvektor-Bildung diente eine Kombination aus *Wikipedia*, einer Online-Enzyklopädie als Teil der Massenmedien und *English Gigaword Fifth Edition*, einer Datenbank, die die Texte von sieben verschiedenen Nachrichtenagenturen archiviert. Die Wortvektoren der jeweiligen Rezension werden anschließend mithilfe einfachen arithmetischen Mittels zu einem 200-dimensionalen Dokumentvektor kombiniert (vgl. Abschnitt 3.4). Dieses Modell wird im weiteren Verlauf als *GloVe-Wiki* umschrieben.

Das zweite Modell unterscheidet sich zum Ersten nur in der Art der Zusammenfassung. Dabei wird ein gewichtetes arithmetisches Mittel gebildet, wobei die TF-IDF-Werte der jeweiligen Rezension (mit der eben beschriebenen Erweiterung) als multiplikative Gewichte fungieren: $v_{k,j}^{tfidf} = tfidf_{k,w_j} \cdot v_j$ mit dem zum Wort w_j gehörenden Vektor v_j der k -ten Rezension. Dies hat zum Vorteil, dass zu häufig vorkommende Wörter heruntergewichtet werden. Das Modell wird als *GloVe-Wiki (gew.)* bezeichnet. Beim Testen dieses Modells erfolgt die Bildung von TF-IDF-Repräsentationen bzw. -Gewichten durch die Heranziehung von jeweiligem Testdatensatz, sodass die Document Term Matrix darauf basierend neu erstellt wird. Dadurch wird auf mögliche Übertragungseffekte hinsichtlich der Gewichtung verzichtet.

Beim dritten Modell werden neben den vortrainierten Wortvektoren, auch eigene, sprich auf Basis des gesamten normalisierten Video Games-Datensatzes gebildete Wortvektoren eingesetzt. Dabei wird die Dimension auf 200 festgesetzt und Wörter, die weni-

ger, als 5 mal im gesamten Korpus vorkommen, ignoriert (vgl. Abschnitt 3.4). Die Implementierung erfolgt mittels des Pakets `text2vec` in R (Selivanov and Wang, 2018). Die zugehörige Funktion `extract_glove()` befindet sich im Anhang (vgl. Funktion 3). Bei dieser Merkmalsextrahierung wird so vorgegangen, dass den Wörtern zunächst die auf dem Video Games-Korpus trainierten Vektoren zugewiesen werden, sofern vorhanden. Da die vortrainierten Vektoren eine viel größere Basis an Vokabular haben, wird beim Fehlen des Wortes im Video Games-Vokabular, auf die vortrainierten Vektoren zugegriffen. Zuletzt wird für jede Rezension das einfache arithmetische Mittel gebildet. Dieses Modell wird im weiteren Verlauf mit *Glove - Wiki, Video Games* abgekürzt.

Die vierte und somit letzte Modellvariation hinsichtlich der Merkmalsextrahierung mit GloVe ergibt sich dadurch, dass im Gegensatz zum dritten Modell lediglich die Wortvektoren verwendet werden, die auf Basis vom Video Games-Datensatz trainiert wurden. Wörter, die in diesem Vokabular nicht vorhanden sind, bleiben also unberücksichtigt. Dies ist vor allem für die Untersuchung der Frage interessant, inwiefern sich das auf einem bestimmten Korpusinhalt trainierte GloVe-Modell auf einen anderen Korpus übertragen lässt. Die Zusammenfassung erfolgt wieder mittels einfachen arithmetischen Mittels. Das Modell wird mit *Glove - Video Games* umschrieben. Für alle Modelle gilt, dass die Wörter, die beim Trainieren nicht vorhanden waren, einen Null-Vektor bekommen.

Somit entstehen insgesamt die folgenden acht Möglichkeiten für die Merkmalsextrahierung: **BoW (Unigram)**; **BoW (Bigram)**; **TF-IDF (Unigram)**; **TF-IDF (Bigram)**; **GloVe - Wiki**; **GloVe - Wiki (gew.)**; **GloVe - Wiki, Video Games**; **GloVe - Video Games**.

Nach der Bildung von numerischen Dokumentrepräsentationen, können verschiedene Machine Learning Klassifikationsverfahren für die Entwicklung eines Systems zur automatischen Einordnung von Rezensionen eingesetzt werden. Dabei werden die in Kapitel 4 dargelegten Verfahren herangezogen: Logistische Regression mit Ridge- und Lasso-Penalisation, Naïve Bayes-Modell und Random Forests-Verfahren. An dieser Stelle wird noch einmal unterstrichen, dass für die BoW-basierten Modelle das multinomiale und für die GloVe-Basierten das gaußsche Naïve Bayes zum Einsatz kommt, abhängig von der getroffenen Verteilungsannahme über die a-priori Wahrscheinlichkeiten. Für jedes der entstehenden 32 Modelle mit einer bestimmten Merkmalsextrahierungsmethode und einem der vier Klassifikationsverfahren, wird der schematische Aufbau des Anwendungsprozesses in Abbildung 8 dargestellt.

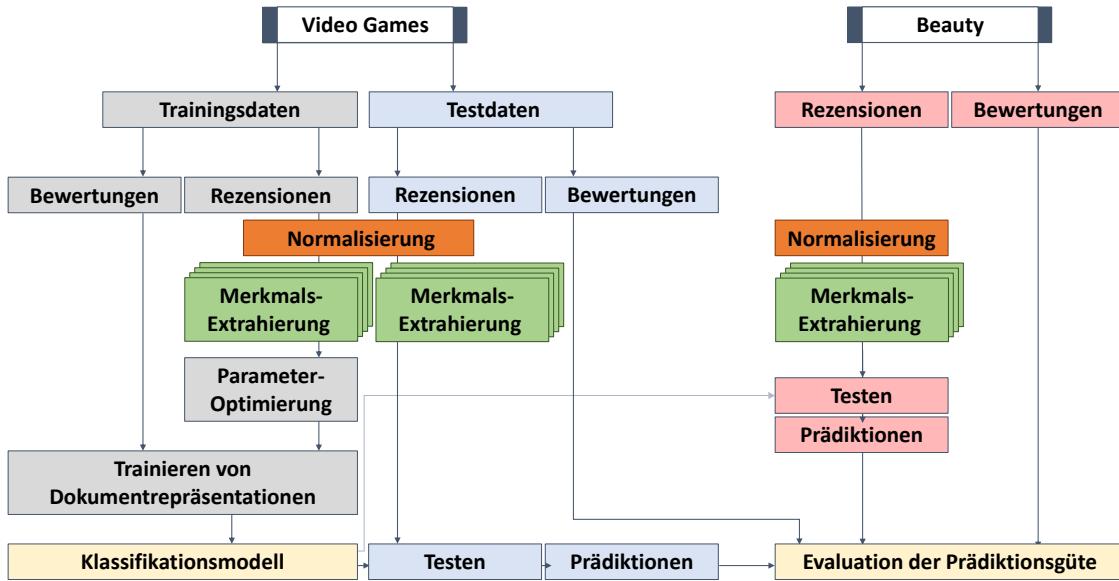


Abbildung 8: Schematischer Aufbau des Anwendungsprozesses. Dabei wird ein Teil des Video Games-Datensatzes zur Parameteroptimierung und zum Trainieren verwendet. Der restliche Teildatensatz, sowie der Beauty-Datensatz kommen als Testdaten zur Beurteilung der Prädiktionsgüte zum Einsatz

Der Ausgangspunkt ist die zufällige Aufteilung des Video Games-Datensatzes (Referenz) in Trainings- (grau) und Teststichprobe (hellblau) im Verhältnis 8:2, die jeweils eine Menge an Rezensionstexten und zugehörigen Bewertungen enthalten. Diese Aufteilung erfolgt nach dem stratifizierten Prinzip, sodass in beiden Teildatensätzen die Verteilung des Gesamtdatensatzes bezüglich der Bewertungskategorien beibehalten wird. Der Trainingsdatensatz enthält somit 185.424 (139.991 positive und 45.433 negative) und der Testdatensatz 46.356 (34.998 positive und 11.358 negative) Rezensionen. Ein zweiter Testdatensatz ist der Beauty-Datensatz (hellrot) mit insgesamt 198.502 Rezensionen (154.272 positive und 44.230 negative). Dieser hat die gleiche Funktion, wie der erste Testdatensatz, nämlich die Evaluation der Prädiktionsgüte, jedoch beinhaltet die Rezensionstexte darin einen anderen Grundkontext (vgl. Kapitel 2). Dies ist insbesondere für die Merkmalsextrahierungsmethode *GloVe - Video Games* im Bezug auf die Übertragbarkeit interessant. Nachdem alle Rezensionen gemäß dem am Anfang dieses Kapitels beschriebenen Prinzip normalisiert und als numerische Vektoren repräsentiert werden, sollen die Hyperparameter auf Basis der Trainingsdaten optimiert werden, wobei aus einer vorgegebenen Mengen jeweils die Parameter mit dem besten oder höchsten F1-Score als Kriterium gewählt werden, da dieser eine Kombination von den übrigen Auswertungskennzahlen dargestellt (vgl. Abschnitt 4.5). An dieser Stelle ist zu betonen, dass die Bildung von BoW-basierten Repräsentationen für die Test-Rezensionen auf Basis der Document Term Matrix von den Trainingsdaten erfolgt, sodass neue Wörter aus den Testdaten, die im Vokabular der Trainingsdaten nicht enthalten sind, unberücksichtigt bleiben (Sarkar, 2016). Bei der vorliegenden Anwendung haben die Dokumentvektoren der BoW und TF-IDF Modelle als Unigram-Darstellung eine Dimension von 178.431, während sich diese für die Bigram-Modelle auf 5.067.342 erhöht. Dementsprechend haben diese Modelle eine höhere Laufzeit. Als Hyperparameter der logistischen Regression mit beiden

Penalisierungsmöglichkeiten kommen λ , der Löser-Algorithmus für das Optimierungsproblem, sowie die maximale Anzahl an Iterationen für den Löser-Algorithmus in Frage. Für weitere Details zu den Letzteren sei auf Defazio et al. (2014) und Fan et al. (2008) verwiesen. Beim multinomialen Naïve Bayes-Verfahren wird der Glättungsparameter α optimiert (vgl. Abschnitt 4.2). Für Random Forests werden in der vorliegenden Arbeit die folgenden drei Hyperparameter getunet: die Anzahl der Bäume, die maximale Anzahl der beim jeden Split zu ziehenden Kovariablen m sowie die maximale Tiefe der Bäume, also die maximale Anzahl an Stufen der Bäume (vgl. Abschnitt 4.3). Nach der Parameteroptimierung mit dem sogenannten `GridSearchCV()` aus dem Paket `sklearn` im Zuge einer stratifizierten 5-fach-Kreuzvalidierung, wird das Klassifikationsmodell mit den in diesem Sinne besten Parametern gewählt und auf Basis der Trainingsdaten trainiert (Pedregosa et al., 2011). Die default- sowie die getunten Parameter jedes in dieser Arbeit aufgestellten Modells befinden sich im Anhang B. Das endgültige Modell soll anschließend die Bewertungskategorien der Rezensionen aus den beiden Testdatensätzen prädiktieren. Die Güte lässt sich anhand der Auswertungskennzahlen, die sich im Rahmen des Vergleichs mit den wahren Bewertungskategorien ergeben, evaluieren. Die Funktion `display_confusion_matrix()` zur Erstellung der Konfusionsmatrix wird von Sarkar (2016) übernommen.

5.2 Diskussion der Ergebnisse

Zunächst sind die primären 32 Modelle zu evaluieren. Die Testergebnisse von diesen sind in den nachfolgenden Tabellen dargestellt, wobei die Zeilen jeweils der Merkmalsextrahierungsmethode, und die Spalten den Klassifikationsverfahren entsprechen. Die Abkürzungen R , P und $F1$ stehen für die Auswertungskennzahlen Recall (Anteil der richtig spezifizierten positiven Rezensionen unter allen tatsächlich positiven Rezensionen), Precision (Anteil richtig spezifizierter positiver Rezensionen unter allen als positiv klassifizierten Rezensionen) und F1-Score (vgl. Abschnitt 4.5). Da Letzterer eine Zusammenfassung von Ersteren beiden darstellt, wird der höchste F1-Score der jeweiligen Zeile markiert. Die Farbe der Markierung resultiert aus der Datengrundlage des Modells: Die Testergebnisse der 20%-Zufallsstichprobe vom Video Games-Datensatz werden mit hellblau (*VG*) und die vom Beauty-Datensatz (*B*) mit hellrot gekennzeichnet.

Die Tabelle 12 zeigt, dass die F1-Scores der verschiedenen Modelle insgesamt ähnlich aussehen, das heißt, alle vier Klassifikatoren resultieren in ähnlichen Ergebnissen, wobei die Ridge-Regression bei allen Extrahierungsvarianten die beste Prädiktionsgüte aufweist. Auch durch die Lasso-Penalisation wird nahezu gleichermaßen prädiktiert, sodass der F1-Score teilweise exakt identisch ausfällt. Auffällig sind die sehr hohen Recall-, und die hingegen vergleichsweise niedrigen Precision-Werte.

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
BoW (Unigram)	VG	0.95	0.87	0.91	0.95	0.87	0.91	0.94	0.85	0.9	1.0	0.75	0.86
BoW (Bigram)	VG	0.95	0.86	0.9	0.93	0.87	0.9	0.99	0.79	0.88	1.0	0.75	0.86
TF-IDF (Unigram)	VG	0.95	0.88	0.91	0.95	0.88	0.91	0.99	0.8	0.88	1.0	0.75	0.86
TF-IDF (Bigram)	VG	0.96	0.85	0.91	0.93	0.87	0.9	1.0	0.77	0.87	1.0	0.75	0.86

Tabelle 12: Ergebnistabelle der BoW-basierten Modelle. Zeilenweise sind die BoW- bzw. TF-IDF-Modelle mit jeweils Uni- und Bigram-Betrachtung dargestellt. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierungsverfahren, multinomiale Naïve Bayes sowie Random Forests. Grundlage hierbei ist ein Teil der Video Games-Rezensionen als Testdatensatz. Die Ergebnistabelle für den Beauty-Test befindet sich im Anhang C

Im Vergleich zu den BoW-basierten Modellen, erreichen die GloVe-Modelle generell kleinere F1-Scores, dennoch bleibt die beste Prädiktionsgüte fast ausschließlich bei der logistischen Regression mit beiden Penalisiertechniken. Auffällig ist die sehr niedrige Trefferquote und somit der niedrige F1-Score der Naïve Bayes-Modelle mit der Gaußverteilungsannahme. Untereinander unterscheiden sich die verschiedenen Trainierungs- und Zusammenfassungsarten von Wortvektoren minimal.

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
GloVe - Wiki	VG	0.94	0.83	0.88	0.94	0.83	0.88	0.44	0.85	0.58	0.99	0.78	0.87
GloVe - Wiki (gew.)	VG	0.99	0.77	0.86	0.98	0.77	0.86	0.22	0.85	0.34	1.0	0.76	0.86
GloVe - Wiki, Video Games	VG	0.94	0.85	0.89	0.94	0.85	0.89	0.46	0.85	0.6	0.98	0.79	0.88
GloVe - Video Games	VG	0.94	0.85	0.89	0.94	0.85	0.89	0.46	0.85	0.6	0.98	0.79	0.87

Tabelle 13: Ergebnistabelle der GloVe-basierten Modelle. Zeilenweise sind die verschiedenen Vektorbildungsvarianten dargestellt. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierungsverfahren, gaußsche Naïve Bayes sowie Random Forests. Grundlage hierbei ist ein Teil der Video Games-Rezensionen als Testdatensatz. Die Ergebnistabelle für den Beauty-Test befindet sich im Anhang C

Zusammenfassend lässt sich sagen, dass auf den ersten Blick relativ gute Ergebnisse erreicht werden mit dem F1-Score zwischen 0.86 (0.34 mit gaußschem Naïve Bayes) und 0.91. Dabei stellt sich die logistische Regression mit den beiden Penalisiertechniken als das vergleichsweise bessere Klassifikationsverfahren heraus. Offenbar schneiden die

Precision-Werte bei allen Modellen, insbesondere beim Random Forests-Algorithmus relativ niedrig ab und ziehen somit den F1-Score herunter (mit der Ausnahme des gaußschen Naïve Bayes-Verfahrens). Dies deutet darauf hin, dass wegen der Überrepräsentiertheit von positiven Bewertung relativ viele negative Rezensionen fälschlicherweise als positiv eingeordnet werden, was zu verzerrten Ergebnissen führt.

5.2.1 Umgang mit unbalancierten Daten

Im Folgenden wird anhand eines der vorher dargestellten Random-Forests-Modelle das Problem der unausgewogenen Daten mit der überrepräsentierten positiven Bewertungskategorie veranschaulicht.

		Prädiktiert	
		Positiv	Negativ
Wahr	Positiv	34998	0
	Negativ	11358	0

Tabelle 14: Die Konfusionsmatrix als Gegenüberstellung zwischen den tatsächlichen und prädiktierten Klassenzugehörigkeiten. Prädiktionen auf Basis vom Random Forests-Klassifikationsmodell mit Merkmalsextrahierung BoW (Unigram)

In der Konfusionsmatrix (Tabelle 14) mit den absoluten Häufigkeiten wird deutlich, dass alle Rezensionen der Teststichprobe ausnahmslos als positiv klassifiziert werden. Wie in Tabelle 15 als Klassifikationsbericht dargelegt, deuten die Auswertungskennzahlen der positiven Klasse daher auf eine relativ hohe Modellgüte (mit einem Recall von 1.00 und Precision von 0.75) hin, ein genauer Blick auf die Kennzahlen der negativen Klasse beweist jedoch die Verzerrtheit der Prädiktionen. An dieser Stelle soll hervorgehoben werden, dass es sich bei Random Forests durch die Maximierung der Klassenanteile an den Baumknoten sowie durch das generelle Prinzip von Bagging um ein extremes Beispiel handelt (vgl. Abschnitt 4.3). Bei den restlichen Klassifikationsmodellen ist die Prädiktionsgüte der negativen Klasse dank der Beibehaltung der Gesamtverteilung (stratifizierte Aufteilung, vgl. Abschnitt 4.4) zwar niedriger, als die der Positiven, besteht jedoch nicht zwangsläufig ausschließlich aus 0, da hierbei auch das Erlernen der negativen Rezensionen gewährleistet wird. Weiterhin wurde bei einer detaillierteren Auswertung auffällig, dass die negativen Rezensionen besonders bei den Bigram-Modellen mit dem multinomialen Naïve Bayes-Verfahren eine verhältnismäßig geringe Prognosegüte aufweisen.

Bewertung	R	P	F1	Anzahl
Negativ	0.00	0.00	0.00	11358
Positiv	1.00	0.75	0.86	34998

Tabelle 15: Klassifikationsbericht des Random Forests-Verfahrens mit Merkmalsextrahierung BoW (Unigram): Auswertungskennzahlen für beide Bewertungskategorien

Um das Problem der unbalancierten Bewertungskategorien zu lösen und damit die Prädiktionen aller Klassifikationsmodelle zu verbessern, kommt das sogenannte *Undersampling* zum Einsatz. Dabei geht es darum, die Anzahl an Rezensionen von der

überrepräsentierten Klasse sowohl in Trainings-, als auch in Testdaten mittels Ziehung ohne Zurücklegen so lange zu reduzieren, bis beide Klassenrepräsentationen im Gleichgewicht stehen (Chawla, 2009). Für die Bildung der Dokumentrepräsentationen spielt dies keine Rolle, da der Undersampling-Prozess unmittelbar vor dem Einsatz der Klassifikationsmodelle angewendet wird und das feature extraction nicht beeinflusst. Insgesamt ergeben sich die folgenden Datenmengen: Trainingsdaten: 90.866; Testdaten (Teil des Video Games-Datensatzes): 22.716; Beauty-Datensatz (als zweiter Test): 88.460, wobei die positiven und die negativen Rezensionen jeweils die Hälfte umfassen.

Die unten stehenden Tabellen 18 und 19 weisen im Vergleich zum unausgewogenen Fall niedrigere, aber dennoch zufriedenstellende Prädiktionsgüte auf mit F1-Scores zwischen 0.59 (0.32 mit gaußschem Naïve Bayes) und 0.81. Bei diesen Tabellen werden nun zur besseren Übersichtlichkeit die Ergebnisse des Video Games-Tests und die des Beauty-Tests in jeweils einer Tabelle dargestellt. Nach wie vor entsprechen die farbigen Markierungen in hellblau (Video Games) und in hellrot (Beauty) dem jeweils besten F1-Score. Es ist zwar nachteilig, dass viele Rezensionen und die darin steckende Information beim Trainieren des Modells durch das Undersampling verloren gehen, jedoch führt das zu einer Lösung des eben erläuterten Problems und ermöglicht somit robustere Schätzungen. Dies lässt sich wieder an dem oben aufgeführten Beispiel des Random Forests-Modells mit BoW (Unigram) Methode belegen. Nun sehen die neue Konfusionsmatrix (Tabelle 16) und der neue Klassifikationsbericht (Tabelle 17) wie folgt aus:

		Prädiktiert	
		Positiv	Negativ
Wahr	Positiv	8740	2618
	Negativ	4652	6706

Tabelle 16: Die Konfusionsmatrix als Gegenüberstellung zwischen den tatsächlichen und prädiktierten Klassenzugehörigkeiten. Prädiktionen auf Basis vom Random Forests-Klassifikationsmodell mit Merkmalsextrahierung BoW (Unigram) nach dem Undersampling

Bewertung	R	P	F1	Anzahl
Negativ	0.59	0.72	0.65	11358
Positiv	0.77	0.65	0.71	11358

Tabelle 17: Klassifikationsbericht des Random Forests-Klassifikationsmodells mit Merkmalsextrahierung BoW (Unigram): Auswertungskennzahlen für beide Bewertungskategorien nach dem Undersampling

Die Prädiktionsgüte der positiven Klasse sinkt zwar vergleichsmäßig, offensichtlich erreicht man hierbei jedoch bessere Prädiktionen der negativen Rezensionen.

Eine nähere Betrachtung der dem Undersampling unterzogenen BoW-basierten Modelle (Tabelle 18) zeigt, dass auch hier die besten Ergebnisse durch die logistische Regression erreicht werden, bis auf die Ausnahme des TF-IDF (Bigram) Beauty-Tests, bei dem der F1-Score des Naïve Bayes mit 0.72 am höchsten ist. Zeilenweise unterscheiden sich die Ergebnisse der verschiedenen Merkmalsextrahierungen minimal, wobei der Beauty-Test bei allen Modellen schlechter abschneidet, wie beispielsweise beim TF-IDF

(Unigram) Modell mit einem F1-Score 0.81 (Video Games-Test, Ridge Regression) und 0.75 (Beauty-Test, Ridge Regression). Dies hängt möglicherweise damit zusammen, dass die neuen Wörter in den darin enthaltenen Rezensionen bei der Vektorbildung ignoriert werden. Zu berücksichtigen ist die Tatsache, dass das Trainieren des Modells auf Basis vom Video Games-Kontext erfolgte. Zudem ist die Differenz zwischen den Recall- und Precision-Werten bei den meisten Modellen nicht mehr so groß, wie im unbalancierten Fall.

			Ridge			Lasso			Naïve Bayes			Random Forests		
			R	P	F1	R	P	F1	R	P	F1	R	P	F1
BoW (Unigram)	VG	0.83	0.79	0.81	0.83	0.79	0.81	0.79	0.76	0.78	0.77	0.65	0.71	
	B	0.81	0.7	0.76	0.81	0.7	0.76	0.7	0.69	0.7	0.93	0.54	0.68	
BoW (Bigram)	VG	0.81	0.78	0.79	0.81	0.79	0.8	0.75	0.79	0.77	0.83	0.67	0.74	
	B	0.79	0.67	0.73	0.76	0.64	0.7	0.67	0.69	0.68	0.95	0.54	0.69	
TF-IDF (Unigram)	VG	0.82	0.81	0.81	0.82	0.81	0.81	0.76	0.77	0.76	0.75	0.65	0.7	
	B	0.79	0.72	0.75	0.82	0.71	0.76	0.69	0.65	0.67	0.91	0.55	0.68	
TF-IDF (Bigram)	VG	0.8	0.8	0.8	0.79	0.78	0.78	0.75	0.83	0.79	0.8	0.68	0.74	
	B	0.7	0.73	0.71	0.66	0.7	0.68	0.7	0.74	0.72	0.93	0.55	0.69	

Tabelle 18: Ergebnistabelle der BoW-basierten Modelle. Zeilenweise sind die verschiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierungsverfahren, multinomiale Naïve Bayes sowie Random Forests

In Tabelle 19 sind die Auswertungskennzahlen der GloVe-Modelle auf Basis von balancierten Video Games- sowie Beauty-Testdaten zusammengetragen.

			Ridge			Lasso			Naïve Bayes			Random Forests		
			R	P	F1	R	P	F1	R	P	F1	R	P	F1
GloVe - Wiki	VG	0.74	0.77	0.75	0.74	0.77	0.75	0.42	0.65	0.51	0.7	0.74	0.72	
	B	0.86	0.64	0.73	0.86	0.64	0.73	0.68	0.57	0.62	0.63	0.7	0.68	
GloVe - Wiki (gew.)	VG	0.55	0.76	0.64	0.58	0.76	0.66	0.22	0.64	0.32	0.65	0.69	0.67	
	B	0.68	0.69	0.68	0.73	0.67	0.7	0.32	0.54	0.4	0.5	0.71	0.59	
GloVe - Wiki, Video Games	VG	0.76	0.79	0.78	0.76	0.79	0.78	0.44	0.66	0.53	0.72	0.75	0.74	
	B	0.72	0.75	0.73	0.72	0.74	0.73	0.56	0.6	0.58	0.59	0.76	0.67	
GloVe - Video Games	VG	0.76	0.79	0.78	0.76	0.79	0.78	0.44	0.66	0.53	0.72	0.75	0.74	
	B	0.7	0.76	0.73	0.7	0.76	0.73	0.54	0.6	0.57	0.58	0.77	0.66	

Tabelle 19: Ergebnistabelle der GloVe-basierten Modelle. Zeilenweise sind die verschiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierungsverfahren, gaußsche Naïve Bayes sowie Random Forests

Auch hier gilt, dass die logistische Regression für nahezu alle Modelle die besten Prädiktionen liefert. Erwähnenswert ist, dass eine verschwindend geringe Penalisierung zu verhältnismäßig schlechteren Ergebnissen führt, wie in Tabelle 20 dargelegt.

		Logistische Regression		
		R	P	F1
GloVe -	VG	0.70	0.73	0.71
	B	0.83	0.61	0.7
GloVe -	VG	0.53	0.74	0.62
	B	0.59	0.64	0.61
GloVe - Wiki,	VG	0.69	0.73	0.71
	B	0.67	0.74	0.7
Video Games	VG	0.74	0.68	0.71
	B	0.66	0.7	0.68

Tabelle 20: Ergebnistabelle der GloVe-basierten Modelle. Zeilenweise sind die verschiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert wird das Klassifikationsverfahren der Logistischen Regression mit einer verschwindend geringen Penalisation (Ridge, $1/\lambda = 10^8$)

Im Allgemeinen sind die GloVe-Ergebnisse der Tabelle 19 tendenziell schlechter im Vergleich zur BoW-Tabelle 18. Dies lässt sich damit begründen, dass die hinter GloVe stehende Methodik zwar zweckmäßiger und erfolgversprechender ist, problematisch ist dennoch die Wahl des Korpus, worauf die Wortvektoren trainiert werden. Im Falle der basierend auf (unter anderem) Wikipedia-Texten vortrainierten Wortvektoren, ist der zugrundeliegende Korpus möglicherweise zu allgemein, während der Video Games-Datensatz ein zu spezifisches Vokabular liefert. Zudem handelt es sich bei der Zusammenfassung von Wortvektoren mithilfe des einfachen sowie gewichteten arithmetischen Mittels um einen heuristischen Ansatz. Die Gewichtung der vortrainierten Wortvektoren mit TF-IDF scheint tendenziell nicht hilfreich zu sein: Der durch das einfache Mittel erreichte F1-Score wird beispielsweise (beim Video Games-Test, Ridge) von 0.75 auf 0.64 reduziert. Bei der Übertragung der auf Video Games-Kontext trainierten Wortvektoren auf den Beauty-Kontext (GloVe - Video Games) weisen die Ergebnisse eine Reduktion der Prädiktionsgüte auf mit beispielsweise 0.78 zu 0.73, was auf ein Übertragungsproblem hindeutet kann. Dies gilt jedoch nicht für die Methode der gewichteten Zusammenfassung von Vektoren, GloVe - Wiki (gew.), was vermutlich mit dem zugrundeliegenden Wikipedia-Korpus zusammenhängt. Da aufgrund dessen Umfangs relativ allgemeine Vektoren geliefert werden, ist beim Testen nicht zwangsläufig ein Vorteil für Video Games zu erwarten. Die Heranziehung von vortrainierten Wortvektoren beim Fehlen eines Wortes im Video Games-Vokabular liefert eine annähernd gleiche Prädiktionsgüte (GloVe - Wiki, Video Games), wie die reine Berücksichtigung von Video Games-Wortvektoren. Auffällig ist, dass im Vergleich zu den anderen Klassifikationsverfahren das gaußsche Naïve Bayes die schlechtesten Ergebnisse zeigt. Vermutlich spielt hierbei die Annahme, dass die Kovariablen unabhängig sind, eine schwerwiegende Rolle. Letztlich ist diese Annahme insbesondere bei den GloVe-basierten Modellen verletzt, da es sich um Richtungsvektoren handelt, deren Einträge voneinander nicht unabhängig sind. Auch der Einsatz des Random Forests-Verfahrens muss in dieser Hinsicht möglicherweise in Zweifel gezogen werden, da die Wortvektoren als eine Einheit und nicht die Vektoreinträge unabhängig betrachtet werden sollten, wie bei der Konstruktion von Klassifikationsbäumen. Bei den BoW-basierten Modellen führt hingegen die separate Behandlung der Kovariablen, zumindest inhaltlich, zu keinem drastischen Problem, da es sich dabei um einzelne Wörter handelt, wobei manche vom Algorithmus möglicherweise als unwichtig identifiziert werden.

Wie bereits erwähnt, erfolgt die Normalisierung der Rezensionstexte für die GloVe-basierten Modelle prinzipiell unter Weglassung von Lemmatisierung sowie Entfernung von Stopwörtern (vgl. Abschnitt 5.1). Der Vollständigkeit halber seien in Tabelle 21 die Auswertungskennzahlen der Modelle dargestellt, bei denen die Rezensionen zusätzlich zur Tabelle 19 lemmatisiert und die Stopwörter entfernt wurden. Es wird deutlich, dass die Ergebnisse dadurch eher negativ beeinflusst werden. Ausgenommen von der letzteren Festlegung sind zum einen GloVe-Wiki (gew.), möglicherweise weil diese Teilschritte für die Bildung von TF-IDF-Gewichten eine Relevanz darstellen, und zum anderen für das gaußsche Naïve Bayes, allerdings ist die Beurteilung dieser wegen allgemein sehr niedriger Prädiktionsgüte schwierig. Dennoch ist der Effekt dieser Normalisierungsschritte gering, weshalb auch im Laufe der weiteren Analysen auf diese zwei Teilprozedere für GloVe verzichtet wird.

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
GloVe - Wiki	VG	0.73	0.75	0.74	0.73	0.75	0.74	0.47	0.64	0.54	0.56	0.73	0.63
GloVe - Wiki (gew.)	VG	0.71	0.72	0.71	0.57	0.73	0.64	0.29	0.62	0.39	0.66	0.68	0.67
GloVe - Wiki, Video Games	VG	0.77	0.78	0.77	0.77	0.78	0.78	0.51	0.66	0.57	0.74	0.75	0.75
GloVe - Video Games	VG	0.77	0.78	0.77	0.77	0.78	0.77	0.51	0.66	0.57	0.74	0.75	0.75

Tabelle 21: Ergebnistabelle der GloVe-basierten Modelle. Der Unterschied zur Tabelle 13 besteht darin, dass bei der Normalisierung die Rezensionstexte lemmatisiert und die Stopwörter entfernt wurden. Zeilenweise sind die verschiedenen Vektorbildungsvarianten dargestellt. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierungsverfahren, gaußsche Naïve Bayes sowie Random Forests. Grundlage hierbei ist ein Teil der Video Games-Rezensionen als Testdatensatz

Alles in allem lässt sich festhalten, dass die Ergebnisse der balancierten Modelle robuster sind, da auch für die negative Bewertungskategorie eine relativ hohe Prädiktionsgüte erreicht wird. Durch die Reduktion des Datenumfangs im Rahmen des Undersampling wird das Erlernen der Dokumentrepräsentationen hinsichtlich der positiven Bewertungen erschwert, sodass die Prädiktionen der positiven Kategorie nun größere Fehler aufweisen. Die logistische Regression liefert nach wie vor fast bei allen Modellen die besten Ergebnisse.

5.2.2 Kategorisierung von 3-Sterne-Bewertungen

Wie am Anfang dieses Kapitels erwähnt, ist die Zuordnung der 3-Sterne-Bewertungen in die positive bzw. negative Kategorie wegen ihrer subjektiven Natur als eine Zwischenkategorie nicht eindeutig. Bisher wurden diese als negativ behandelt. Jetzt wird die Frage aufgeworfen, ob und welche Veränderungen sich hinsichtlich der Prädiktionsgüte ergeben, wenn die 3-Sterne-Bewertungen zur positiven Klasse gruppiert werden. Diese Zuordnung erfolgt bereits bei der Datenaufbereitung und resultiert in einer noch mehr unausgewogenen Datensituation mit dem Umfang 28.516 bzw. 203.264 und 21.982 bzw. 176.520

als jeweils negative bzw. positive Bewertungskategorie vom Video Games- und Beauty-Datensatz. Um den im vorherigen Abschnitt illustrierten Problemen zu entgegnen, werden im Folgenden alle Modelle basierend auf den mittels Undersampling balancierten Daten aufgestellt. Zu beachten ist, dass der gesamte Datenumfang noch weiter reduziert wird.

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
BoW (Unigram)	VG	0.83	0.83	0.83	0.83	0.82	0.83	0.8	0.82	0.81	0.86	0.77	0.81
	B	0.82	0.74	0.78	0.82	0.74	0.78	0.71	0.75	0.73	0.9	0.63	0.74
BoW (Bigram)	VG	0.82	0.79	0.81	0.76	0.73	0.74	0.84	0.81	0.82	0.82	0.75	0.78
	B	0.79	0.68	0.73	0.73	0.7	0.71	0.7	0.75	0.73	0.78	0.63	0.7
TF-IDF (Unigram)	VG	0.84	0.84	0.84	0.84	0.84	0.84	0.79	0.83	0.81	0.84	0.77	0.8
	B	0.78	0.77	0.78	0.83	0.74	0.78	0.7	0.71	0.71	0.9	0.62	0.73
TF-IDF (Bigram)	VG	0.81	0.82	0.82	0.79	0.8	0.79	0.83	0.83	0.83	0.82	0.74	0.78
	B	0.68	0.75	0.71	0.62	0.73	0.67	0.73	0.75	0.74	0.79	0.62	0.7

Tabelle 22: Ergebnistabelle der BoW-basierten Modelle. Die 3-Sterne-Bewertungen werden zur positiven Kategorie eingeordnet. Zeilenweise sind die verschiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierungsverfahren, multinomiale Naïve Bayes sowie Random Forests

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
GloVe - Wiki	VG	0.78	0.8	0.79	0.78	0.8	0.79	0.44	0.67	0.53	0.73	0.78	0.75
	B	0.86	0.66	0.75	0.86	0.66	0.75	0.63	0.6	0.61	0.68	0.74	0.71
GloVe - Wiki (gew.)	VG	0.6	0.77	0.67	0.66	0.76	0.71	0.23	0.62	0.34	0.69	0.71	0.7
	B	0.71	0.7	0.71	0.78	0.67	0.72	0.35	0.53	0.42	0.56	0.71	0.63
GloVe - Wiki, Video Games	VG	0.8	0.82	0.81	0.8	0.82	0.81	0.47	0.69	0.56	0.76	0.79	0.77
	B	0.73	0.77	0.75	0.73	0.77	0.75	0.48	0.63	0.54	0.53	0.81	0.64
GloVe - Video Games	VG	0.8	0.82	0.81	0.8	0.82	0.81	0.47	0.69	0.56	0.76	0.79	0.77
	B	0.7	0.79	0.74	0.74	0.7	0.72	0.46	0.64	0.53	0.55	0.8	0.65

Tabelle 23: Ergebnistabelle der GloVe-basierten Modelle. Die 3-Sterne-Bewertungen werden zur positiven Kategorie eingeordnet. Zeilenweise sind die verschiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierungsverfahren, gaußsche Naïve Bayes sowie Random Forests

Die Einordnung der 3-Sterne-Bewertungen zur positiven Kategorie führt bei nahezu allen Modellen zu einer Verbesserung der Ergebnisse mit dem F1-Score zwischen 0.63 (0.34 mit gaußschem Naïve Bayes) und 0.84, wie den Tabellen 22 und 23 entnommen werden kann. Aus dieser Tatsache kann möglicherweise geschlossen werden, dass die Vergabe der 3-Sterne-Bewertung eher einer positiven Meinungsäußerung entspricht. Die Präzisionsgüte der Beauty-Rezensionen fällt nach wie vor nahezu bei allen Model-

len niedriger aus. Für die Bigram-Modelle gewinnt hier das multinomiale Naïve Bayes-Verfahren an Bedeutung mit der besten Prädiktionsgüte. Ansonsten erhält die logistische Regression wieder die Rolle des besten Klassifikators aufrecht.

Zusammenfassend lässt sich sagen, dass die Verwendung balancierter Daten, sowie die Kategorisierung der 3-Sterne Bewertungen zur positiven Kategorie, die Prädiktionsgüte positiv beeinflussen. Die F1-Scores liegen dann zwischen 0.63 (0.34 mit gaußschem Naïve Bayes) und 0.84. Die besten Prädiktionen werden für nahezu alle Modelle durch die penalisierte logistische Regression erreicht. Die BoW-basierten Modelle liefern mit einem kleinen Unterschied bessere Ergebnisse, als die GloVe-basierten Modelle. Vermutlich spielt hierbei die Wahl des Korpus eine große Rolle. Die besten Prädiktionen der Bigram-Modelle sind ein wenig niedriger, als die der zugehörigen Unigramme, was möglicherweise mit der sehr großen Dimension der Dokumentrepräsentationen und der Überanpassungsgefahr zusammenhängt. Zwischen verschiedenen Variationen für die GloVe-Modelle sind die Unterschiede minimal, bis auf die mit TF-IDF gewichtete Mittelbildung der numerischen Repräsentationen: Dabei sinkt die Prädiktionsgüte im Vergleich zum ungewichteten Fall, was für das einfache arithmetische Mittel als Zusammenführungsvariante der Wortvektoren spricht. Zudem führt die Übertragung des Modells auf den zweiten Testdatensatz mit den Rezensionen für Beauty-Artikel zu einer nahezu konstanten Verschlechterung der Ergebnisse. Von Ausnahmen ist meist das gaußsche Naïve Bayes getroffen. Da Letzteres aber generell eine sehr niedrige Prädiktionsgüte aufweist, die fast den Eindruck einer willkürlichen Klassifizierung erweckt, ist eine dahingehende Interpretation schwierig. Möglicher Grund dafür ist die Verletzung der Unabhängigkeitsannahme der Kovariablen für die GloVe-Modelle.

6 Fazit und Ausblick

Um einen Vergleich zwischen verschiedenen Klassifikationsverfahren zur Vorhersage der negativen bzw. positiven Bewertungskategorie auf Basis von einer gegebenen Kundenrezension zu ermöglichen, wurde nach einer anfänglichen Normalisierung zwei Hauptmethodiken zur Merkmalsextrahierung nachgegangen, damit die Rezensionstexte durch Zahlen repräsentiert werden können: zum einen erfolgte dies unter der alleinigen Betrachtung der Worthäufigkeiten mithilfe der Bag of Words basierten Modelle, zum anderen wurde auch der semantische Zusammenhang zwischen den Wörtern im Rahmen des Global Vectors Modells berücksichtigt. Während der erstere Modellierungsansatz auf einer vergleichsmäßig einfachen Idee fußt und meist trotzdem zufriedenstellende Ergebnisse liefert, erweist dieser den Nachteil, dass die Reihenfolge der Wörter ignoriert wird, und dadurch möglicherweise zwei Rezension gegensätzlichen Inhalts durch die gleichen Vektoren repräsentiert werden (vgl. Abschnitt 3.3). Der zweite Ansatz über GloVe basiert auf einem Algorithmus, bei dem alle im Korpus vorkommenden Wörter anhand ihrer Häufigkeiten und der sie umgebenden Wörter auf eine geeignete Weise in einem multidimensionalen Vektorraum platziert werden, sodass Wörter, die oft in einem ähnlichen Kontext vorkommen, eine ähnliche Vektordarstellung erhalten. Nachteilig ist hierbei die Auswahl bezüglich des Umfangs und der Qualität des Korpus zur Schätzung von solchen Wortvektoren. Als Machine Learning Verfahren zur Klassifikation kommen die logistische Regression mit Ridge- und Lasso-Penalisation, Naïve Bayes sowie Random Forests zum Einsatz.

Zusammenfassend kann festhalten werden, dass es durchaus sinnvoll ist, sowohl zum Trainieren, als auch zum Testen der Modelle einen auf balancierten Bewertungskatego-

rien basierenden Datensatz zu verwenden, wenn eine robuste Schätzung und eine zufriedenstellende Prädiktionsgüte für die beiden Kategorien zum Ziel gesetzt werden. Weiterhin führt die Behandlung der 3-Sterne-Bewertungen als positiv, zu einer nahezu konstanten Verbesserung der Modelle. Zum größten Teil werden die besten Prädiktionen durch die penalisierte logistische Regression erzielt, wobei die Ridge- und Lasso-Penalisiungsverfahren sehr ähnliche bis exakt gleiche Ergebnisse liefern. Zudem resultiert das gaußsche Naïve Bayes-Verfahren in vergleichsweise schlechten Prädiktionen. Die Art der Merkmalsextrahierung bringt im Allgemeinen minimale Unterschiede mit Vorteil für die BoW-basierten Modelle. Durch die Heranziehung der vtrainierten Wortvektoren kann kaum substanzialer Effekt gemessen werden. Die Übertragung der auf dem Referenzdatensatz trainierten Wortvektoren auf die Beauty-Rezensionen führt bei nahe überall zur Reduktion der Prädiktionsgüte. Deutlich wird insbesondere die Problematik der Wahl eines geeigneten Korpus zum Trainieren von GloVe-Vektoren. Der wohl deutlichste Unterschied, der im Hinblick auf die BoW-basierten Modelle beobachtet werden kann, ist die geringfügig schlechtere Prädiktion der Bigram-Modelle im Vergleich zu den Unigrammen. Dies lässt sich vermutlich durch die zu große Vektordimension und das damit verbundene Generalisierungsproblem der Bigram-Modelle erklären.

Ein sicherlich interessanter Aspekt für die weiteren Analysen ist die Betrachtung von *word2vec*, einem anderen im Jahr 2013 entwickelten Algorithmus zum Trainieren von Wortvektoren unter Berücksichtigung der Bedeutung von Wörtern, die in einem Korpus vorkommen. Dabei handelt es sich um ein Verfahren prädiktiver Natur, welches auf neuronalen Netzen basiert. Die grundlegende Idee ist, wie beim GloVe, aufgrund eines Korpus als Input, die Wortvektoren als Output zu erhalten. Nach der Aufstellung eines Vokabulars basierend auf dem Korpus, werden die Vektorrepräsentationen für die Wörter mittels CBOW und Skipgram erlernt. Durch verschiedene Techniken, wie einfaches oder mit TF-IDF gewichtetes Mittel können diese Wortvektoren zu einer Dokumentrepräsentation zusammengefasst werden. Idealerweise sollen auch hier die Distanzen zwischen den Wortvektoren semantisch sinnvolle Resultate erweisen. Für weitere Details sei auf Mikolov et al. (2013) verwiesen. Zudem kann es durchaus aufschlussreich sein, die Verarbeitung auf Buchstabenebene durchzuführen, sodass nun nicht mehr Wörter, sondern die einzelnen Buchstaben als Input zum Trainieren verwendet werden.

Abschließend kann hervorgehoben werden, dass die aus dieser Untersuchung gewonnenen Erkenntnisse durchaus interessant für die Entwicklung eines automatisierten Prozesses zur Vergabe der Bewertungskategorien sind. Die vertiefte Analyse von den im Rahmen des Natural Language Processing zur Verfügung gestellten Methodiken ist nicht nur ein sehr spannendes Thema, sondern kann auch zur Erleichterung der diversen Alltagsfacetten eingesetzt werden.

Literatur

- Amazon.com (2014). Eine vom Amazon-Nutzer lucas veröffentlichte Rezension. https://www.amazon.com/gp/customer-reviews/R1JJVXDHCM0H3U/ref=cm_cr_getr_d_rvw_ttl?ie=UTF8&ASIN=B00000DMB3, abgerufen am 10. Mai 2019.
- Bitkom e.V. (2017). 11.01.2017 - Kundenbewertungen sind wichtigste Kaufhilfe. <https://www.bitkom.org/Presse/Presseinformation/Kundenbewertungen-sind-wichtigste-Kaufhilfe.html>, abgerufen am 04. Mai 2019.
- Breiman, L. (2001). Random forests, *Machine learning*.
- Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). Classification and regression trees.
- Camacho-Collados, J. and Pilehvar, M. T. (2017). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis, *arXiv preprint arXiv:1707.01780*.
- Chan, T. F., Golub, G. H. and LeVeque, R. J. (1982). Updating formulae and a pairwise algorithm for computing sample variances, *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, Springer.
- Chawla, N. V. (2009). Data mining for imbalanced datasets: An overview, *Data mining and knowledge discovery handbook*, Springer.
- Chen, C., Liaw, A., Breiman, L. et al. (2004). Using random forest to learn imbalanced data, *University of California, Berkeley*.
- Ciresan, D., Giusti, A., Gambardella, L. M. and Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images.
- Defazio, A., Bach, F. and Lacoste-Julien, S. (2014). Saga: A fast incremental gradient method with support for non-strongly convex composite objectives.
- Fahrmeir, L., Heumann, C., Künstler, R., Pigeot, I. and Tutz, G. (2016). Statistik: Der Weg zur Datenanalyse, Springer-Verlag.
- Fahrmeir, L., Kneib, T., Lang, S. et al. (2007). Regression, Springer.
- Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. and Lin, C.-J. (2008). Liblinear: A library for large linear classification, *Journal of machine learning research*.
- Feldman, R. and Sanger, J. (2007). The text mining handbook: advanced approaches in analyzing unstructured data, Cambridge university press.
- Friedman, J., Hastie, T., Tibshirani, R. et al. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors), *The annals of statistics*.
- Hastie, T., Tibshirani, R., Friedman, J. et al. (2002). The elements of statistical learning.
- Heyer, G., Quasthoff, U. and Wittig, T. (2006). Text Mining: Wissensrohstoff Text: Konzepte, Algorithmen, Ergebnisse, W3L-Verlag Dortmund.

- James, G., Witten, D., Hastie, T. and Tibshirani, R. (2014). An introduction to statistical learning, Springer.
- Jurafsky, D. (2000). Speech & language processing, Pearson Education India.
- Kibriya, A. M., Frank, E., Pfahringer, B. and Holmes, G. (2004). Multinomial naive bayes for text categorization revisited, *Australasian Joint Conference on Artificial Intelligence*, Springer.
- Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval, *European conference on machine learning*, Springer.
- Manning, C., Raghavan, P. and Schütze, H. (2010). Introduction to information retrieval, *Natural Language Engineering*.
- McAuley, J., Targett, C., Shi, Q. and Van Den Hengel, A. (2015). Image-based recommendations on styles and substitutes, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, pp. 43–52.
- McAuley, Julian (2015). Amazon product data. jmcauley.ucsd.edu/data/amazon, abgerufen am 10. Mai 2019.
- Mccallum, A. and Nigam, K. (2001). A comparison of event models for naive bayes text classification, *Work Learn Text Categ*.
- Meyer, Jens (2002 - 2007). HTML-Entities. <http://unicode.e-workers.de/entities.php>, abgerufen am 10. Mai 2019.
- Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). Efficient estimation of word representations in vector space, *arXiv preprint arXiv:1301.3781*.
- Miner, G., Elder IV, J., Fast, A., Hill, T., Nisbet, R. and Delen, D. (2012). Practical text mining and statistical analysis for non-structured text data applications, Academic Press.
- Norvig, Peter (2007). How to Write a Spelling Corrector. <http://norvig.com/spell-correct.html>, abgerufen am 19. Mai 2019.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research*. <https://scikit-learn.org>, Python package version 0.20.1.
- Pennington, J., Socher, R. and Manning, C. (2014). Glove: Global vectors for word representation.
- Pennington, Jeffrey and Socher, Richard and Manning, Christopher (2014). Pre-trained word vectors. <https://nlp.stanford.edu/projects/glove>, abgerufen am 10. Mai 2019.
- Rezaeinia, S. M., Ghodsi, A. and Rahmani, R. (2017). Improving the accuracy of pre-trained word embeddings for sentiment analysis, *arXiv preprint arXiv:1711.08609*.
- Sarkar, D. (2016). Text analytics with python.

- Selivanov, D. and Wang, Q. (2018). text2vec: Modern Text Mining Framework for R. <https://CRAN.R-project.org/package=text2vec>, R package version 0.5.1.
- Sharafi, A., Wolf, P. and Krcmar, H. (2010). Knowledge discovery in databases on the example of engineering change management., *ICDM 2010*.
- Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval, *Journal of documentation* .
- Strobl, C., Malley, J. and Tutz, G. (2009). An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests., *Psychological methods* .
- Tutz, G. (2011). Regression for categorical data, Cambridge University Press.
- Van Rijsbergen, C. (1979). Information retrieval buttersmiths, *London* .

Anhang

A Quellcode: Funktionen

Nachfolgend werden Ausschnitte des Quellcodes präsentiert. Ausgenommen der Funktion 3 (R, Version Version 3.5.0), entstammen die restlichen Funktionen aus der Programmiersprache Python, Version 2.7.15.

```

from contractions import CONTRACTION_MAP # Abkuerzungsliste
import re
import nltk
nltk.download('punkt')
import string
from nltk.stem import WordNetLemmatizer
from HTMLParser import HTMLParser
import unicodedata
from pattern.en import suggest
import pattern
from pattern.en import tag
from nltk.corpus import wordnet as wn

stopword_list = nltk.corpus.stopwords.words('english')
stopword_list = stopword_list + ['mr', 'mrs', 'come', 'go', 'get',
        ,
        'tell', 'listen', 'one', 'two', 'three',
        'four', 'five', 'six', 'seven', 'eight',
        'nine', 'zero', 'join', 'find', 'make',
        'say', 'ask', 'tell', 'see', 'try',
        'back',
        'also']

wnl = WordNetLemmatizer()
html_parser = HTMLParser()

def tokenize_text(text):
    tokens = nltk.word_tokenize(text)
    tokens = [token.strip() for token in tokens]
    return tokens

def expand_contractions(text, contraction_mapping):

    contractions_pattern = re.compile('({})'.format('|'.join(
        contraction_mapping.keys())),
        flags=re.IGNORECASE|re.DOTALL)

    def expand_match(contraction):
        match = contraction.group(0)
        first_char = match[0]
        expanded_contraction = contraction_mapping.get(match) \
            if contraction_mapping.get(match) \
            else contraction_mapping.get(match.lower())
        return expanded_contraction
    return contractions_pattern.sub(expand_match, text)

```

```

expanded_contraction = first_char+expanded_contraction[1:]
return expanded_contraction

expanded_text = contractions_pattern.sub(expand_match, text)
expanded_text = re.sub("''", "", expanded_text)
return expanded_text

def pos_tag_text(text):

    def penn_to_wn_tags(pos_tag):
        if pos_tag.startswith('J'):
            return wn.ADJ
        elif pos_tag.startswith('V'):
            return wn.VERB
        elif pos_tag.startswith('N'):
            return wn.NOUN
        elif pos_tag.startswith('R'):
            return wn.ADV
        else:
            return None

    tagged_text = tag(text)
    tagged_lower_text = [(word.lower(), penn_to_wn_tags(pos_tag))
                         for word, pos_tag in
                         tagged_text]
    return tagged_lower_text

def lemmatize_text(text):

    pos_tagged_text = pos_tag_text(text)
    lemmatized_tokens = [wnl.lemmatize(word, pos_tag) if pos_tag
                         else word
                         for word, pos_tag in pos_tagged_text]
    lemmatized_text = '_'.join(lemmatized_tokens)
    return lemmatized_text

def remove_special_characters(text):
    tokens = tokenize_text(text)
    pattern = re.compile('[' + string.punctuation + ']')
    filtered_tokens = filter(None, [pattern.sub('_', token) for
                                    token in tokens])
    filtered_text = '_'.join(filtered_tokens)
    return filtered_text

def remove_stopwords(text):
    tokens = tokenize_text(text)
    filtered_tokens = [token for token in tokens if token not in
                      stopword_list]
    filtered_text = '_'.join(filtered_tokens)
    return filtered_text

```

```

def keep_text_characters(text):
    filtered_tokens = []
    tokens = tokenize_text(text)
    for token in tokens:
        if re.search('[a-zA-Z]', token):
            filtered_tokens.append(token)
    filtered_text = '_'.join(filtered_tokens)
    return filtered_text

def unescape_html(parser, text):

    return parser.unescape(text)

from HTMLParser import HTMLParser

class MLStripper(HTMLParser):
    def __init__(self):
        self.reset()
        self.fed = []
    def handle_data(self, d):
        self.fed.append(d)
    def get_data(self):
        return '_'.join(self.fed)

def strip_html(text):
    html_strippe = MLStripper()
    html_strippe.feed(text)
    return html_strippe.get_data()

def remove_repeation(text):
    pattern = re.compile(r"(.)\1{2,}")
    return pattern.sub(r"\1\1", text)

def spell_corrector(text):
    corrected_tokens = []
    tokens = tokenize_text(text)
    for token in tokens:
        remove_repeation(token)
        suggestions = suggest(token)
        strings, numbers = zip(*suggestions)
        correct_token = strings[0]
        corrected_tokens.append(correct_token)
    corrected_text = '_'.join(corrected_tokens)
    return corrected_text

def normalize_corpus(corpus, lemmatize=True,
                      only_text_chars=False,
                      tokenize=False,
                      stopwords_removal = True,
                      lower_case = False, spell_correct = False):

```

```

normalized_corpus = []
for index, text in enumerate(corpus):

    text = html_parser.unescape(text)
    text = strip_html(text)
    text = expand_contractions(text, CONTRACTION_MAP)
    if lemmatize:
        text = lemmatize_text(text)
    if lower_case:
        text = text.lower()
    text = remove_special_characters(text)
    if spell_correct:
        text = spell_corrector(text)
    if stopwords_removal:
        text = remove_stopwords(text)
    if only_text_chars:
        text = keep_text_characters(text)
    if tokenize:
        text = tokenize_text(text)
        normalized_corpus.append(text)
    else:
        normalized_corpus.append(text)
return normalized_corpus

```

Funktion 1: Normalisierung

```

from sklearn.feature_extraction.text import CountVectorizer,
    TfidfVectorizer
import pandas as pd
def build_feature_matrix(documents, feature_type='frequency',
    ngram_range=(1, 1), min_df=0.0, max_df=1.0):

    feature_type = feature_type.lower().strip()

    if feature_type == 'binary':
        vectorizer = CountVectorizer(binary=True, min_df=min_df,
            max_df=max_df, ngram_range=ngram_range, token_pattern=u
            "(?u)\\b\\\\w+\\b")
    elif feature_type == 'frequency':
        vectorizer = CountVectorizer(binary=False, min_df=min_df,
            max_df=max_df, ngram_range=ngram_range, token_pattern=u
            "(?u)\\b\\\\w+\\b")
    elif feature_type == 'tfidf':
        vectorizer = TfidfVectorizer(min_df=min_df, max_df=max_df,
            ngram_range=ngram_range, token_pattern=u"(?u)\\b\\\\w+\\b")
    else:
        raise Exception("Wrong_feature_type_entered._Possible_
            values:_'binary',_'_frequency',_'_tfidf' ")

```

```
feature_matrix = vectorizer.fit_transform(documents).astype(  
    float)  
  
return vectorizer, feature_matrix
```

Funktion 2: Merkmalsextrahierung: BoW, TF-IDF

```

1 library(tidyverse)
2 library(jsonlite)
3 library(checkmate)
4 library(backports)
5 library(data.table)
6 library(text2vec)
7
8
9 ##' Oeffne .json-Datei
10##' @param json_file Bezeichnung von .json-Datei mit Endung
11
12 import_json <- function(json_file) {
13   assert_true(endsWith(json_file, ".json"))
14   loaded_json <- stream_in(file(json_file))
15   datatable <- data.table(loaded_json)
16   return(datatable)
17 }
18
19
20##' Trainiere GloVe-Wortvektoren
21##' @param input_reviews data.table / data.frame mit Daten (
22# Rezensionen)
23##' @param data_source "normalized" oder "raw" (Sind die
24# Rezensionen normalisiert?)
25##' @param bigram TRUE or FALSE
26##' @param output String (Wie soll die Output-Datei benannt
27# werden?)
28
29 extract_glove <- function(input_reviews, data_source = "
30   normalized", bigram = FALSE, output){
31
32   if(data_source == "normalized") {
33     data <- read_delim(input_reviews, "\t", escape_double = FALSE
34
35           ,
36           trim_ws = TRUE)
37
38     # Tokenisierung
39     tokens <- space_tokenizer(data$reviewText_normalized)
40   } else {
41     data <- import_json(input_reviews)
42
43     # Tokenisierung
44     tokens <- space_tokenizer(data$reviewText)
45   }
46
47
48   it <- itoken(tokens, progressbar = FALSE)
49   vocab <- create_vocabulary(it)

```

```

40
41 vocab <- prune_vocabulary(vocab, term_count_min = 5L)
42
43 vectorizer <- vocab_vectorizer(vocab)
44 # Bilde gewichtete co-occurrence Matrix
45 tcm <- create_tcm(it, vectorizer, skip_grams_window = 5L)
46 # Konstruiert das GloVe-Modell
47 glove <- GlobalVectors$new(word_vectors_size = 200, vocabulary
48   = vocab, x_max = 100)
49
50 wv_main <- glove$fit_transform(tcm, n_iter = 20)
51
52 wv_context <- glove$components
53
54 word_vectors <- wv_main + t(wv_context)
55
56 file_name <- paste(output, "glove.txt", sep = "_")
57
58 write.table(word_vectors, file_name, append = FALSE, sep = " ",
59   dec = ".",
60   row.names = TRUE, col.names = FALSE, quote=FALSE)
61
62 }
```

Funktion 3: Merkmalsextrahierung: GloVe

```

from sklearn.metrics import classification_report, make_scorer,
accuracy_score, precision_score, recall_score, f1_score
from sklearn import metrics
import numpy as np
import pandas as pd

def report(y_true, y_pred):
    print classification_report(y_true, y_pred)
    return accuracy_score(y_true, y_pred)

def show_metrics(true_labels, predicted_labels, positive_class
=1):
    content = {'Accuracy': [np.round(metrics.accuracy_score(
        true_labels, predicted_labels),2)],
    'Precision': [np.round(metrics.precision_score(
        true_labels, predicted_labels, pos_label=
        positive_class,average='binary'),2)],
    'Recall': [np.round(metrics.recall_score(true_labels,
        predicted_labels, pos_label=positive_class,
        average='binary'), 2)],
    'F1_Score': [np.round(metrics.f1_score(true_labels,
        predicted_labels, pos_label=positive_class,average
        ='binary'),2)]}

    content_df = pd.DataFrame(data = content, columns=['Accuracy',
    'Precision', 'Recall', 'F1_Score'])
```

```
    , 'Precision', 'Recall', 'F1_Score'])
return content_df

def display_confusion_matrix(true_labels, predicted_labels,
classes=[1,0]):

    cm = metrics.confusion_matrix(y_true=true_labels, y_pred=
        predicted_labels, labels=classes)
    cm_frame = pd.DataFrame(data=cm,
        columns=pd.MultiIndex(levels=[[ 'Predicted:' ],
            classes], labels=[[0,0],[0,1]]),
        index=pd.MultiIndex(levels=[[ 'Actual:' ],
            classes], labels=[[0,0],[0,1]]))
return cm_frame

def model_assessment(final_model, test_features, test_sentiments
, filename): # test_sentiments, filename

    pred_sentiments = final_model.predict(test_features)

    evaluation_metrics = show_metrics(true_labels=test_sentiments
        , predicted_labels=pred_sentiments, positive_class=1)
    class_report = classification_report(y_true=test_sentiments,
        y_pred=pred_sentiments)
    conf_matrix = display_confusion_matrix(true_labels=
        test_sentiments, predicted_labels=pred_sentiments)

return evaluation_metrics conf_matrix
```

Funktion 4: Modellwahl und Evaluation

B Wahl der Modellparameter

In diesem Abschnitt werden die default- sowie die optimierten Parameter aller Modelle, deren Ergebnisse im Rahmen dieser Arbeit gezeigt werden, tabellarisch dargestellt.,

Ridge:	class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, multi_class='warn', n_jobs=None, penalty='l2', random_state=None, tol=0.0001, verbose=0, warm_start=False
Lasso:	class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, multi_class='warn', n_jobs=None, penalty='l1', random_state=None, tol=0.0001, verbose=0, warm_start=False
M. Naïve Bayes:	class_prior=None, fit_prior=True
G. Naïve Bayes:	priors=None, var_smoothing=1e-09
Random Forests:	bootstrap=True, class_weight=None, criterion='gini', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False

Tabelle 24: Für alle Exrahierungsmodelle gleichbleibende Parameter

Exrahierung	Klass.modell	Parameter
GloVe - Wiki	Ridge	C = 5, max_iter = 140, solver = saga
GloVe - Wiki	Lasso	C = 4, max_iter = 140, solver = saga
GloVe - Wiki	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - Wiki(gew.)	Ridge	C = 5, max_iter = 140, solver = saga
GloVe - Wiki(gew.)	Lasso	C = 5, max_iter = 120, solver = liblinear
GloVe - Wiki(gew.)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - Wiki, VG	Ridge	C = 2, max_iter = 100, solver = saga
GloVe - Wiki, VG	Lasso	C = 1, max_iter = 120, solver = liblinear
GloVe - Wiki, VG	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - VG	Ridge	C = 3, max_iter = 100, solver = saga
GloVe - VG	Lasso	C = 5, max_iter = 120, solver = saga
GloVe - VG	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12

Tabelle 25: Wahl der Modellparameter im Falle balancierter Daten, wobei die Rezensionen beim Normalisierungsschritt zusätzlich lemmatisiert und die Stoppwörter entfernt wurden

Exrahierung	Klass.modell	Parameter
BoW (Unigram)	Ridge	C = 5, max_iter = 140, solver = saga
BoW (Unigram)	Lasso	C=5,max_iter=140,solver= saga
BoW (Unigram)	M. Naïve Bayes	alpha = 2.0
BoW (Unigram)	Random Forests	max_features=sqrt,n_estimators=200, max_depth=12
BoW (Bigram)	Ridge	C = 1, max_iter = 100, solver = liblinear
BoW (Bigram)	Lasso	C = 1, max_iter = 120, solver = liblinear
BoW (Bigram)	M. Naïve Bayes	alpha = 1.0
BoW (Bigram)	Random Forests	max_features=sqrt,n_estimators=100, max_depth=8
TF-IDF (Unigram)	Ridge	C = 2, max_iter = 100, solver = saga
TF-IDF (Unigram)	Lasso	C = 1, max_iter = 110, solver = liblinear
TF-IDF (Unigram)	M. Naïve Bayes	alpha = 0.1
TF-IDF (Unigram)	Random Forests	max_features=sqrt,n_estimators=200, max_depth=12
TF-IDF (Bigram)	Ridge	C = 5, max_iter = 120, solver = saga
TF-IDF (Bigram)	Lasso	C = 4, max_iter = 110, solver = saga
TF-IDF (Bigram)	M. Naïve Bayes	alpha = 0.1
TF-IDF (Bigram)	Random Forests	max_features=sqrt,n_estimators=100, max_depth=8
GloVe - Wiki	Ridge	C = 5, max_iter = 100, solver = liblinear
GloVe - Wiki	Lasso	C = 5, max_iter = 120, solver = liblinear
GloVe - Wiki	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - Wiki(gew.)	Ridge	C = 5, max_iter = 120, solver = liblinear
GloVe - Wiki(gew.)	Lasso	C = 4, max_iter = 100, solver = liblinear
GloVe - Wiki(gew.)	Random Forests	max_features=sqrt,n_estimators=200, max_depth=12
GloVe - Wiki, VG	Ridge	C = 4, max_iter = 100, solver = saga
GloVe - Wiki, VG	Lasso	C = 5, max_iter = 140, solver = liblinear
GloVe - Wiki, VG	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - VG	Ridge	C = 3, max_iter = 130, solver = saga
GloVe - VG	Lasso	C = 5, max_iter = 120, solver = liblinear
GloVe - VG	Random Forests	max_features=sqrt,n_estimators=100, max_depth=12

Tabelle 26: Wahl der Modellparameter im Falle unbalancierter Daten

Exrahierung	Klass.modell	Parameter
BoW (Unigram)	Ridge	C = 3, max_iter = 140, solver = saga
BoW (Unigram)	Lasso	C = 2, max_iter = 140, solver = saga
BoW (Unigram)	M. Naïve Bayes	alpha = 0.5
BoW (Unigram)	Random Forests	max_features=sqrt,n_estimators=200, max_depth=12
BoW (Bigram)	Ridge	C = 1, max_iter = 100, solver = liblinear
BoW (Bigram)	Lasso	C = 5, max_iter = 140, solver = saga
BoW (Bigram)	M. Naïve Bayes	alpha = 0.5
BoW (Bigram)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
TF-IDF (Unigram)	Ridge	C = 2, max_iter = 100, solver = saga
TF-IDF (Unigram)	Lasso	C = 1, max_iter = 120, solver = liblinear
TF-IDF (Unigram)	M. Naïve Bayes	alpha = 0.1
TF-IDF (Unigram)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
TF-IDF (Bigram)	Ridge	C = 5, max_iter = 110, solver = saga
TF-IDF (Bigram)	Lasso	C = 5, max_iter = 120, solver = saga
TF-IDF (Bigram)	M. Naïve Bayes	alpha = 1.0
TF-IDF (Bigram)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - Wiki	Ridge	C = 5, max_iter = 100, solver = liblinear
GloVe - Wiki	Lasso	C = 5, max_iter = 120, solver = liblinear
GloVe - Wiki	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - Wiki(gew.)	Ridge	C = 5, max_iter = 100, solver = liblinear
GloVe - Wiki(gew.)	Lasso	C = 5, max_iter = 120, solver = liblinear
GloVe - Wiki(gew.)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - Wiki, VG	Ridge	C = 5, max_iter = 100, solver = liblinear
GloVe - Wiki, VG	Lasso	C = 5, max_iter = 120, solver = saga
GloVe - Wiki, VG	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - VG	Ridge	C = 5, max_iter = 100, solver = liblinear
GloVe - VG	Lasso	C = 5, max_iter = 140, solver = liblinear
GloVe - VG	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12

Tabelle 27: Wahl der Modellparameter im Falle balancierter Daten

Exrahierung	Klass.modell	Parameter
BoW (Unigram)	Ridge	C = 4, max_iter = 130, solver = saga
BoW (Unigram)	Lasso	C = 5, max_iter = 140, solver = saga
BoW (Unigram)	M. Naïve Bayes	alpha = 10
BoW (Unigram)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
BoW (Bigram)	Ridge	C = 1, max_iter = 100, solver = liblinear
BoW (Bigram)	Lasso	C = 5, max_iter = 140, solver = saga
BoW (Bigram)	M. Naïve Bayes	alpha = 10
BoW (Bigram)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
TF-IDF (Unigram)	Ridge	C = 1, max_iter = 140, solver = saga
TF-IDF (Unigram)	Lasso	C = 1, max_iter = 140, solver = liblinear
TF-IDF (Unigram)	M. Naïve Bayes	alpha = 0.5
TF-IDF (Unigram)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
TF-IDF (Bigram)	Ridge	C = 5, max_iter = 110, solver = saga
TF-IDF (Bigram)	Lasso	C = 5, max_iter = 120, solver = saga
TF-IDF (Bigram)	M. Naïve Bayes	alpha = 1.0
TF-IDF (Bigram)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - Wiki	Ridge	C = 5, max_iter = 100, solver = liblinear
GloVe - Wiki	Lasso	C = 5, max_iter = 140, solver = liblinear
GloVe - Wiki	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - Wiki(gew.)	Ridge	C = 5, max_iter = 100, solver = liblinear
GloVe - Wiki(gew.)	Lasso	C = 5, max_iter = 120, solver = liblinear
GloVe - Wiki(gew.)	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - Wiki, VG	Ridge	C = 5, max_iter = 120, solver = saga
GloVe - Wiki, VG	Lasso	C = 4, max_iter = 140, solver = liblinear
GloVe - Wiki, VG	Random Forests	max_features=sqrt,n_estimators=500, max_depth=12
GloVe - VG	Ridge	C = 4, max_iter = 110, solver = saga
GloVe - VG	Lasso	C = 5, max_iter = 120, solver = liblinear
GloVe - VG	Random Forests	max_features=sqrt,n_estimators=200, max_depth=12

Tabelle 28: Wahl der Modellparameter im Falle balancierter Daten, wobei die 3-Sterne-Bewertungen zur positiven Kategorie eingeordnet werden

C Kreuzvalidierungs- und Generalisierungsfehler

Hierbei kommen alle für die vorliegende Arbeit relevanten und vorgestellten Modelle in Frage. Im Folgenden werden jeweils die Kreuzvalidierungsfehler dokumentiert. Für alle Tabellen gilt: Die Abkürzungen R, P und F1 stehen für die Auswertungskennzahlen Recall, Precision und F1-Score. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso-Penalisierte Verfahren, Naïve Bayes sowie Random Forests. Für die BoW-basierten Modelle kommt das multinomiale und für die GloVe-basierten Modelle das gaußverteilte Naïve Bayes Verfahren zum Einsatz. Grundlage für das Trainieren ist ein Teil der Video Games-Rezensionen. Zeilenweise sind die BoW-beziehungsweise TF-IDF-Modelle mit jeweils Uni- und Bigram-Betrachtung dargestellt. In anderen Tabellen sind zeilenweise die verschiedenen Vektorbildungsvarianten für GloVe zusammengetragen. Der höchste F1-Score des jeweiligen Merkmalsextrahierungsvariante wird markiert.

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
BoW (Unigram)	VG	0.95	0.87	0.91	0.95	0.87	0.91	0.94	0.85	0.9	1.0	0.75	0.86
BoW (Bigram)	VG	0.95	0.86	0.9	0.93	0.86	0.89	0.94	0.83	0.88	1.0	0.75	0.86
TF-IDF (Unigram)	VG	0.95	0.88	0.91	0.95	0.88	0.91	0.99	0.8	0.88	1.0	0.75	0.86
TF-IDF (Bigram)	VG	0.97	0.84	0.91	0.95	0.87	0.9	0.97	0.8	0.88	1.0	0.75	0.86

Tabelle 29: Mittlerer Fehler von der stratifizierten Kreuzvalidierung der BoW-basierten Modelle

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
GloVe - Wiki	VG	0.94	0.83	0.88	0.94	0.83	0.88	0.44	0.85	0.58	0.99	0.78	0.87
GloVe - Wiki (gew.)	VG	0.99	0.77	0.86	0.98	0.77	0.86	0.22	0.85	0.34	1.0	0.76	0.86
GloVe - Wiki, Video Games	VG	0.94	0.85	0.89	0.94	0.85	0.89	0.46	0.85	0.6	0.99	0.79	0.88
GloVe - Video Games	VG	0.94	0.85	0.89	0.94	0.85	0.89	0.46	0.85	0.6	0.99	0.79	0.88

Tabelle 30: Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-basierten Modelle

			Ridge			Lasso			Naïve Bayes			Random Forests		
			R	P	F1	R	P	F1	R	P	F1	R	P	F1
BoW (Unigram)	VG	0.83	0.79	0.81	0.83	0.79	0.81	0.8	0.76	0.78	0.77	0.66	0.71	
BoW (Bigram)	VG	0.81	0.78	0.79	0.81	0.79	0.8	0.78	0.78	0.78	0.83	0.67	0.74	
TF-IDF (Unigram)	VG	0.82	0.81	0.82	0.82	0.82	0.82	0.77	0.77	0.77	0.75	0.66	0.7	
TF-IDF (Bigram)	VG	0.8	0.8	0.8	0.79	0.78	0.78	0.76	0.82	0.79	0.8	0.68	0.74	

Tabelle 31: Mittlerer Fehler von der stratifizierten Kreuzvalidierung der BoW-basierten Modelle. Grundlage hierbei sind die Trainingsdaten, die dem Downsampling unterzogen wurden

			Ridge			Lasso			Naïve Bayes			Random Forests		
			R	P	F1	R	P	F1	R	P	F1	R	P	F1
GloVe - Wiki	VG	0.74	0.77	0.75	0.74	0.76	0.75	0.42	0.65	0.51	0.7	0.73	0.71	
GloVe - Wiki (gew.)	VG	0.55	0.76	0.64	0.57	0.75	0.65	0.22	0.64	0.32	0.65	0.69	0.67	
GloVe - Wiki, Video Games	VG	0.76	0.79	0.78	0.76	0.79	0.78	0.44	0.66	0.53	0.72	0.75	0.74	
GloVe - Video Games	VG	0.76	0.79	0.78	0.76	0.79	0.78	0.44	0.66	0.53	0.72	0.75	0.74	

Tabelle 32: Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-Modelle. Grundlage hierbei sind die Trainingsdaten, die dem Downsampling unterzogen wurden

Logistische Regression				
		R	P	F1
GloVe - Wiki	VG	0.71	0.73	0.72
GloVe - Wiki (gew.)	VG	0.53	0.75	0.62
GloVe - Wiki, Video Games	VG	0.69	0.73	0.71
GloVe - Video Games	VG	0.74	0.68	0.71

Tabelle 33: Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-basierten Modelle. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert wird das Klassifikationsverfahren der Logistischen Regression mit einer verschwindend geringen Penalisierung

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
BoW (Unigram)	VG	0.84	0.83	0.83	0.84	0.83	0.83	0.8	0.82	0.81	0.86	0.77	0.81
BoW (Bigram)	VG	0.82	0.79	0.81	0.81	0.8	0.8	0.84	0.81	0.82	0.76	0.76	0.76
TF-IDF (Unigram)	VG	0.84	0.84	0.84	0.84	0.84	0.84	0.79	0.83	0.81	0.84	0.77	0.8
TF-IDF (Bigram)	VG	0.81	0.82	0.82	0.79	0.8	0.79	0.83	0.83	0.83	0.81	0.75	0.78

Tabelle 34: Mittlerer Fehler von der stratifizierten Kreuzvalidierung der BoW-basierten Modelle. Die 3-Sterne-Bewertungen werden in die positive Kategorie eingeordnet. Trainingsdaten wurden dem Undersampling unterzogen

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
GloVe - Wiki	VG	0.78	0.8	0.79	0.78	0.8	0.79	0.44	0.67	0.53	0.73	0.78	0.75
GloVe - Wiki (gew.)	VG	0.6	0.77	0.67	0.66	0.76	0.71	0.25	0.62	0.36	0.69	0.71	0.7
GloVe - Wiki, Video Games	VG	0.81	0.82	0.81	0.81	0.82	0.81	0.47	0.69	0.56	0.76	0.79	0.77
GloVe - Video Games	VG	0.81	0.82	0.81	0.81	0.82	0.81	0.47	0.69	0.56	0.76	0.79	0.77

Tabelle 35: Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-basierten Modelle. Die 3-Sterne-Bewertungen werden in die positive Kategorie eingeordnet. Trainingsdaten wurden dem Undersampling unterzogen

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
GloVe - Wiki	B	0.73	0.75	0.74	0.73	0.75	0.74	0.47	0.64	0.54	0.71	0.71	0.71
GloVe - Wiki (gew.)	B	0.56	0.72	0.63	0.56	0.72	0.64	0.29	0.61	0.23	0.67	0.68	0.67
GloVe - Wiki, Video Games	B	0.77	0.78	0.78	0.77	0.78	0.78	0.51	0.66	0.58	0.75	0.75	0.75
GloVe - Video Games	B	0.77	0.78	0.78	0.77	0.78	0.78	0.51	0.66	0.57	0.75	0.75	0.75

Tabelle 36: Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-basierten Modelle. Beim Normalisierungsschritt wurden die Rezensionstexte lemmatisiert und die Stoppwörter entfernt wurden. Trainingsdaten wurden dem Undersampling unterzogen

Abschließend die Tabellen zu den Generalisierungsfehlern für den Beauty-Test im unbalancierten Fall.

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
BoW (Unigram)	B	0.96	0.83	0.89	0.96	0.83	0.89	0.9	0.84	0.9	1.0	0.78	0.87
BoW (Bigram)	B	0.97	0.82	0.89	0.95	0.82	0.88	0.96	0.81	0.88	1.0	0.78	0.87
TF-IDF (Unigram)	B	0.96	0.84	0.89	0.96	0.84	0.89	0.99	0.79	0.88	1.0	0.78	0.87
TF-IDF (Bigram)	B	0.97	0.82	0.89	0.94	0.83	0.88	0.99	0.79	0.88	1.0	0.78	0.87

Tabelle 37: Ergebnistabelle der BoW-basierten Modelle. Im Gegensatz zur Tabelle 12 fungiert hierbei der Beauty-Datensatz als Datengrundlage. Zeilenweise sind die BoW- bzw. TF-IDF-Modelle mit jeweils Uni- und Bigram-Betrachtung dargestellt. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierte Verfahren, multinomiale Naïve Bayes sowie Random Forests

		Ridge			Lasso			Naïve Bayes			Random Forests		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
GloVe - Wiki	B	0.96	0.81	0.88	0.96	0.81	0.88	0.7	0.82	0.76	0.78	0.78	0.88
GloVe - Wiki (gew.)	B	0.98	0.8	0.88	0.97	0.8	0.88	0.32	0.81	0.45	1.0	0.78	0.88
GloVe - Wiki, Video Games	B	0.92	0.85	0.88	0.92	0.85	0.88	0.59	0.83	0.69	0.98	0.79	0.88
GloVe - Video Games	B	0.91	0.85	0.88	0.91	0.85	0.88	0.56	0.83	0.67	0.98	0.79	0.87

Tabelle 38: Ergebnistabelle der GloVe-basierten Modelle. Im Gegensatz zur Tabelle 13 fungiert hierbei der Beauty-Datensatz als Datengrundlage. Zeilenweise sind die BoW- bzw. TF-IDF-Modelle mit jeweils Uni- und Bigram-Betrachtung dargestellt. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierte Verfahren, gaußsche Naïve Bayes sowie Random Forests

Abbildungsverzeichnis

1	Beispiel für eine im Datensatz Video Games enthaltene Kundenrezension auf Amazon.com (2014)	5
2	Wordclouds: Die häufigsten Wörter in den Kundenrezensionen des jeweiligen Datensatzes	6
3	Prozentuale Häufigkeitsverteilung der Bewertungssterne	6
4	Logistisches Regressionsmodell basierend auf einem Beispieldatensatz. Aufgrund einer metrischen Einflussgröße auf der x-Achse, wird die Wahrscheinlichkeit geschätzt (y-Achse), mit der ein Datenpunkt in die fiktive Klasse 1 eingeordnet wird. Die horizontale Linie stellt dabei die Schwellle dar. Für die orangen Datenpunkte wird die Klassenzugehörigkeit 1, für die hellblauen Datenpunkte die Klasse 0 vorhergesagt	17
5	Beispiel für einen Klassifikationsbaum. Der Wurzel- und der innere Knoten werden als gelbe Kreise, und die Endknoten als grüne Quadrate dargestellt	23
6	Qualitative Darstellung der disjunkten Aufteilung vom Gesamtdatensatz zur Validierung und Beurteilung der Modellgüte	25
7	Prozentuale Häufigkeitsverteilung der Bewertungskategorien. Hierbei werden 1-, 2-, 3-Sterne-Bewertungen zur negativen und 4-, 5- zur positiven Kategorie zusammengefasst	28
8	Schematischer Aufbau des Anwendungsprozesses. Dabei wird ein Teil des Video Games-Datensatzes zur Parameteroptimierung und zum Trainieren verwendet. Der restliche Teildatensatz, sowie der Beauty-Datensatz kommen als Testdaten zur Beurteilung der Prädiktionsgüte zum Einsatz .	31

Tabellenverzeichnis

1	Übersicht der Variablen in den verwendeten Datensätzen	4
2	Absolute Häufigkeitstabelle für jedes Wort, das in einem Dokument vor kommt. Jede Zeile stellt gleichzeitig die jeweilige Dokumentrepräsentation im BoW-Modell dar	10
3	Berechnung der Dokumentenfrequenz (DF) und der Inversen Dokumentenfrequenz (IDF) für jedes Wort w_j auf Basis von drei Beispielsätzen. Bemerkenswert ist, dass zum Beispiel das Wort <i>likes</i> , welches zur Identifikation der Sätze im vorliegenden Korpus-Beispiel nicht hinreichend ist, ein IDF-Gewicht von 0 bekommt. Wohingegen dem Wort <i>tomato</i> eine vergleichsweise hohe Relevanz verliehen wird	11
4	Dokumentrepräsentationen im TF-IDF Modell für jeweils einen Beispielsatz. Dabei kommt jede Zeile durch die Multiplikation von der Inversen Dokumentenfrequenz (IDF) aus Tabelle 3 mit den entsprechenden Termfrequenzen (TF) aus Tabelle 2 zustande	11
5	Einträge der ungewichteten co-occurrence Matrix in tabellarischer Form. Grundlage ist der Beispielsatz 2 als Korpus, Fenstergröße beträgt 2	13
6	Einträge der mit dem inversen Abstand zum Kontextwort gewichteten co-occurrence Matrix in tabellarischer Form. Grundlage ist der Beispielsatz 2 als Korpus, Fenstergröße beträgt 2	13
7	Daten bestehend aus 15 Beobachtungen zum Trainieren eines Naïve Bayes Modells. Zielvariable: Stimmung; Kovariablen: Wetter und Beschäftigung	21
8	Relative Häufigkeitstabelle $P(x_w y)$. Datengrundlage dabei ist Tabelle 7	21
9	Relative Häufigkeitstabelle $P(x_B y)$. Datengrundlage dabei ist Tabelle 7	21
10	Definition der vier möglichen Fälle bei einer binären Klassifikation	26
11	Die Konfusionsmatrix als Gegenüberstellung zwischen den tatsächlichen und prädiktierten Klassenzugehörigkeiten. Die zugehörigen Abkürzungen werden in Tabelle 10 definiert	26
12	Ergebnistabelle der BoW-basierten Modelle. Zeilenweise sind die BoW- bzw. TF-IDF-Modelle mit jeweils Uni- und Bigram-Betrachtung dargestellt. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierte Verfahren, multinomiale Naïve Bayes sowie Random Forests. Grundlage hierbei ist ein Teil der Video Games-Rezensionen als Testdatensatz. Die Ergebnistabelle für den Beauty-Test befindet sich im Anhang C	33
13	Ergebnistabelle der GloVe-basierten Modelle. Zeilenweise sind die verschiedenen Vektorbildungsvarianten dargestellt. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierte Verfahren, gaußsche Naïve Bayes sowie Random Forests. Grundlage hierbei ist ein Teil der Video Games-Rezensionen als Testdatensatz. Die Ergebnistabelle für den Beauty-Test befindet sich im Anhang C	33
14	Die Konfusionsmatrix als Gegenüberstellung zwischen den tatsächlichen und prädiktierten Klassenzugehörigkeiten. Prädiktionen auf Basis vom Random Forests-Klassifikationsmodell mit Merkmalsextrahierung BoW (Unigram)	34

15	Klassifikationsbericht des Random Forests-Verfahrens mit Merkmalsex- tahierung BoW (Unigram): Auswertungskennzahlen für beide Bewertungs- kategorien	34
16	Die Konfusionsmatrix als Gegenüberstellung zwischen den tatsächlichen und prädiktierten Klassenzugehörigkeiten. Prädiktionen auf Basis vom Random Forests-Klassifikationsmodell mit Merkmalsexstahierung BoW (Unigram) nach dem Undersampling	35
17	Klassifikationsbericht des Random Forests-Klassifikationsmodells mit Merk- malsexstahierung BoW (Unigram): Auswertungskennzahlen für beide Be- wertungskategorien nach dem Undersampling	35
18	Ergebnistabelle der BoW-basierten Modelle. Zeilenweise sind die ver- schiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert werden vier Klassifi- kationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisie- rungsverfahren, multinomiale Naïve Bayes sowie Random Forests	36
19	Ergebnistabelle der GloVe-basierten Modelle. Zeilenweise sind die ver- schiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert werden vier Klassifi- kationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisie- rungsverfahren, gaußsche Naïve Bayes sowie Random Forests	36
20	Ergebnistabelle der GloVe-basierten Modelle. Zeilenweise sind die ver- schiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert wird das Klassifi- kationsverfahren der Logistischen Regression mit einer verschwindend ge- ringen Penalisierung (Ridge, $1/\lambda = 10^8$)	37
21	Ergebnistabelle der GloVe-basierten Modelle. Der Unterschied zur Ta- belle 13 besteht darin, dass bei der Normalisierung die Rezensionstexte lemmatisiert und die Stoppwörter entfernt wurden. Zeilenweise sind die ver- schiedenen Vektorbildungsvarianten dargestellt. Evaluiert werden vier Klassifi- kationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisie- rungsverfahren, gaußsche Naïve Bayes sowie Random Forests. Grundlage hierbei ist ein Teil der Video Games-Rezensionen als Testda- tensatz	38
22	Ergebnistabelle der BoW-basierten Modelle. Die 3-Sterne-Bewertungen werden zur positiven Kategorie eingeordnet. Zeilenweise sind die ver- schiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert werden vier Klassifi- kationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisie- rungsverfahren, multinomiale Naïve Bayes sowie Random Forests	39
23	Ergebnistabelle der GloVe-basierten Modelle. Die 3-Sterne-Bewertungen werden zur positiven Kategorie eingeordnet. Zeilenweise sind die ver- schiedenen Vektorbildungsvarianten dargestellt. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert werden vier Klassifi- kationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisie- rungsverfahren, gaußsche Naïve Bayes sowie Random Forests	39
24	Für alle Exrahierungsmodelle gleichbleibende Parameter	52

25	Wahl der Modellparameter im Falle balancierter Daten, wobei die Rezensionen beim Normalisierungsschritt zusätzlich lemmatisiert und die Stoppwörter entfernt wurden	52
26	Wahl der Modellparameter im Falle unbalancierter Daten	53
27	Wahl der Modellparameter im Falle balancierter Daten	54
28	Wahl der Modellparameter im Falle balancierter Daten, wobei die 3-Sterne-Bewertungen zur positiven Kategorie eingeordnet werden	55
29	Mittlerer Fehler von der stratifizierten Kreuzvalidierung der BoW-basierten Modelle	56
30	Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-basierten Modelle	56
31	Mittlerer Fehler von der stratifizierten Kreuzvalidierung der BoW-basierten Modelle. Grundlage hierbei sind die Trainingsdaten, die dem Downampling unterzogen wurden	57
32	Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-Modelle. Grundlage hierbei sind die Trainingsdaten, die dem Downsampling unterzogen wurden	57
33	Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-basierten Modelle. Trainings- und Testdaten wurden dem Undersampling unterzogen. Evaluiert wird das Klassifikationsverfahren der Logistischen Regression mit einer verschwindend geringen Penalisation	57
34	Mittlerer Fehler von der stratifizierten Kreuzvalidierung der BoW-basierten Modelle. Die 3-Sterne-Bewertungen werden in die positive Kategorie eingeordnet. Trainingsdaten wurden dem Undersampling unterzogen	58
35	Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-basierten Modelle. Die 3-Sterne-Bewertungen werden in die positive Kategorie eingeordnet. Trainingsdaten wurden dem Undersampling unterzogen	58
36	Mittlerer Fehler von der stratifizierten Kreuzvalidierung der GloVe-basierten Modelle. Beim Normalisierungsschritt wurden die Rezensionstexte lemmatisiert und die Stoppwörter entfernt wurden. Trainingsdaten wurden dem Undersampling unterzogen	58
37	Ergebnistabelle der BoW-basierten Modelle. Im Gegensatz zur Tabelle 12 fungiert hierbei der Beauty-Datensatz als Datengrundlage. Zeilenweise sind die BoW- bzw. TF-IDF-Modelle mit jeweils Uni- und Bigram-Betrachtung dargestellt. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierungsverfahren, multinomiale Naïve Bayes sowie Random Forests	59
38	Ergebnistabelle der GloVe-basierten Modelle. Im Gegensatz zur Tabelle 13 fungiert hierbei der Beauty-Datensatz als Datengrundlage. Zeilenweise sind die BoW- bzw. TF-IDF-Modelle mit jeweils Uni- und Bigram-Betrachtung dargestellt. Evaluiert werden vier Klassifikationsmodelle: Logistische Regression mit Ridge- und Lasso- Penalisierungsverfahren, gaußsche Naïve Bayes sowie Random Forests	59

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Quellen entnommen sind, wurden unter Angabe der Herkunft kenntlich gemacht. Hierzu zählen auch Zeichnungen, Skizzen sowie Internetquellen.

Ort, Datum

Asmik Nalmpatian