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

Identifizierung und Analyse von Privacy Issues in App Store Reviews

Detecting and Analyzing PrivacyIssues in App Store Reviews

Masterarbeit

im Studiengang Informatik

von

Raymond Baruth

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüferin: Dr. Jil Ann-Christin Klünder
Betreuer: M. Sc. Wasja Brunotte

Hannover, 01.06.2021

Erklärung der Selbstständigkeit

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

Hannover, den 01.06.2021

Raymond Baruth

Zusammenfassung

Identifizierung und Analyse von Privacy Issues in App Store Reviews

Bewertungen zu Apps aus App Stores stellen einen wichtigen Kommunikationskanal zwischen Nutzern und Entwicklern dar. Häufig wird in Bewertungen darauf hingewiesen, dass es Probleme mit den angebotenen Apps gibt. Dies können unter anderem Sicherheits-, Nutzbarkeits-, aber auch Datenschutzprobleme sein. Leider gehen diese Informationen häufig in der großen Masse der Bewertungen unter und wichtige Hinweise auf solche Probleme gehen verloren. Mit Verfahren aus dem Bereich des Maschinellen Lernens und des Natural Language Processing lassen sich Funktionen entwickeln, die das Finden solcher Probleme vereinfachen. Ein Verfahren zum Auffinden von Bewertungen, die einen Hinweis auf eine Privatsphäre- oder Datenschutzverletzung darstellen, wird in dieser Arbeit vorgestellt. Dazu wurden insgesamt mehr als 82 Millionen Bewertungen von mehr als 5.000 Apps aus dem Google Play Store und dem Apple App Store heruntergeladen um im nächsten Schritt ein Wortmodell mittels des Verfahren „word2vec“ zu trainieren. Mit Hilfe dieses Modells wird anschließend eine Menge ähnlicher Begriffe zum Themenbereich „Privacy Issue“ erstellt. Anhand dieser Menge an Worten, wurden 118 Apps aus verschiedenen Kategorien analysiert um so vorselektierte Bewertungen zu erhalten. Diese wurden anschließend manuell gelabelt um ein Trainingsset für verschiedene Verfahren des überwachten Lernens zu erhalten. So entstand ein Modell welches geeignet ist, Datenschutz- und Privatsphäreverletzungen in Bewertungen zu finden. Ein F1-Score von $95,8\%$ beweist unter anderem die Nutzbarkeit dieses Ansatzes. Abschließend wurde eine Knowledge Base für insgesamt 1.277 identifizierte Sätze aus Bewertungen mit Hinweisen auf Datenschutz- und Privatsphäreverletzungen erstellt und die darin enthaltenen Verletzungen in zehn Kategorien eingeteilt.

Abstract

Detecting and Analyzing Privacy Issues in App Store Reviews

App reviews from App Stores are an important communication channel between users and developers. Many of the reviews refer to problems that users have with the respective app. These problems are often due to usability, security or privacy issues. Unfortunately, these information is often overlooked in such a large collection of reviews and important indications to such problems are lost.By employing machine learning techniques in conjunction with natural language processing, these problems can be detected automatically so that they can subsequently be eliminated. This work presents an approach to automatically detect privacy-related issues in app reviews. For this purpose, a total of more than 82 million reviews of more than 5,000 apps were downloaded from the Google Play Store and the Apple App Store in order to train a word model using the „word2vec“ method. This model is then used to create a set of of similar terms for the topic „Privacy Issue“. Based on this set of words, 118 apps from different categories were analyzed to obtain pre-selected ratings. The ratings were manually labeled to obtain a training set for various methods of supervised learning. Therefore, a model was created that is suitable for finding privacy violations in ratings. Among other things, a F1-Score of 95.8% proves the usefulness of this approach. Finally, a knowledge base was created for a total of 1,277 identified sentences from reviews with indications of privacy violations, and the violations contained in those sentences were divided into ten categories.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Software begleitet uns durch unseren gesamten Tagesablauf. Wir nutzen sie auf der Arbeit, beim Sport in Form von Health Apps oder in unserer Freizeit. In nahezu jeder Software werden Daten erhoben, die wir teils freiwillig, teils aber auch unfreiwillig hinterlassen. Viele Nutzer wissen nicht einmal, das und in welchem Umfang ihre persönlichen Daten gesammelt und verwendet werden. Gerade deshalb ist es wichtig, dass die verantwortlichen Software-Hersteller mit diesen Daten verantwortungsbewusst, transparent und gesetzeskonform umgehen, da sonst ein Vertrauensverlust eintreten kann. Ist dies aber gewährleistet, kann das Vertrauen und die Akzeptanz von Seiten der Nutzer gestärkt werden. Noch wichtiger wird der Datenschutzaspekt, wenn persönliche Daten von Algorithmen genutzt werden, die diskriminierende Absichten verfolgen. Nicht selten werden so Nutzer, von diskriminierenden Algorithmen in eine Ecke gedrängt. Einmal in dieser Falle bekommen die Nutzer gar nicht mit, dass sie Opfer von diskriminierenden Algorithmen geworden sind und es ist essentiell wichtig in diesem Bereich mehr Datenschutz zu betreiben. Auch dazu kann Benutzer-Feedback in Form von App Store Bewertungen genutzt werden, Meinungen und Bedürfnisse von Nutzern zu analysieren und somit mögliche Probleme hinsichtlich verschiedener Aspekte der Softwarequalität zu identifizieren [14]. So hat auch der Stellenwert von App Store Bewertungen aus einem gemeinsamen Katalog wie dem Google Play Store oder dem Apple App Store stark zugenommen [2], sodass auch die Anzahl der verfassten Bewertungen zugenommen hat. Zum aktuellen Zeitpunkt existieren bspw. im Google Play Store ungefähr 3 Millionen Apps [1]. Daraus ergeben sich neue Möglichkeiten aber auch Probleme für die App Entwickler. So kann es durchaus schwierig sein, alle neu

verfassten Bewertungen zu analysieren und den Inhalt zu selektieren. Eine geeignete Filterfunktion zu verschiedenen Kategorien gibt es bisher nicht. Es existieren beispielsweise keine Filter, mit Hilfe derer Bewertungen angezeigt werden können die Verbesserungsvorschläge oder Probleme adressieren. Eine solche Filterfunktion würde es nicht nur den Entwicklern einfacher machen verschiedene Kategorien schneller zu sichten, um somit die Qualität der App zu erhöhen, auch Nutzer könnten sich so schneller und effizienter über eine App informieren. Ist ein Nutzer nur am Thema Privacy interessiert, könnten relevante Bewertungen schnell zusammengefasst und präsentiert werden. Der so entstandene Vorteil hätte direkten Einfluss auf die Nutzung der App durch Endanwender, da so wichtige Informationen schneller und einfacher ersichtlich wären und die Nutzer entscheiden könnten ob und wie sie die App sicher nutzen können. Weiterhin könnte eine solche Filterfunktion die Qualität einer App erheblich steigern, da auch wichtige Hinweise von Nutzern in den Bewertungen effizienter gefunden werden könnten, die dann wiederum von App Entwicklern genutzt werden könnten um die App zu verbessern. Allgemein gibt es in diesem speziellen Bereich der Privatsphäre- und Datenschutzanalyse in Bezug auf App Store Bewertungen kaum wissenschaftliche Untersuchungen. Häufig beziehen sich die durchgeführten Analysen der geschriebenen Bewertungen auf einfache Stimmungsanalysen [41] oder die Zuordnung einzelner Bewertungen in Kategorien [25]. Wissenschaftliche Untersuchungen von App Store Bewertungen speziell auf Privatsphäre- oder Datenschutzverletzungen existieren fast nicht. Da dies jedoch ein wichtiger Aspekt für Nutzer bei der Verwendung von Apps ist, greift diese Arbeit dieses Thema auf und stellt einen möglichen Lösungsansatz vor, um dieses Problem zu analysieren und zu lösen.

1.2 Problemstellung

Um Privacy Issues in App Store Bewertungen zu finden ist eine manuelle Suche nötig, diese wiederum ist aber u.a. aus Zeit- und Kostengründen schlichtweg in der Praxis nicht durchführbar. Zudem benötigt der Leser, bestimmtes Wissen aus dem Bereich „Datenschutz und Privatsphäre“ um Verletzungen der Privatsphäre überhaupt identifizierten zu können. Auch die App Entwickler müssten über dieses Know-how verfügen, um den Inhalt sicher zu erkennen. Der Zeitaufwand um einen Hinweis auf eine Verletzung der Privatsphäre zu finden, ist aktuell sehr hoch und nicht jeder Entwickler oder Nutzer macht sich die Mühe und sucht hunderte oder tausende Bewertungen nach möglichen Verletzungen ab. Dieser Umstand führt daher nicht zu einer Steigerung der Qualität einer App die in einem

App Store angeboten wird. In dieser Arbeit wird nun eine Möglichkeit vorgestellt, das Qualitätsmerkmal der Privatsphäre einer App zu steigern, indem eine geeignete Filterfunktion zur Verfügung gestellt wird. Das Analysieren und Kategorisieren von Bewertungen die in Textform vorliegen, ist ein komplexes Problem welches in den letzten Jahren zum Schwerpunkt vieler Forschungsarbeiten geworden ist (siehe Kapitel 7). Daher beschäftigt sich diese Arbeit mit der Analyse von schriftlichen Bewertungen aus App Stores. Diese Bewertungen werden mit Ansätzen aus dem Bereich Natural Language Processing und Machine Learning analysiert und ausgewertet.

1.3 Lösungsansatz

Um das Qualitätsmerkmal der Privatsphäre von mobilen Apps zu verbessern, wurde in dieser Arbeit ein Ansatz entwickelt, mit dem es möglich ist, automatisch eine große Masse an Bewertungen auf Privatsphäre- und Datenschutzverletzungen zu untersuchen. Weiterhin können identifizierte Privacy Issues kategorisiert und in einer Knowledge Base abgespeichert werden. Die vorhandenen Kategorien der identifizierten Privacy Issues wurden aus den in dieser Arbeit gefundenen Privacy Issues abgeleitet.

Die Bewertungen wurden mit Hilfe von Machine Learning Algorithmen analysiert und auf Privacy Issues hin untersucht. Es wurden Verfahren aus dem Bereich des überwachten Lernens verwendet, welche es erforderlich machten im Vorfeld Bewertungen die ein oder mehrere Privacy Issues adressieren zu identifizieren und dementsprechend zu beschriften. Da es laut vergangenen Untersuchungen [25] voraussichtlich nur 2,4% im Apple App Store beziehungsweise 0,7% im Google Play Store an Privacy Issues in den Bewertungen geben wird, wurde versucht die Masse an Bewertungen die Privacy Issues enthalten könnten, einzuschränken. Dazu wurde ein Wortmodell mit dem Verfahren word2vec [26] trainiert um anschließend mit einer intelligenten Schlüsselwortsuche, die Bewertungen vorzusortieren. Die so sortierten Bewertungen wurden anschließend manuell gelabelt um damit weitere Verfahren des überwachten Lernens anzuwenden. Das so erstellte Modell wurde anschließend dazu genutzt, um weitere Bewertungen auf Privacy Issues hin zu untersuchen. Mit diesem Verfahren kann nun ein entsprechender Filter für die beiden untersuchten App Stores bereitgestellt werden um die Qualität der App entsprechend zu verbessern. Letzteres ist nicht Bestandteil dieser Arbeit.

1.4 Struktur der Arbeit

Diese Arbeit ist wie folgt in acht Kapitel und vier Anhänge untergliedert. In Kapitel 1 wird das Thema eingeleitet, die Problemstellung geschildert, ein möglicher Lösungsansatz skizziert und die Struktur der Arbeit beschrieben. In Kapitel 2 werden einige wichtige Grundlagen in den Bereichen Privatsphäre, maschinelles Lernen, Deep Learning und Natural Language Processing erläutert. In Kapitel 3 wird das Konzept des Ansatzes beschrieben und auf die genauen Einzelheiten der Verarbeitungsschritte der App Bewertungen eingegangen. Des Weiteren werden verschiedene Möglichkeiten im Bereich maschinellen Lernens verglichen, die das Problem auf unterschiedliche Weise lösen können. Abschließend werden Überlegungen und Lösungsalternativen beschrieben. In Kapitel 4 wird die Architektur des entwickelten Programms beschrieben, sowie Details zur Umsetzung erläutert. Kapitel 5 listet alle erzielten Ergebnisse der Arbeit auf und geht speziell auf das Thema Labeling-Evaluation ein. Kapitel 6 diskutiert und interpretiert die erzielten Ergebnisse und beschreibt Limitationen der Arbeit sowie einige Herausforderungen. In Kapitel 7 wird über die Anwendbarkeit der Arbeit berichtet und ein Überblick über interessante wissenschaftliche Arbeiten gegeben, die dieser Arbeit inhaltlich ähnlich sind. Außerdem wird eine mögliche Schnittstelle zu einer verwandten Arbeit beschrieben. Schließlich erfolgt in Kapitel 8 die Zusammenfassung der Arbeit und ein Ausblick auf mögliche Weiterentwicklungen.

Kapitel 2

Grundlagen

In diesem Kapitel werden benötigte Grundlagen erörtert, die über die üblichen Lehrinhalte eines Bachelor of Science im Studiengang Informatik hinausgehen. Insbesondere wird der Begriff der Privatsphäre definiert, um ein einheitliches Verständnis zu schaffen. Weitere Schwerpunkte bilden die Themenbereiche des maschinellen Lernens und des Natural Language Processing.

2.1 Benutzerfeedback

Mit Softwareverteilungsplattformen wie dem Google Play Store oder dem Apple App Store, wurde Nutzern eine Möglichkeit gegeben, Feedback in Form von Bewertungen zu den entsprechenden Apps abzugeben. Diese Form der Kommunikation zwischen Entwicklern und Nutzern, beziehungsweise Nutzern und anderen Nutzern, erfreut sich höchster Beliebtheit. So zeigten bereits Pagano et al. [31], dass sich dieser Kommunikationskanal sehr gut eignet um beispielsweise Bugs zu kennzeichnen oder Verbesserungsvorschläge auszutauschen. Die Bewertungen bestehen aus verschiedenen Metainformationen, wie Text, einem Score von 1 (schlecht) bis 5 (sehr gut), einem Zeitstempel und einem Nutzernamen. Die Entwickler einer App sind außerdem in der Lage, auf Bewertungen zu antworten, um so ein beanstandetes Problem gegebenenfalls als „behoben“ zu kennzeichnen. Weiter ist es möglich, die Bewertungen mit einem „hilfreich“ oder „nicht hilfreich“ zu markieren. Die Sortierung der Bewertungen kann nach Score, Relevanz oder Datum erfolgen. Eine Möglichkeit diese Bewertungen nach Kategorien zu filtern, gibt es wie bereits erwähnt nicht. Um gezielt nach gesuchten Informationen zu filtern, ist ein Nutzer gezwungen sämtliche Bewertungen zu lesen. Einzige Hilfestellung dabei kann das „hilfreich“ Attribut bieten,

was wiederum ausschließlich auf der subjektiven Wahrnehmung anderer Nutzer basiert. Dennoch hat dieses System das Potential vorhandene Fehler schnell zu erkennen, da eine Vielzahl an Bewertungen direkt nach der Veröffentlichung verfasst werden [31]. So können Entwickler schneller auf das Feedback reagieren und das Problem gegebenenfalls beheben. Betrachtet man allerdings Apps wie Facebook, so wird dies für Nutzer und Entwickler schwierig bis unmöglich, da für diese und weitere Apps die Masse an eingehenden Bewertung zu groß ist um alle lesen zu können. Beispielsweise erhielt die App Facebook 4.275 Bewertungen [31] von Nutzern an einem Tag.

2.2 Privatsphäre

2.2.1 Definition Privatsphäre

Um ein einheitliches Verständnis des Begriffs der Privatsphäre zu etablieren, muss der Begriff zuvor klar definiert werden. Hierzu gibt es viele Ansätze diesen Begriff zu beschreiben. Das Verständnis wird auch je nach verwendeter Domäne differenziert definiert. Das Oxford Dictionary [30] bietet hier zwei Definitionen: 1. „*The state of being alone and not watched or interrupted by other people*“ und 2. „*The state of being free from the attention of the public*“. Diese Definitionen beschreiben zwei Hauptmerkmale der Privatsphäre. Zum einen betrifft es die zu schützende Person und das Recht einen privaten Raum um diese zu schaffen. Zum anderen beschreibt die Definition alle anderen Personen, die nicht in den Raum der zu schützenden Person eindringen dürfen. Im Kern geht es hier um Grenzen die nicht verletzt werden dürfen., jedoch existieren noch weitere Definitionen, die bereits von Renau et al. [37] gut zusammengefasst wurden. Die folgende Definition wird auch in dieser Arbeit verwendet, um die Bewertungen aus den App Stores zu analysieren.

Definition 1
Privacy is the faculty and right that a person has to define, preserve and control the boundaries that limit the extent to which the rest of society can interact with or intrude upon. At the same time, he or she retains full control over information generated by, and related to, him or her. [37]

Bei dieser Definition wird die Privatsphäre eines Menschen, als Menschenrecht angesehen und gibt einer Person somit das Recht, die Kontrolle und Verantwortung für die Abgrenzung der persönlichen Grenzen zu tragen. Weiterhin wird angenommen, dass die bloße Existenz eines Menschen schützenswerte Informationen beinhaltet, die gegen Offenlegung geschützt werden müssen und unter der Kontrolle des betreffenden Menschen bleiben müssen. Insgesamt wird in der Arbeit [37] folgende Erklärung vorgeschlagen:

> *Privatsphäre ist das Recht einer Person die Offenlegung von persönlichen Daten zu kontrollieren und in Ruhe gelassen zu werden indem die persönlichen Grenzen durchgesetzt werden.[37]*

Ein anderes Konzept stellt die Vertraulichkeit [43] dar, bei der es um die Weitergabe oder Offenlegung persönlicher Informationen durch Dritte geht. Die Vertraulichkeit befasst sich also mit der Verantwortung eines Dritten zur Wahrung der persönlichen Informationen einer anderen Person. Hingegen sich die Privatsphäre mit der Kontrolle einer Person über ihre eigenen persönlichen Informationen befasst. Werden private Informationen offengelegt, muss dem Verwahrer vertraut werden, dass dieser die Vertraulichkeit der Daten schützt. Leider kann dies in der Realität oft nicht sichergestellt werden, sodass die betroffenen Personen wenigstens in die Lage versetzt werden sollten, die Kontrolle über ihre Daten zu behalten. Das kann erreicht werden, indem beispielsweise geeignete Kontrollmechanismen zur Verfügung stehen um das Risiko einer Privatsphäre Verletzung minimieren zu können. Ein geeigneter Kontrollmechanismus, kann der in dieser Arbeit beschriebene Ansatz zum Auffinden von Privatsphäreverletzungen sein.

2.2.2 Definition: Privacy Issue

In dieser Arbeit wird eine Bewertung oder ein Satz als „Privacy Issue“ betrachtet, sobald eine Verletzung der Definition 1 aus Unterkapitel 2.2.1 vorliegt. Hierbei wurde besonderer Wert auf das Vorhandensein, einer negativen Auswirkung oder Aussage gelegt. Bemerkungen wie beispielsweise „This app protects your privacy on the web.“ werden in dieser Arbeit nicht als Privacy Issue betrachtet, da nur die negativen Auswirkungen einer App auf die Privatsphäre erfasst werden sollen. In positiven Aussagen in den Bewertungen, ist kein Handlungsbedarf für den Entwickler zu erwarten, da diese Bewertungen darauf abzielen ein Lob zu erklären. Der hier erklärte Ansatz der Arbeit, soll dafür verwendet werden die Qualitätsaspekte der Software zu verbessern und dies ist nur möglich, wenn die Entwickler über die negativen Einflüsse ihrer Apps unterrichtet werden.

2.3 Maschinelles Lernen

Maschinelles Lernen, im Englischen Machine Learning, ist ein Teilgebiet der künstlichen Intelligenz. Mit diesen Verfahren sind IT-Systeme in der Lage, verschiedenste Muster und Gesetzmäßigkeiten in vorliegenden Daten zu erkennen und selbstständig Lösungen für Probleme zu finden. Die

folgenden Kapitel beschreiben die wichtigsten Verfahren und geben einen grundlegenden Überblick über diesen Themenbereich. Vertiefende Literatur zu diesem Themenbereich bieten bspw. die Bücher „Real-World Machine Learning“ von Brink et al. [38] und „A Concise Introduction to Machine Learning“ von Anita Faul [11].

2.3.1 Unüberwachtes Lernen

Beim unüberwachten Lernen oder Unsupervised Learning [18, S. 8], handelt es sich um ein Verfahren welches Kategorien von nicht gelabelten Daten lernen kann. Dazu werden verschiedene Verfahren wie „Clustering“ oder „Dimensionsreduktion“ verwendet. Angenommen es handelt sich um Daten einer Kochrezeptdatenbank mit Angaben über die Speisekategorie wie Kuchen, Fleischspeisen oder Getränke, so würde ein Verfahren aus dem Bereich des unüberwachten Lernens, neue Rezepte automatisch einer dieser Kategorien zuordnen können.

2.3.2 Überwachtes Lernen

Das überwachte Lernen oder Supervised Learning [18, S. 7 - 8], basiert auf dem Lernen aus einem Datensatz welcher ein Label für jedes Exemplar aus den Daten zur Verfügung stellt. Beispielsweise könnte dies eine Bewertung eines Produkts aus dem Amazon Marketplace sein, in der ein Score (1-5) vergeben wurde. Der Score fungiert nun als Label, der Bewertungstext als Feature. Nach der Lernphase kann das so erstellte Modell nun anhand eines eingegebenen neuen Textes einen voraussichtlichen Score voraussagen.

Weiterhin unterscheidet man zwei Verfahren beim Supervised Learning:

- **Klassifikation** weißt einer Eingabe eine eindeutige Kategorie zu. Ein Beispiel für ein Klassifikationsverfahren wird in Tabelle 2.1 gegeben. Dort handelt es sich um Bewertungen einer App, bei der ein Label anhand der Sentiments vergeben wurde. Ein neuer Eingabetext „ *This app is amazing!*“ könnte nun entweder der Klasse „positive“ oder „negative “ zugeordnet werden.

- **Regressionsverfahren** weisen einer Eingabe einen numerischen Wert zu. Ein Beispiel für ein Regressionsproblem ist in Tabelle 2.2 dargestellt. Dort werden die gleichen Bewertungen aus dem vorhergehenden Beispiel verwendet um jetzt einen numerischen Wert vorherzusagen, dem vergebenen Score vom 1 bis 5. Der gleiche Satz aus dem Beispiel könnte nun ein Label aus dem Wertebereich: $1 \leq score \leq 5$ erhalten.

Bewertungstext	**Label**
I like the app very much.	positive
The app violates the GDPR.	negative
Users should not install this app.	negative
The game is very fun to play.	positive

Tabelle 2.1: Beispiel für eine Klassifikation

Bewertungstext	Score
I like the app very much.	5
The app violates the GDPR.	1
Users should not install this app.	2
The game is very fun to play.	4

Tabelle 2.2: Beispiel für eine Regression

2.3.3 Labeling

Wie in Unterkapitel 2.3.2 bereits erwähnt, werden bei Verfahren des überwachten Lernens, Datensätze benötigt die neben den eigentlichen Daten auch ein eindeutiges Label benötigen [18, S. 41 - 43]. Diese Datensätze müssen vor dem Lernprozess der verschiedenen Algorithmen, in den meisten Fällen manuell gelabelt werden. Label können als Kategorie betrachtet werden, in die die Daten einsortiert werden müssen. Man unterschiedet hier zwischen binären labeln und dem Multiklassenlabeling. Beim binären Labeling gibt es lediglich die Einteilung der Daten in zwei Kategorien, wohingegen beim Multiklassenlabeling, mehrere Kategorien vorhanden sind. Die zu labelnden Daten werden dabei oft mit X bezeichnet und das entsprechende Label mit y. So wird jedem X ein oder mehrere y zugeordnet. Ein Beispiel für ein binäres Labeling könnte ein Datensatz von Bildern die Rehe und Kühe abbilden sein und ein entsprechendes Label könnte mit den Zahlen 0 und 1 codiert werden. Ist auf dem Bild eine Kuh abgebildet, so wird das Label 0 gewählt, ist ein Reh abgebildet so wird dieses mit 1 gelabelt. Dieses Datenset kann anschließend als Eingabe für einem Algorithmus des überwachten Lernens genutzt werden, um ein entsprechendes Modell zu trainieren. Da bei der Regression keine Einteilung in Kategorien erfolgt wie bei der Klassifikation, wird hier nicht der Begriff des Labels verwendet, vielmehr handelt es sich hier um eine „Prediction“ oder einem „Target“ welches vergeben wird.

2.4 Klassifizierung

In diesem Kapitel werden Algorithmen aus dem Bereich des überwachten Lernens vorgestellt. Die Klassifizierung beschreibt einen Vorgang, bei dem Eingabedaten in zwei oder mehr Klassen eingeordnet werden können [17, S. 7 - 9]. Dazu muss mit gelabelten Eingabedaten ein Modell trainiert werden (Trainingsphase), welches anschließend in der Lage ist, eine Einordnung in vorher definierte Klassen durchzuführen (Prediction-Phase). Im Bereich des Natural Language Processing werden Textdaten als Eingabedaten für die Klassifizierungsalgorithmen genutzt, welche wiederum in „Merkmale“ zerlegt werden. Diese Merkmale können beispielsweise Text-Heuristiken, wie die Wortlänge oder Auftrittshäufigkeiten sein, aber auch Repräsentationen als Vektoren sind möglich. Das Unterkapitel 2.4.1 beschreibt einige wichtige Klassifizierungsalgorithmen und in Unterkapitel 2.4.2 und 2.4.3 werden zwei spezielle Einsatzmethoden dieser Klassifizierungsalgorithmen vorgestellt.

2.4.1 Klassifizierungsalgorithmen

Der in dieser Arbeit beschriebene Ansatz, verwendet unter anderem die hier folgenden Verfahren zur Klassifizierung :

- **Logistische Regression** Bei der logistischen Regression [18, S. 61 - 62] handelt es sich um eine bestimmte Form der Regression die verwendet wird, wenn die abhängige Variable nominalskaliert beziehungsweise ordinalskaliert ist. Ein Beispiel hierfür wäre die Variable „Spam-E-Mail“ mit den zwei Ausprägungen „Spam“ oder „kein Spam“.

  Bei einer Zuordnung in lediglich zwei Ausprägungen, spricht man von der binären logistischen Regression. Während man bei mehreren Ausprägungen von der multinominalen logistischen Regression spricht. Der in dieser Arbeit beschriebene Ansatz verwendet ausschließlich die binäre logistische Regression, weshalb hier auch nur dieser Ansatz erläutert wird.

  In der Grundform kann die logistische Regression die Variablen 0 oder 1 prognostizieren. Hierfür wird die Wahrscheinlichkeit zwischen 0 und 1 für das entsprechende Merkmal geschätzt. Um dies zu ermöglichen, wird die logistische Funktion zur Berechnung genutzt. Diese Funktion besitzt die Eigenschaft, dass für Eingabewerte von minus und plus unendlich, immer nur Werte zwischen 0 und 1 angenommen werden, welches auf der Abbildung 2.1 dargestellt ist. Die Gleichung zur logistischen Funktion lautet:

$$f(z) = \frac{1}{1+e^{-(b_1*x_1+\cdots+b_k*x_k+a)}}$$

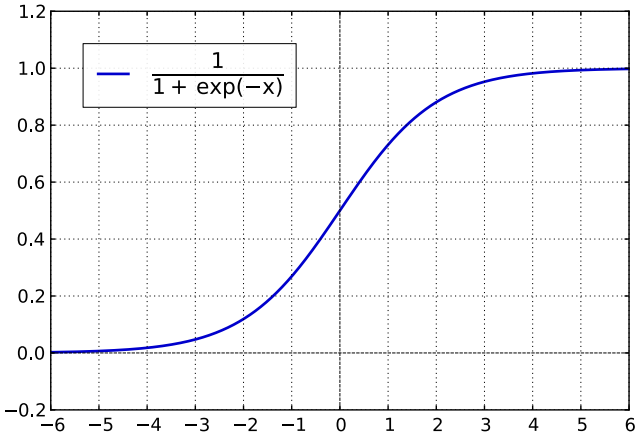


Abbildung 2.1: Logistische Funktion

- **Support Vector Machine (SVM)** Die SVM [18, S. 67 - 68] ist eine mathematische Methode, die im Bereich des maschinellen Lernens verwendet wird um Eingabedaten zu klassifizieren. Dabei wird die lineare als auch die nicht-lineare Klassifizierung unterstützt. Bei einer SVM werden die Eingabedaten als Vektoren in einem hochdimensionalen Vektorraum betrachtet. In diesem Vektorraum wird versucht eine Hyperebene zu ermitteln, welche eine möglichst fehlerfreie Separierung der Eingabedaten in Klassen ermöglicht. Die so entstandenen Klassengrenzen werden möglichst breit gewählt, sodass ein möglichst großer, freier Bereich zwischen den Objekten entsteht. Diese Methode wird *Maximum Margin Separation* genannt.

- **Decision Tree Classifier** Auch der Decision Tree Algorithmus [11, S. 123 - 135] gehört zur Familie der überwachten Lernalgorithmen und kann zur Lösung von Regression- und Klassifikationsproblemen eingesetzt werden. Ein Decision Tree stellt eine feste Abfolge von Entscheidungen dar, die getroffen werden müssen um ein bestimmtes Ergebnis zu erzielen. Er besteht aus drei Teilen:

  - *Nodes*: Test des Wertes eines bestimmten Attributes.
  - *Edges / Branch*: Entsprechen dem Ergebnis eines Tests und verbinden den nächsten Node oder Leaf.
  - *Leaf Nodes*: Endknoten die das Ergebnis ausgeben.

Alle genannten Teile des Decision Trees, werden in der Trainingsphase aufgebaut, um anschließend in der Prediction-Phase eine Klassifikation oder Regression durchführen zu können. Hierbei wird immer am Wurzelknoten begonnen und je nach Entscheidung an den Edges ein neuer Node oder Leaf besucht. In der Praxis wird die Tiefe des Baum oft begrenzt, um Overfitting zu vermeiden.

Das folgende Beispiel in Abbildung 2.2 zeigt eine Entscheidungsfindung anhand eines zuvor trainierten Decision Trees. Angenommen das Problem besteht darin, ein Lebewesen in eine bestimmte Klasse einzuordnen, dann sind die entsprechenden Fragen mit den Nodes verknüpft und die Leaf Nodes stellen die letztendlich vorhergesagte Klasse dar. Wird beispielsweise eine „Katze“ als Eingabe verwendet, wird der Baum aus Abbildung 2.2 wie folgt durchlaufen:

1. Geht auf vier Beinen? - Yes
2. Hat es ein Fell? - Yes
3. Ist ein Haustier? - Yes
4. Macht es „Miau“? - Yes
5. Entscheidung: Katze

Nach dem ein Leaf Node erreicht wurde, kann die Eingabe eindeutig einer Kategorie zugeordnet werden und der Algorithmus wird beendet.

- **Random Forest Classifier**

  Der Random Forest [38, S. 74] ist eng verwandt mit dem Decision Tree, da der Aufbau dem des Decision Trees ähnelt. Allerdings besteht der Random Forest aus zufällig variierten Entscheidungsbäumen und kombiniert die Ausgabe dieser Bäume um eine endgültige Entscheidung zu treffen. Dieses Vorgehen wird auch als Ensemble-Learning bezeichnet, welches in Kapitel 2.4.2 noch genauer erläutert wird. Durch die Variation der verschiedenen Bäume kann eine bessere Entscheidung getroffen werden. Ein Beispiel soll den Unterschied zwischen Decision Tree und Random Forest verdeutlichen. Angenommen eine Bank möchte entscheiden ob sie einem Kunden einen Kredit geben sollte oder nicht. Sie setzt dabei den Decision Trees aus Abbildung 2.3 zur Entscheidung ein. Der Kunde hat bereits einen laufenden Kredit über €500. Die erste Entscheidung im Decision Tree lautet: „Hat der Kunde bereits einen laufenden Kredit?“ welches mit „Ja“ beantwortet wird. Der Algorithmus terminiert an dieser Stelle mit der Absage des neuen Kredites. Hätte die Bank gewusst, dass der laufende Kredit lediglich

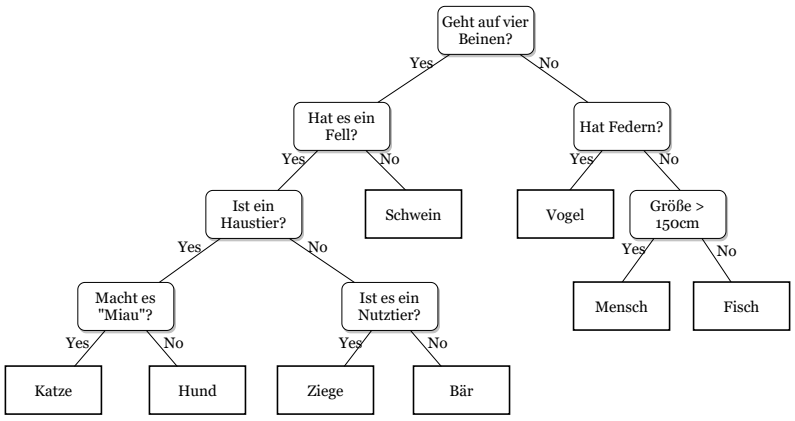


Abbildung 2.2: Beispiel Decision Tree

über €500 bestand, hätte sie den neuen Kredit jedoch genehmigt. So ist der Bank ein gutes Geschäft verwehrt geblieben. Jetzt setzt die Bank einen Random Forest zur Entscheidung ein. Bei diesem werden die drei Fragen an den Nodes variiert. So erhält man ein positives Abstimmungsergebnis, da die Frage nach dem laufenden Kredit nicht gleich zu Anfang zum Terminieren des Algorithmus führt, sondern auch die anderen Fragen betrachtet werden können. Hätte die Bank einen Random Forest eingesetzt, hätte sie eine bessere Entscheidung treffen können und der neue Kredit wäre genehmigt worden.

2.4.2 Ensemble

Bei einem Ensemble handelt es sich um eine Kombination von eingesetzten Klassifizierungsalgorithmen und deren Entscheidungen. Hierbei werden die Entscheidungen der einzelnen Klassifizierungsalgorithmen kombiniert, sodass beispielsweise ein Mehrheitsentscheid eingesetzt werden kann. Diese Methode wird auch Voting Classifier (VC) genannt. Jeder Klassifizierungsalgorithmus wird dabei unabhängig, aber mit den gleichen Trainingsdaten trainiert. Anschließend gibt jeder Klassifizierer seine Entscheidung an den VC weiter, welcher dann die endgültige Entscheidung des Ensembles trifft. Bei dieser Art

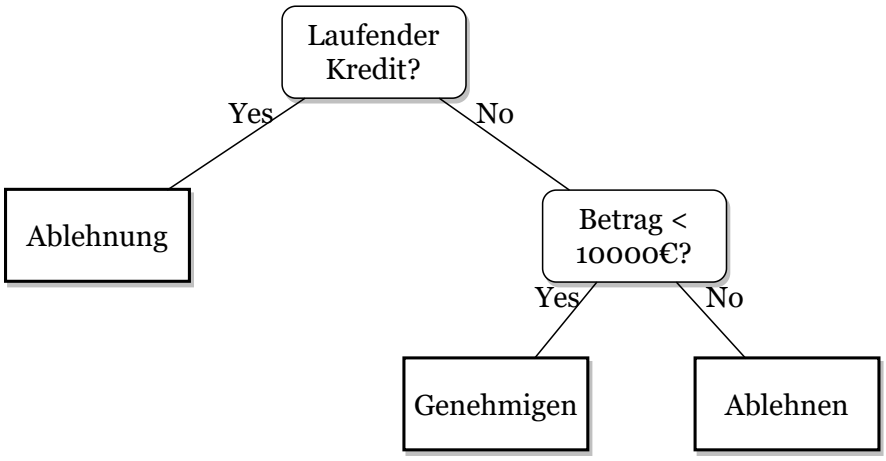


Abbildung 2.3: Beispiel für einen Decision Tree zur Kreditgenehmigung.

der Kombination wird in „soft“ und „hard“ unterschieden. „Soft“ bedeutet, dass die Entscheidung anhand der ermittelten Wahrscheinlichkeiten der einzelnen Klassifizierer ermittelt wird, während bei „hard“ ein einfacher Mehrheitsentscheid verwendet wird.

2.4.3 Stacking Classifier

Ein Stacking Classifier ist dem Ensemble aus Kapitel 2.4.2 sehr ähnlich, jedoch werden die Ergebnisse der einzelnen Klassifizierungsalgorithmen (Basismodelle) nicht direkt für eine Entscheidung genutzt. Hier kommt ein sogenannter Meta-Classifier (Metamodell) zum Einsatz, der die Ergebnisse der einzelnen Klassifizierungsalgorithmen als Eingabe für einen weiteren Klassifizierungsalgorithmus nutzt (siehe Abbildung 2.4). Die finale Klassifizierung wird anschließend vom Meta-Classifier entschieden. Das Metamodell basiert auf den Vorhersagen der Basismodelle. Validierungsdaten aus den Trainingsdaten der Basismodelle werden genutzt, um das Metamodell zu trainieren. Die so entstandenen Eingabe- und Ausgabepaare der Basismodelle zusammen mit den Validierungsdaten, werden zur Anpassung des Metamodells genutzt. Im Falle einer Klassifizierung werden hier die vorhergesagten Wahrscheinlichkeiten der Basismodelle verwendet.

2.4.4 Modell Performance

Bei einer Klassifikation werden nicht nur richtige Entscheidungen von den Klassifizierungsalgorithmen getroffen, es werden im Allgemeinen auch fehlerhafte Zuordnungen entschieden. Um die Leistungsfähigkeit von Modellen des

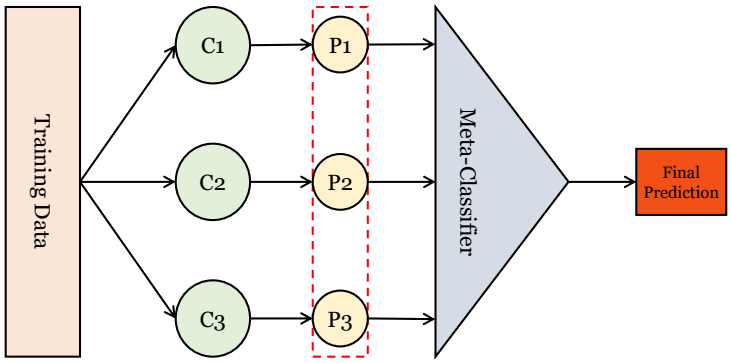


Abbildung 2.4: Schematische Darstellung eines Stacking Classifiers. C1, C2, C3 stellen die Basis-Classifier dar. P1, P2, P3 sind die Vorhersagen des Basismodells, die zum Training des Meta-Classifiers genutzt werden. Abbildung in Anlehnung an Ceballos [5].

maschinellen Lernens zu ermitteln, werden Metriken wie *Accuracy, Precision, Recall und* F_1 *Score* eingesetzt [18, S. 47 - 49]. Diese Metriken basieren auf der Einteilung der zu klassifizierenden Objekte in *True Positives (TP), False Positives (FP), True Negatives (TN)* und *False Negatives (FN)*. Die Erläuterung dieser Begriffe erfolgt in Kapitel 2.4.5.

- **Precision (P)** oder *Positive predictive value* beschreibt das Verhältnis der korrekt positiven Vorhersagen zu den gesamten vorhergesagten Positiven.

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

- **Recall (R)** oder *True positive rate* oder *hit rate* oder *sensitivity* beschreibt das Verhältnis der korrekten positiven Vorhersagen zu allen positiven Elementen.

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

- **Accuracy (A)** beschreibt das Verhältnis der korrekt vorhergesagten Elemente zu den gesamten Elementen.

$$Accuracy = \frac{TN+TP}{(TP+FN+FP+TN)}$$

- F_1 **Score** beschreibt das Gleichgewicht zwischen Precision und Recall.

$$F_1 Score = 2 \cdot \frac{Precision \cdot Recall}{(Precision + Recall)}$$

2.4.5 Konfusionsmatrix

Wenn Objekte durch einen Klassifizierungsalgorithmus in Klassen eingeordnet werden, kann der Klassifikator Fehlentscheidungen treffen, sodass Objekte einer falschen Klasse zugeordnet werden. Aus der Häufigkeit der Fehlentscheidungen lässt sich eine Beurteilung der eingesetzten Klassifizierungsalgorithmen ableiten. In dieser Arbeit wird lediglich die Funktionsweise eines binären Klassifikators erläutert, da ausschließlich dieser zum Einsatz kommen wird. Bei der binären Klassifikation, handelt es sich immer um die Frage, ob ein Objekt zur Klasse gehört oder nicht. Dazu ist es nötig, die richtige Klasse des Objektes zu kennen um beurteilen zu können, ob der Klassifikator dieses Objekt in die korrekte Klasse eingeordnet hat. Ein Beispiel soll diese Möglichkeiten verdeutlichen. Angenommen ein Klassifikator soll auf Bildern rote Ampeln erkennen, dann gibt es folgende mögliche Zuordnungen:

- True Positive (TP): Auf dem Bild ist eine rote Ampel, und der Klassifikator hat diese korrekt erkannt.
- False Negative (FN): Auf dem Bild ist eine rote Ampel, und der Klassifikator hat diese nicht erkannt.
- False Positive (FP): Auf dem Bild ist keine rote Ampel, und der Klassifikator hat eine rote Ampel erkannt.
- True Negative (TN): Auf dem Bild ist keine rote Ampel, und der Klassifikator hat dies erkannt.

Somit ergibt sich folgende Matrix, die in Tabelle 2.3 dargestellt ist. Diese Matrix wird Konfusionsmatrix genannt [18, S. 47].

		Actual class	
		P	**N**
Predicted	**P**	TP	FP
	N	FN	TN

Tabelle 2.3: Darstellung einer Konfusionsmatrix.

2.5 Deep Learning

Deep Learning [17, S. 31 - 43] ist ein Teilgebiet des maschinellen Lernens, bei dem künstliche neuronale Netze und große Datenmengen eine zentrale Rolle übernehmen. Künstliche neuronale Netze sind Algorithmen, die nach dem biologischen Vorbild des menschlichen Gehirns erstellt wurden. Diese werden eingesetzt um Muster zu erkennen, Texte zu deuten oder helfen können Cluster zu bilden. Außerdem sind diese Netze sehr gut geeignet um Objekte auf Bildern zu klassifizieren. Wie schon bei den klassischen Verfahren des maschinellen Lernens, werden auch neuronale Netze anhand von Daten trainiert. Durch die hohe Anzahl an trainierbaren Parametern, sind diese Netze oft sehr komplex, was die getroffenen Entscheidungen oft schwer nachvollziehbar erscheinen lässt. Ein einfaches neuronales Netz ist in Abbildung 2.5 abgebildet. Es besteht aus einer Eingabeschicht (Input Layer), einer versteckten Schicht (Hidden Layer) sowie einer Ausgabeschicht (Output Layer). Die Neuronen der Hidden Layer verfügen über sogenannte Gewichte, die den Input-Signalen ein bestimmtes Output-Signal zuweisen. Bei tiefen neuronalen Netzen, wird davon ausgegangen, dass mehr als ein Hidden Layer existiert.

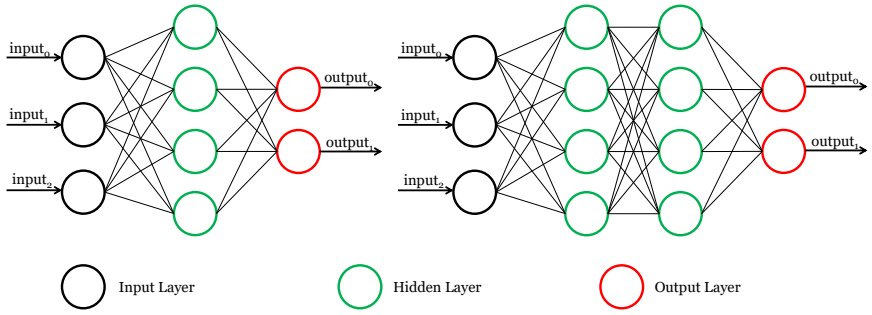


Abbildung 2.5: Aufbau eines neuronalen Netzes. Links: einfaches neuronales Netz, rechts: tiefes neuronales Netz mit mehreren Hidden Layern.

2.5.1 Perceptron

Das einfachste und zugleich älteste neuronale Netz wird als Perceptron [18, S. 143 - 146] bezeichnet welches in Abbildung 2.6 dargestellt ist. Das Konzept wurde bereits 1958 von Frank Rosenblatt in seiner Arbeit „The perceptron: a probabilistic model for information storage and organization in the brain.[40]“ vorgestellt. Es addiert die Eingabeparameter, wendet eine Aktivierungsfunktion an und sendet das Ergebnis an die Ausgabeschicht.

Die Ausgabe ist binär und damit, mit einer binären Klassifikation zu vergleichen. Indem der Wert der Aktivierungsfunktion mit einem Schwellwert verglichen wird, erhält man die Entscheidung. Dabei wird beim Überschreiten des Schwellwerts eine „1“ ausgegeben bzw. eine „0“ bei unterschreiten des Schwellwerts. Weitere Aktivierungsfunktionen wurden im Laufe der Zeit entwickelt damit auch weitere Werte zwischen 0 und 1 ausgegeben werden können. Eine der bekanntesten Aktivierungsfunktionen, ist die Sigmoid-Funktion welche in Abbildung 2.1 dargestellt ist.

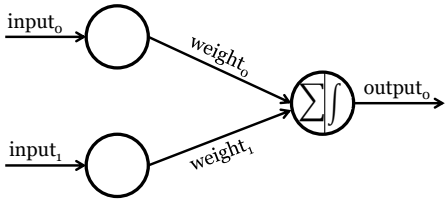


Abbildung 2.6: Schematischer Aufbau eines Single-Layer Preceptrons.

2.5.2 Feedforward Neural Network

Ein Feedforward Neural Network (FNN) zeichnet sich dadurch aus, dass die einzelnen Schichten nur mit den nächst höheren Schichten verbunden sind. So können sich die Eingaben nur in eine Richtung im Netz bewegen. Es gibt bei diesem Netzwerk keine rückwärts gerichteten Kanten, somit sieht der Trainingsprozess eines Feedforward Netzes wie folgt aus:

1. alle Knoten sind verbunden
2. Aktivierung läuft von Eingabe- zu Ausgabeschicht
3. es existiert mindestens eine Schicht zwischen Eingabe- und Ausgabeschicht

Existieren mehrere Schichten zwischen Eingabe- und Ausgabeschicht, spricht man von einem Deep Feedforward Neural Network.

2.5.3 Convolutional Neural Network

Convolutional Neural Network (CNN) sind künstliche neuronale Netze [17, S. 43 - 50] [18, S. 263 - 310] die besonders effizient bei der Verarbeitung von 2D und 3D Eingabedaten sind. Der Unterschied zu klassischen neuronalen

Netzen besteht in der Architektur von CNNs. Die Hidden Layer eines CNNs bestehen aus einer Folge von Faltungs- und Pooling-Operationen. Bei der Faltung wird ein sogenannter Kernel oder Filter über die Daten geführt, welcher in Abbildung 2.7 dargestellt ist. Darauf wird eine „Faltung“ berechnet, was mit einer Multiplikation vergleichbar ist. Anschließend werden die Gewichte der einzelnen Neuronen aktualisiert. Ein anschließender Pooling-Layer kann genutzt werden, um die Ergebnisse zu glätten oder zu vereinfachen. Auf diese Weise bleiben nur die wichtigsten Informationen erhalten. Wird dieser Vorgang fortgeführt, ergibt sich am Ende in der Ausgabeschicht ein Vektor, dem „Fully Connected Layer“. Dieser abschließende Layer besitzt immer genau so viele Neuronen wie Klassen die klassifiziert werden sollen. Die Zuordnung zu diesen Klassen erfolgt über Wahrscheinlichkeiten.

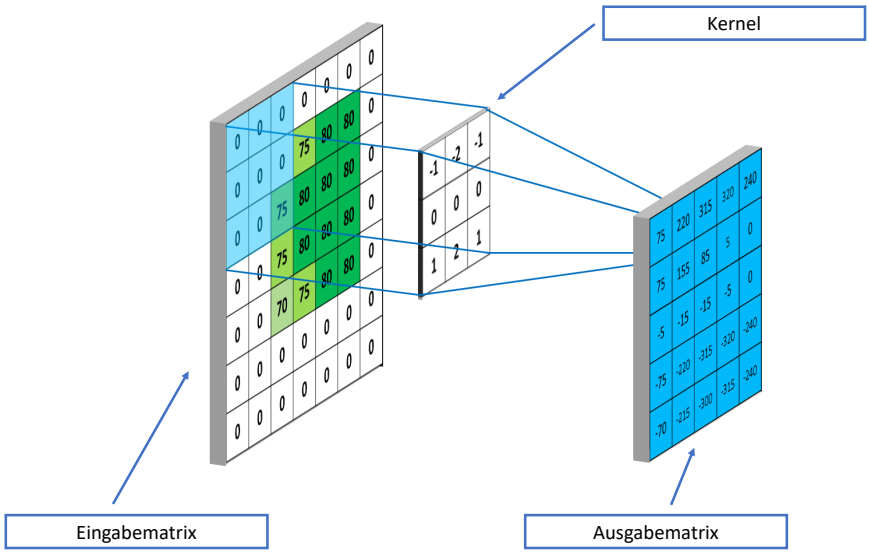


Abbildung 2.7: Aufbau einer Faltungsoperation mit Anwendung eines Filters / Kernels. Der verwendete Filter (3x3) bewirkt unter anderem eine Dimensionsreduktion der Eingabematrix von 7x7 zu 6x6. Abbildung in Anlehnung an [46].

Fully Connected Layer

Bei einem Fully Connected Layer oder Dense Layer handelt es sich um eine neuronale Netzstruktur, bei der alle Neuronen mit allen Inputs und allen Outputs verbunden sind. Um die Ausgaben der Convolutional- und Pooling Layer in einem Dense Layer verarbeiten zu können, muss dieser zunächst

mit einem Flatten Layer ausgerollt werden. So wird beispielsweise aus der Eingabedimension (batch_size, 2,2), die Ausgabedimension des Flatten Layers (batch_size, 4).

Pooling Layer

Ein Pooling- oder Downsampling-Layer ist für die Reduzierung der räumlichen Größe der Eingabematrix zuständig indem die Ergebnisse aggregiert werden. Diese Layer werden verwendet, wenn es nötig ist nach mehreren Stufen anderer Schichten, den Rechenaufwand schrittweise zu reduzieren sowie die Wahrscheinlichkeit eines Overfittings zu minimieren. Dazu wird bspw. bei einem MaxPooling Layer der höchste Wert der Kernel-Matrix verwendet, wohingegen die anderen Werte verworfen werden. Dieses Vorgehen dient dazu, nur die wichtigsten Signale an die folgenden Layer weiterzugeben und so eine Abstraktion des Inhalts zu erreichen. Weiterhin wird durch das Verwerfen von Werten, die Anzahl der Parameter eines Netzes reduziert und somit tritt eine Dimensionsreduktion ein. Wird bspw. eine Kernel-Matrix von 2x2 als Eingabe genutzt und darauf ein MaxPooling angewandt, erhält man als Ausgabe lediglich die höchste der vier (2x2) Zahlen als Ausgabe.

ReLU Aktivierungsfunktion

Bei einem CNN werden die Ergebnisse jedes Layers durch eine Aktivierungsfunktion aktiviert. Eine häufig eingesetzte Aktivierungsfunktion ist die Rectified Linear Unit (ReLU)-Aktivierungsfunktion. Diese Funktion veranlasst, dass alle Werte die kleiner als Null sind, zu Null werden und alle Werte, die größer als Null sind so erhalten bleiben. Bei einem Multiklassen-Klassifizierungsproblemen erhält die letzte Schicht häufig eine Softmax-Aktivierungsfunktion. Diese bewirkt, dass die Ausgaben aller Output-Neuronen sich zu 1 addieren und die jeweiligen Wahrscheinlichkeiten des entsprechenden Outputs ausgegeben werden. Handelt es sich um ein binäres Klassifizierungsproblemen, wird auch häufig die Sigmoid-Funktion als Aktivierung des letzten Layers genutzt.

2.5.4 Recurrent Neural Network

Recurrent Neural Network (RNN) [17, S. 50 - 53] [18, S. 315 - 365] fügen den künstlichen neuronalen Netzen Zellen hinzu, die es dem Netz ermöglichen ein Gedächtnis zu erhalten. Es ist dabei vergleichbar mit einem einfachen Feedforward Netz. Dennoch existiert eine Vielzahl verschiedener Varianten für diese Art Netz. So kann die Anzahl der Eingangsknoten variiert, eine

variable Verzögerung konfiguriert, oder verschiedene Architekturen gewählt werden. In Abbildung 2.8 wird eine RNN Zelle detailliert dargestellt. Diese Art von neuronalem Netz wird häufig dann verwendet, wenn der Kontext der zu analysierenden Daten von Bedeutung ist und wird daher häufig im Bereich des Natural Language Processings verwendet. Aber auch bei der Verarbeitung von Bildsequenzen ist der zeitliche Zusammenhang häufig wichtig. So werden diese Netze auch vorzugsweise für das autonome Fahren herangezogen.

Long Short-Term Memory - LSTM

Bereits 1997 stellten Hochreiter und Schmidhuber in ihrer Arbeit *Long Short-Term Memory* [15] erstmals die Long short-term memory (LSTM) Architektur vor [18, S. 324 - 325]. Dabei handelt es sich um eine spezielle Art eines RNN-Netzes, welches speziell dafür entworfen wurde, Langzeitabhängigkeiten zu erlernen. Die Netzarchitektur besteht aus vier Layern wobei zusätzlich zur Zellausgabe, auch der Zellstatus der vorangegangenen Zelle mit verarbeitet werden kann. Um die Weitergabe des Zellstatus zu steuern werden sogenannte *Gates* verwendet. Sie bestehen aus sigmoidalen neuronalen Netzschichten und einer punktweisen Multiplikation. Diese Schicht steuert mit Werten zwischen 0 und 1, wie stark Informationen der einzelnen Layer mit in den Zellstatus einbezogen werden sollen. Abbildung 2.9 zeigt den beschriebenen Aufbau einer LSTM-Zelle.

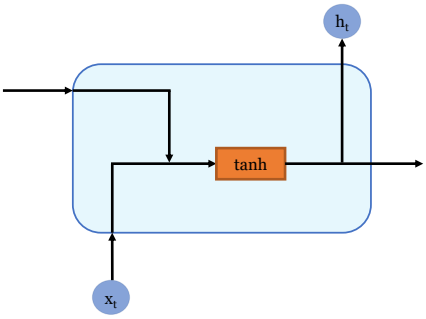


Abbildung 2.8: Detaillierter Aufbau einer RNN-Zelle. Die Eingabe ist stark mit dem Zellstatus der Vorgängerzelle verbunden. Eine Steuerung des Einflusses auf die Ausgabe ist nicht möglich. Abbildung in Anlehnung an Olah [29].

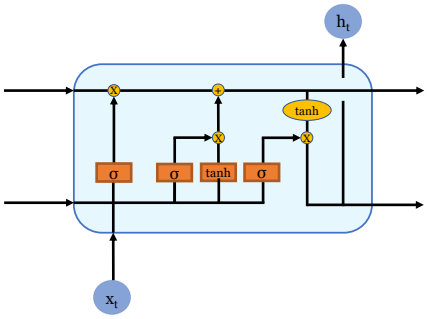


Abbildung 2.9: Detaillierter Aufbau einer LSTM-Zelle. Die obere horizontale Linie stellt den Zellstatus der Vorgängerzelle dar. Eine Steuerung des Einflusses kann mittels Gates erfolgen. Abbildung in Anlehnung an Olah [29].

2.5.5 Word Embedding

Ein Word Embedding stellt eine Repräsentation gelernter Wörter dar, bei der Wörter mit selber Bedeutung, eine ähnliche Repräsentation haben. Dabei handelt es sich um eine Klasse von Techniken, bei denen Wörter als Vektoren mit reellen Werten in einem vordefinierten Vektorraum dargestellt werden. Dieser Vektorraum besitzt typischerweise bis zu mehreren Hundert Dimension. Dies steht im Unterschied zu einer Sparse Darstellung der Wörter, die je nach Wortumfang mehrere Tausend oder Millionen Dimensionen besitzen kann. Die Verteilung der Vektoren wird anhand der Verwendung der Wörter gelernt, was es ermöglicht ähnliche Wörter zu ähnlichen Repräsentationen zuzuordnen. Daraus lässt sich anschließend die Bedeutung der Wörter erfassen. Word Embeddings lernen daher eine Vektordarstellung aus einem vordefinierten Textkorpus. Dieser Lernprozess kann entweder mit einem neuronalen Netz für eine bestimmte Aufgabe, wie dem Klassifizieren eines Textes oder durch einen Unsupervised Learning Algorithmus, bei dem Dokumentenstatistiken verwendet werden, erfolgen. In dieser Arbeit werden zwei dieser Techniken verwendet und erläutert. Dies ist zum einen der „Embedding Layer“ und zum anderen das Verfahren „word2vec“.

Embedding Layer

Bei einem Embedding Layer [18, S. 277 - 281] handelt es sich um einen Layer eines neuronalen Netzes, bei denen es nötig ist, dass die Eingabetexte vorbereitet werden indem jedes Wort einmalig codiert wird. Die Eingabevektoren werden zunächst mit zufälligen Zahlen initialisiert und durch Backpropagation-Algorithmen, im laufe des Lernprozesses, angepasst. Ein Embedding Layer stellt häufig die erste Schicht eines neuronalen Netzes dar.

word2vec

Word2vec [18, 208] [38, S. 190 - 192] ist ein Verfahren aus dem Bereich des Natural Language Processing (NLP). Der Algorithmus nutzt ein 2-Layer neuronales Netz, um aus einem Textkorpus Beziehungen zwischen darin enthaltenen Wörtern abzuleiten. Wie der Name des Verfahrens vermuten lässt, werden Wörter als Vektoren dargestellt. Diese Vektoren haben typischerweise eine Dimension größer 100. Die Ähnlichkeit von Wörtern in diesem Vektorraum wird mit der „Cosine Similarity“ berechnet. Wörter im Vektorraum die nah beieinander liegen haben also ähnliche Bedeutungen.

Das Verfahren kann beispielsweise dazu genutzt werden, um neue Beziehungen zwischen gegebenen Wörtern herzustellen. Beispielsweise kann so folgende Beziehung vorhergesagt werden: „Berlin“ verhält sich zu „Deutschland“,

wie „Paris“ zu „?“. Die Antwort die das Verfahren geben kann lautet in diesem Beispiel „Frankreich“. Dies ist nochmals in Abbildung 2.10 veranschaulicht. Weiterhin ist es möglich, zu einem Eingabewort ähnliche Wörter zu diesem ausgeben zu lassen. Dies erfordert einen zuvor durchgeführten Trainingsprozess, welcher hier kurz erläutert wird. Word2vec trainiert Wörter gegen andere Wörter, die im Eingabekorpus benachbart sind. Dies geschieht mittels zweier Verfahren: Continuous Bag of Words (CBOW) oder Skip-Gram. Beide Verfahren werden in den folgenden Kapiteln vorgestellt. An dieser Stelle wird die Skip-Gram Methode beschrieben, da sie in der Regel die genaueren Ergebnisse erzielt. Wenn ein zugewiesener Vektor eines Eingabewortes nicht den korrekten Wortkontext vorhersagt, müssen die Parameter des Vektors angepasst werden. Jeder Wortkontext im Korpus dient dabei als „Lehrer“ der Fehler an den Eingabevektor zurückmeldet. Die Vektoren werden mit einem Ähnlichkeitsmaß bewertet und so angepasst, dass sie näher beieinander liegen. Auf diese Weise liegen nach dem Trainingsprozess ähnliche Wörter im Vektorraum nahe beieinander.

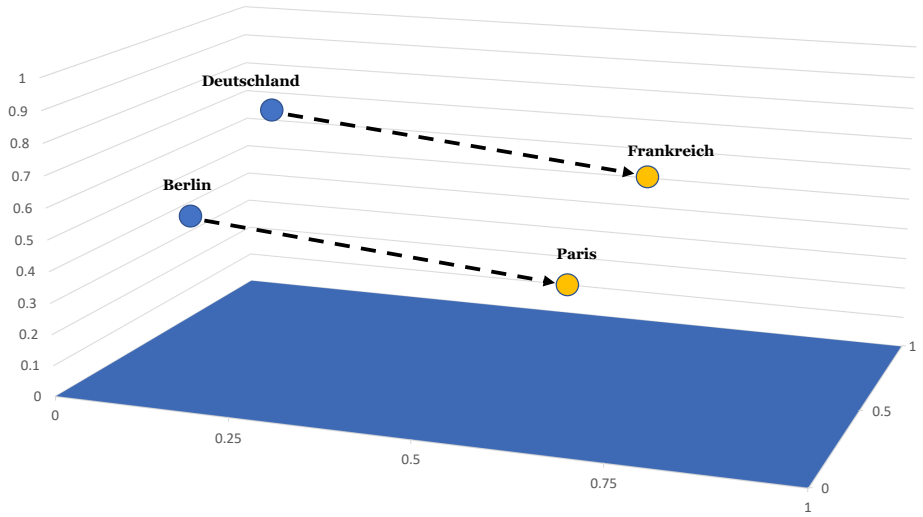


Abbildung 2.10: Beispiel zu Wortbeziehungen im Verfahren word2vec.

2.5.6 Hyperparameteroptimierung

Hyperparameteroptimierung oder Hyperparameter Optimization [38, S. 100 - 105] [18, S. 82], bezeichnet die Suche nach optimalen Hyperparametern. Ein Hyperparameter bezeichnet dabei alle Parameter die zur Steuerung eines Klassifikationsalgorithmus benötigt werden. Bergstra et al. [4] stellten in ihrer Arbeit „Random Search for Hyper-Parameter Optimization“ eine hilfreiche Methode vor, um optimale Hyperparameter zu finden.

2.6 Natural Language Processing

Natural Language Processing [18, S. 11 - 15] ist ein Teilgebiet der Informatik, Linguistik und künstlichen Intelligenz welches sich mit der Verarbeitung natürlicher Sprache durch Computer beschäftigt. Ziel dieses Teilgebiets ist es, Computer in die Lage zu versetzen, bspw. Dokumente die in natürlicher Sprache verfasst sind zu „verstehen“ und deren Kontext zu erfassen.

2.6.1 Tokenisierung

Die Aufgabe der Segmentierung von Text in relevante Wörter oder Einheiten, wird als „Tokenisierung“[18, S. 92 - 93] bezeichnet. Die so entstandenen Token können aus Wörtern, Nummern oder Satzzeichen bestehen und werden in der einfachsten Form durch die Trennung von Leerzeichen erzeugt.

Beispiel 1
The apple doesn’t fall far from the trunk.
|The|, |apple|, |doesn|, |’|, |t|, |fall|, |far|, |from|, |the|, | trunk|, |.|

Abschließend können diese Token in einem weiteren Verarbeitungsschritt betrachtet werden um zum Beispiel die Satzzeichen zu entfernen. Dieser einfache Prozess kann jedoch auch zu Problemen führen, wenn beispielsweise die Bedeutung durch die Tokenisierung verloren geht wie folgendes Beispiel verdeutlicht.

Beispiel 2
I live in New York.
|I|, |live|, |in|, |New|, |York|, |.|

In disem Beispiel wird das Wort „New York“ getrennt, obwohl es im normalen Sprachgebrauch nur zusammengeschrieben seine Bedeutung nicht verliert. Um dieses Problem der Tokenisierung zu lösen, wird häufig das N-Gramm Verfahren angewendet, welches in Kapitel 2.6.5 beschrieben wird.

2.6.2 Stemming

Stemming [18, S. 91] wird im Bereich des NLP genutzt um Wörter zu kategorisieren. Dazu werden einfache heuristische Methoden angewendet, bei dem das Suffix der entsprechenden Wörter entfernt wird. So wird beispielsweise aus dem Wort „Betten“, dessen Grundform „Bett“, wenn die Heuristik „en -> - “ angewendet wird. Bei anderen Worten wie beispielsweise „sehen“ würde diese Heuristik zu keinem guten Ergebnis „seh“ führen. Dies zeigt, dass dieses Vorgehen ein sehr komplexes Regelwerk benötigt um gute Ergebnisse erzielen zu können. Ein guter Regelsatz wurde in der Arbeit von Porter, 1980 [34] beschrieben, welcher auch heute noch verwendet wird.

2.6.3 Lemmatisierung

Bei der Lemmatisierung [18, S. 92] wird ebenfalls versucht, Wörter in ihre Grundform, dem Lemma zurückzuführen. Die Lemmatisierung beachtet dabei auch die Wortart des zu lemmatisierenden Wortes. So wird beispielsweise aus dem Wort „saw“ die Wörter „see“ oder „saw“ abgeleitet. Ein Verlust des Kontextes wird so, im Unterschied zum Stemming, vermieden. Für die Lemmatisierung werden häufig Datenbanken oder Listen eingesetzt, in denen die Wörter mit ihrer entsprechenden Grundform abgelegt werden. Dieser Prozess ist rechenintensiver als Stemming, sodass dieser weniger häufig eingesetzt wird. In der Praxis wird die Lemmatisierung durch die Nutzung von Programmbibliotheken bereitgestellt. Für die Programmiersprache Python stellt „spaCy“[10] eine gut gepflegte Lösung dar.

2.6.4 Stop Words

Stop Words oder auch Stoppwörter [18, S. 93] sind Wörter die in einer Sprache sehr häufig vorkommen. Das Entfernen dieser Wörter hat in der Regel keinen großen Einfluss auf das Verständnis des Satzinhalts. Es gibt verschiedene Listen mit Stop Words die je nach Einsatzzweck variieren können. Beispielsweise kann es in einem Anwendungsszenario hilfreich sein, Adjektive wie „nett, gut, schön“ zu entfernen, wohingegen diese Liste bei einer Stimmungsanalyse nicht so gut eingesetzt werden kann, da die Stimmungen verfälscht werden könnten. Eine Liste von Stop Words bietet die Python Bibliothek „spaCy“[10] welche eine Liste von einigen Hundert Stop Words (siehe Anhang A.2) aus der englischen Sprache bereitstellt.

2.6.5 N-Gramm

Während bei der herkömmlichen Wortrepräsentation nur einzelne Wörter repräsentiert werden, wird beim N-Gramm Verfahren [18, S. 93 - 94] eine Verbindung zwischen mehreren Wörtern hergestellt. Dabei werden N aufeinander folgende Fragmente zusammengefasst.

Beispiel 3
Satz: „What colleges are in New York City?“

- $N = 2$: What colleges, colleges are, are in, in New, New York, York City
- $N = 3$: What colleges are, colleges are in, are in New, in New York, New York City,

Mit diesem Verfahren kann der Kontext der zerlegten Wörter erhalten bleiben.

2.6.6 Continuous Bag of Words und Skip-Gram

Um einem Computer das Lernen von Text zu ermöglichen, ist es notwendig die Daten zuvor in ein Vektorformat zu transformieren, da dieses Format für Computer leichter zu verarbeiten ist. Diese Darstellung von Text in Vektor-Schreibweise wird „Word Representation“ genannt. Word Representations repräsentieren ein Wort in einem Vektorraum, sodass Wörter mit ähnlicher Bedeutung in diesem Vektorraum nahe beieinander liegen. Da das Vokabular einer natürlichen Sprache zu umfangreich ist um sie manuell zu labeln, wird ein effizientes Verfahren aus dem Bereich des unüberwachten Lernens benötigt. CBOW [26] wie auch Skip-Gram [26], sind solche unüberwachten Lernverfahren.

Skip-Gram [18, S. 211 - 212] wird genutzt um ähnliche Wörter (Kontext Wörter) zu einem gegebenen Eingabewort (Target Wort) zu finden, während CBOW [18, S. 209 - 210] für das Gegenteil genutzt wird. Hierbei wird aus verschiedenen Kontext Wörtern ein Target Wort gesucht. In Abbildung 2.11 ist dies schematisch dargestellt.

2.6.7 Document-Term Matrix

Eine Document-Term Matrix (Tabelle 2.4) ist eine mathematische Repräsentation von Dokumenten. Dabei werden die verschiedenen Dokumente in Spalten und die darin enthaltenen einmaligen Wörter in Zeilen aufgeteilt. Um die entsprechenden Werte der Zellen zu ermitteln, gibt es verschiedene

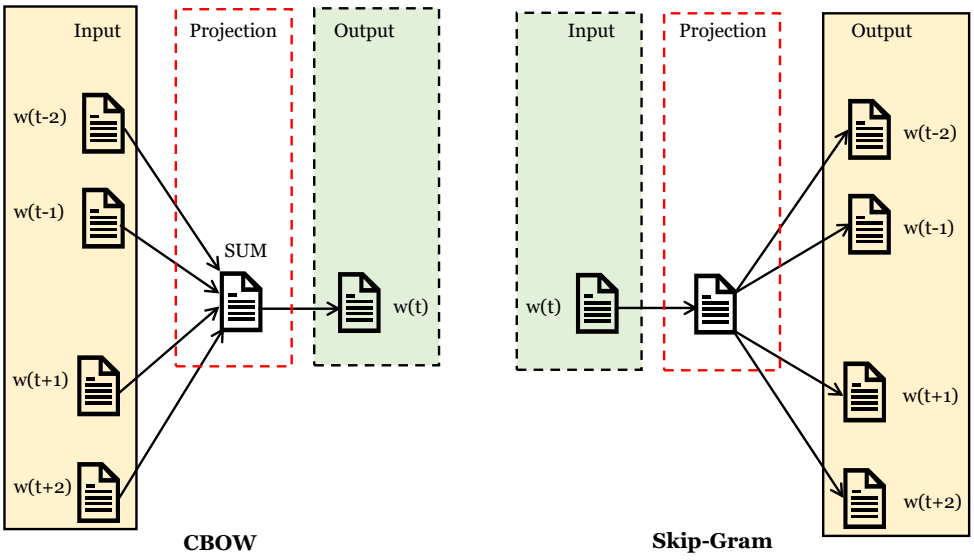


Abbildung 2.11: Architekturmodell CBOW und Skip-Gram. CBOW sagt ein Wort voraus, basierend auf mehreren Eingabewörtern. Skip-Gram sagt ähnliche Wörter zu einem Eingabewort voraus.

Wege. Zwei davon sind in Unterkapitel 2.6.8 und 2.6.9 beschrieben. Tabelle 2.4 zeigt eine Document-Term Matrix für drei Beispieldokumente:

Beispiel 4

1. Dokument: I like apples. I like green apples.
2. Dokument: I like your car. The car is green.
3. Dokument: The car is green like an apple.

2.6.8 Bag-of-Words model

Bag-of-Words [18, S. 95] ist eine Repräsentation von Text auf Grundlage der Auftrittshäufigkeiten eines Wortes in einem Dokument. So wird ein statistischer Aufbau eines Dokuments reflektiert. Eine Repräsentation eines Dokuments wird so als Liste mit den Häufigkeiten von einzigartigen Wörtern erstellt. Der Konvertierungsprozess von Dokumenten in eine „Document-term matrix“ wird „count vectorization“ genannt. In Tabelle 2.4 ist ersichtlich, dass das Bag-of-Words Modell stark von den Auftrittshäufigkeiten einzelner Wörter abhängt. Da in der englischen Sprache viele Wörter ohne großen Informationsgehalt auftreten, sogenannte Stop-Words (siehe Unterkapitel

	Dok1	Dok2	Dok3
I	2	1	0
like	2	1	1
apples	2	0	1
your	0	1	0
car	0	2	1
The	0	1	1
is	0	1	1
green	1	1	1
an	0	0	1

Tabelle 2.4: Document-Term Matrix

2.6.4), wird die Nutzung von Stop-Word Filtern empfohlen, umso Wörter mit höherem Informationsgehalt besser repräsentieren zu können. Detaillierte Untersuchungen führten Haddi et al. [12] in ihrer Arbeit „The Role of Text Pre-processing in Sentiment Analysis“ durch.

2.6.9 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) [36] [18, S. 96] ist ein statistisches Maß, welches evaluiert wie relevant ein bestimmtes Wort für ein Dokument in einer Sammlung von Dokumenten ist. Die Berechnung erfolgt durch die Bildung des Produktes zweier Metriken: wie oft ein Wort in einem Dokument vorkommt (Term Frequency) und die inverse Dokumentenhäufigkeit des Wortes über eine Sammlung von Dokumenten.

- Die **Term Frequency** $\#(t, D)$ gibt an, wie häufig ein Wort t in einem Dokument D vorkommt. Ist beispielsweise das Dokument D_i folgender Satz:

  Beispiel 5
  The green car is at the green traffic light.

  So ist $\#(green, D_i) = 2$.

  Die so berechneten Ergebnisse können zu Verzerrungen in langen Dokumenten führen. Um dies zu verhindern, kann das Ergebnis normalisiert werden indem die absolute Häufigkeit des Auftretens eines Terms in einem Dokument berechnet wird. Dazu wird die Häufigkeit eines Terms t in dem Dokument D durch die maximale Häufigkeit eines Terms in D geteilt. Das Ergebnis liefert nun die relative Vorkommenshäufigkeit:

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

Angewendet auf das Beispiel 5 ergibt sich folgende Berechnung:

$$tf(apple, D) = \frac{\#(apple,D)}{\max_{t' \in D} \#(t',D)} = \frac{2}{9}$$

- Die **inverse Dokumentenhäufigkeit** eines Terms über einer Sammlung von Dokumenten, bedeutet wie häufig oder selten ein Term t in der gesamten Dokumentenmenge vorkommt. Das Ergebnis der Berechnung gibt an, wie Relevant ein Term für die Dokumentensammlung ist. Häufig auftretende Terme (siehe 2.6.4) wie beispielsweise „the“, „it“ oder „a“ haben eine geringe Relevanz und somit einen geringen Informationsgehalt wie Terme die sehr selten in den Dokumenten vorkommen (detaillierte Informationen dazu in „Zipf's Law for Word Frequencies[9]“ und „Understanding inverse document frequency: on theoretical arguments for IDF[39]“). Diese Metrik wird berechnet, indem man die Gesamtzahl der Dokumente nimmt, sie durch die Anzahl der Dokumente, die ein Wort enthalten teilt und daraus den Logarithmus berechnet.

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

  Hierbei steht N für die Anzahl der Dokumente in der Sammlung und $\sum_{D:t \in D} 1$ für die Anzahl der Dokumente, die Term t beinhalten.

  Angenommen es existieren $N = 20$ Dokumente in denen der Term $t = apple$ fünf Mal vorkommt, so ist $idf(apple) = 1,39$. Wäre der Term t jedoch nur einmal in den 20 Dokumenten vorhanden, wäre $idf(apple) = 3$ und hätte somit einen höheren Informationsgehalt. Steigt der Wert der Berechnung des idf an, so hat das entsprechende Wort eine höhere Relevanz in den Dokumenten.

- Zusammen aus idf und tf ergibt sich das Gewicht **TF-IDF**.

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

  Beispiel 6
  Siehe Tabelle 2.5.

  Wie in Tabelle 2.5 zu sehen ist, weist ein höherer TF-IDF-Wert auf eine stärkere Beziehung zu dem Dokument hin. So kann eine bessere Gewichtung der Terme erreicht werden.

	Dok1	Dok2	Dok3
I	0,039	0,02	0
like	0	0	0
apples	0,39	0	0,02
your	0	0,053	0
car	0	0,39	0,02
The	0	0,02	0,02
is	0	0,02	0,02
green	0	0	0
an	0	0	0,053

Tabelle 2.5: TF-IDF Matrix zu Sätzen aus Beispiel 4.

Kapitel 3

Konzept

3.1 Ansatz

In diesem Kapitel wird der entwickelte Ansatz zur Identifizierung von Privacy Issues in App Store Reviews erläutert. Dieser Ansatz umfasst die Beschreibung einer Verarbeitungspipeline, angefangen beim Download der App Reviews aus den App Stores, bis hin zu fertigen Machine Learning (ML)-Modellen mit dem Privacy Issues identifiziert werden können. Der entwickelte Ansatz besteht aus insgesamt acht Phasen, welche in Abb. 3.1 abgebildet sind:

1. Datenerhebung (Review Crawling) - Kapitel 3.2
2. Datenvorverarbeitung - Kapitel 3.3
3. Training eines word2vec Wortmodells unter Verwendung der Daten aus Punkt 2. - Kapitel 3.4
4. Ermitteln von passenden Keywords mithilfe des erstellten Wortmodells - Kapitel 3.4
5. Analysieren von Reviews unter Verwendung der zuvor ermittelten Keywords - Kapitel 3.4
6. Manuelles Labeln der erhaltenen Reviews aus Punkt 5 - Kapitel 3.5
7. Trainieren eines Klassifikators / Deep Learning Modells - Kapitel 3.6 und 3.7
8. Analysieren weiterer Reviews mithilfe der Modelle aus Punkt 7.

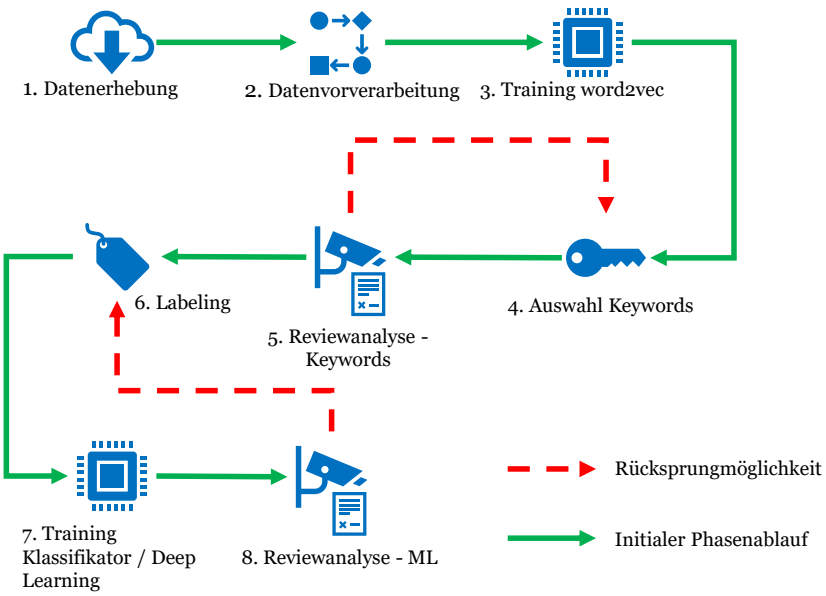


Abbildung 3.1: Übersicht der Phasen der Verarbeitungspipeline mit möglichen Rücksprungzielen.

3.1.1 Vorüberlegungen

Die von den Review-Crawlern erhaltenen Daten enthielten unter anderem den Review-Text, bestehend aus mehreren Sätzen. Daher wurde entschieden, bereits beim Speichern der Reviews den erhaltenen Review-Text in einzelne Sätze zu zerlegen, sodass es im Schritt des Labelns der Daten möglich ist, nicht nur den ganzen Review-Text als Privacy Issue zu kennzeichnen, sondern auch detailliert, einzelne Sätze labeln zu können. Mit diesem Vorgehen sollte eine bessere Performance der ML-Modelle erzielt werden. Ergebnisse zu diesem Vorgehen werden im Kapitel 5.1.2 beschrieben. Eine weitere Überlegung betraf das Auffinden von möglichen Privacy Issues in der großen Masse (82 Millionen Bewertungen) an Review-Daten. Laut einer bereits durchgeführten Untersuchung von McIlroy et al. [25] war nur mit $2,4\%$ im Apple App Store bzw. $0,7\%$ im Google Play Store, mit Privacy Issues in den heruntergeladenen Reviews zu rechnen. Durch diesen Umstand wäre es sehr aufwendig, die Reviews manuell nach Privacy Issues zu durchsuchen. Um das Auffinden dieser wenigen Reviews zu vereinfachen, wurde zunächst eine intelligente Keyword-Suche entwickelt und durchgeführt. Dieser Ansatz versprach eine effiziente Suche nach Privacy Issues, da dies schon von Vu et al. [45] genutzt wurde um verschiedene Issues in Review-Texten zu finden.

Das genaue Verfahren wird in Kapitel 3.4 beschrieben.

3.2 Datenerhebung

3.2.1 Crawling

Der erste Schritt des Lösungsansatz besteht darin, geeignete Reviews aus den App Stores von Google und Apple zu erheben, die in den anschließenden Schritten verarbeitet werden können. Ein wichtiges Kriterium ist hier die Wahl des Datenformats und der Datenstruktur. Anfängliche Überlegungen zur Nutzung einer Datenbank wurden verworfen, da die Daten schneller und einfacher verfügbar sind, wenn sie auf einem lokalen Datenträger abgespeichert sind. Diese Art des Zugriffs ist effizienter und für diesen Einsatzzweck besser geeignet. Die Wahl des Datenformats fiel daher auf das JavaScript Object Notation (JSON) Format, welches ein sehr einfaches und effizientes Datenformat darstellt. Des Weiteren werden ausschließlich Bewertungen in englischer Sprache betrachtet, da hier die Unterstützung durch Programmbibliotheken am größten ist. Dazu werden speziell Daten aus dem amerikanischen App Stores heruntergeladen. So unterscheiden sich die heruntergeladenen App-Bewertungen voraussichtlich von denen, die in den deutschen App Stores zur Verfügung stehen.

3.2.2 Google Play Store

Um Reviews aus dem Google Play Store herunterzuladen, wird eine externe Programmbibliothek Namens „google_play_scraper[1]“ verwendet. Auf eine eigene Entwicklung eines Crawlers wird verzichtet, da der zur Verfügung stehende Crawler im Funktionsumfang gut dokumentiert, für den benötigten Einsatzzweck ausreichend ist und einen aktiven Entwicklungsstatus aufweist.

3.2.3 Apple App Store

Wie beim Crawler für den Google Play Store, wird auch hier ein im Internet verfügbarer Crawler verwendet. Dieser Crawler „AppStoreScraper[2]“ ist ebenfalls in aktiver Entwicklung und im Funktionsumfang ausreichend.

[1] `https://github.com/JoMingyu/google-play-scraper` - Zugriff: 18.04.2021
[2] `https://github.com/cowboy-bebug/app-store-scraper` - Zugriff: 18.04.2021

3.3 Datenvorverarbeitung

Um die Daten im Hinblick auf die maschinelle Verarbeitung vorzubereiten, wurden sie in einer Verarbeitungspipeline oder Preprocessing-Pipeline bearbeitet. Die Vorverarbeitung der heruntergeladenen Reviews, umfasst folgende Schritte welche auch in Abb. 3.2 dargestellt sind:

1. Tokenisierung: Die Texte werden in einzelne Token zerlegt und weiterverarbeitet.

2. Rechtschreibkorrektur: In diesem Schritt werden eventuelle Rechtschreibfehler korrigiert. Dazu wird der Ansatz von Peter Norvig [33] verwendet.

3. Kleinschreibung: Alle Großbuchstaben werden in Kleinbuchstaben konvertiert.

4. Leerzeichen: Unnötige Leerzeichen werden entfernt.

5. Lemmatisierung: Alle Wörter werden zu ihrer Wortbasis zurückgeführt.

6. Satzzeichen: Es werden alle Satz- und Sonderzeichen[3] mittels der String Klasse[4] entfernt.

7. Stop Words: Entfernung aller Stop Words gemäß der Bibliothek spaCy[5]

Nach dieser Verarbeitungs-Pipeline werden die Daten an die nächsten Phasen übergeben. Einige der Phasen benötigen weitere Verarbeitungsschritte (word2vec, Klassifizierungsalgorithmen, Deep Learning Algorithmen) wie Vektorisierung, welche in den einzelnen Phasenbeschreibungen weiter erläutert werden. Da die Verarbeitung dieser Pipeline sehr viel Rechenzeit in Anspruch nehmen wird, empfiehlt es sich den Bearbeitungsstand abzuspeichern. Zu diesem Zweck kann beispielsweise die Programmbibliothek Pickle[6] unter Python genutzt werden. Auf diese Weise muss die Pipeline nicht mehrfach ausgeführt und verarbeitet werden, wenn etwaige Parameter an den folgenden Phasen geändert werden.

[3] (!"#$%&'() * +, −./ :; <=>?@[]^ _‘{}|~)
[4] String: `https://docs.python.org/3/library/string.html` - Zugriff: 17.04.2021
[5] spaCy: `https://spacy.io/api/language` - Zugriff: 17.04.2021
[6] `https://docs.python.org/3/library/pickle.html` - Zugriff: 15.04.2021

Abbildung 3.2: Übersicht der einzelnen Schritte in der Datenvorverarbeitungspipeline.

3.4 Datenvorselektion / Merkmalsselektion

Zur Selektion von Merkmalen die sich auf Privacy Issues beziehen, soll der Algorithmus „word2vec“[26] eingesetzt werden. Dieser wird auf allen heruntergeladenen Reviews trainiert um im Anschluss daran, passende Keywords zum Haupt-Keyword „Privacy Issue“ zu erhalten. Darauf folgt eine Sichtung der erhaltenen Keywords, die mit diesem Verfahren ermittelt wurden. Anschließend wird eine Liste mit etwa 30 Keywords erstellt, die im Kontext der Trainingsdaten als gut geeignet betrachtet werden, um in einer Auswahl von Reviews nach diesen Wörtern zu suchen. Diese Suche wird wiederum mit dem word2vec Algorithmus unterstützt, indem in den Reviews nicht nur nach den festgelegten Keywords gesucht wird, sondern ebenfalls nach Wörtern die dem Keyword ähnlich sind. So wird eine Sammlung von potenziellen Reviews erstellt, die ein Privacy Issue enthalten könnten. In der nächsten Phase werden diese Reviews dahingehend analysiert, ob es sich um Privacy Issues gemäß der Definition aus Kapitel 2.2.2 handelt.

3.5 Ground Truth

Um geeignete Trainingsdaten für die ML-Algorithmen zu erstellen, werden Reviews von mir gelabelt und in einem zweiten Schritt durch mindestens eine weitere Person überprüft. Die so erhaltenen Trainingsdaten dienen anschließend dazu, verschiedene ML-Modelle zu trainieren und deren Per-

formance zu ermitteln. Weiterhin werden alle erhaltenen Privacy Issues in einer Knowledge Base abgespeichert und kategorisiert (siehe Kapitel 3.8) Anschließend sollen die Modelle in der Lage sein, weitere Privacy Issues in weiteren Apps zu identifizieren.

3.5.1 Labeling

Die aus der Phase Merkmalsselektion 3.4 gewonnen Daten, werden in dieser Phase, von einem Menschen manuell gelabelt. Die Daten enthalten ein vollständiges Review unter Angabe des Review-Textes und des vergebenen Scores. Das Labeling erfolgt satzweise in geordneter Reihenfolge nach einzelnen Reviews. So bleibt der Kontext des zu labelnden Satzes erhalten und trägt somit zu einem besseren Verständnis bei.

Als Label wird ein binäres Label verwendet. „0“ falls es sich bei dem zu analysierenden Satz nicht um ein Privacy Issue handelt und „1“ für den Fall, dass es sich um ein Privacy Issue gemäß der Definition aus Kapitel 2.2.2 handelt. Da diese Arbeit vom Menschen eine sehr hohe Konzentration erfordert, wird ein eigenes Modul implementiert, welches das Labeling möglichst einfach und fehlerfrei durchführen lässt. Ein beispielhaftes Labeling wird in Tabelle 3.1 illustriert. Besondere Aufmerksamkeit ist auf Review Nummer 3 zu richten, da diese Bewertung aus zwei Sätzen besteht, bei dem es sich einmal um ein negatives und einmal um ein positives Label handelt. Diese Besonderheit kommt häufig vor und wird durch das differenzierte Labeling einzelner Sätze berücksichtigt.

Nr.	**Review-Text**	**Sätze**	**Label**
1	This application wants access to all my contacts and my personal photo folders. This is a clear violation of confidential personal information.	This application wants access to all my contancts and my personal photo foldes.	1
		This is a clear violation of confidential personal information .	1
2	I love this app. Sometimes it just crashes. It also consumes a lot of power.	I love this app.	0
		Sometimes it just crashes.	0
		It also consumes a lot of power.	0
3	The app is very useful. However, it sells my data to third parties.	The app is very useful.	0
		However, it sells my data to third parties.	1

Tabelle 3.1: Bewertungstexte mit Zerlegung in einzelne Sätze und dazugehörigem Label.

3.5.2 Cohens Kappa Statistik

Bei der Cohens Kappa Statistik handelt es sich um ein statistisches Maß für die Übereinstimmung zweier verbundener kategorialer Stichproben. Dabei wird der Grad der Übereinstimmung zweier Beurteiler / Rater bestimmt. Auf diese Weise wird die Interraterreliabilität gemessen.

Um die gelabelten Daten auf Übereinstimmung zu prüfen, muss mindestens eine weitere Person das bereits vorhandene Labeling kontrollieren. Dazu wird das Verfahren „Weighted Kappa“[7] von Cohen genutzt. Für diese Arbeit wird eine Mindestübereinstimmung von „Almost Perfect“ entsprechend ≥ 0.8 angestrebt [23]. In diesen Test werden 360 zufällig ausgewählte und bereits gelabelte Reviews einbezogen und dem Überprüfenden in gleicher Art, wie in Kapitel 3.5.1 beschrieben wurde, präsentiert. Anschließend sollen auftretende Unterschiede der beiden Label-Vorgänge analysiert und falls notwendig korrigiert werden. So soll eine möglichst einheitliche Masse von gelabelten Daten erzeugt werden.

3.6 Klassifikation

Durch die Auswahl an möglichen Merkmalen, welche mittels Labeling bereits erfolgt ist, kann mit diesen Daten nun ein ML-Modell trainiert werden. Dieses Modell soll anschließend in der Lage sein, weitere Merkmale in neuen Daten zu erkennen und auszugeben. Diese Merkmale sind verschiedene Arten von Privacy Issues, wobei hier streng darauf zu achten ist, dass die Auswirkungen eines Privacy Issues negativ sein müssen. Dies stellt eine besondere Herausforderung an die Algorithmen dar, da in den Reviews oft positiv formulierte Privacy Themen formuliert werden. Der Klassifikationsalgorithmus muss also in der Lage sein, positive und negative Privacy Issues zu unterscheiden. Um ein möglichst gutes Modell zu erhalten, werden einzelne Klassifikationsalgorithmen zu einem Ensemble kombiniert und im weiteren Verlauf mit einem Stacking Classifier verglichen. Es wird eine binäre Klassifizierung implementiert, da für diese Arbeit das Unterscheiden der Review-Texte in *Privacy Issues* und *kein Privacy Issue* genügt.

3.7 Deep Learning

Neben dem Einsatz von Klassifizierungsalgorithmen werden auch Algorithmen aus dem Bereich des Deep Learnings verwendet. Hier soll der Fokus darauf gelegt werden, ob die Verwendung von Deep Learning Algorithmen

Vorteile bei der Klassifizierung bietet. Es wird ein Vergleich zwischen FNNs, CNNs, RNNs, Ensemble und Stacking Classifier gezeigt.

3.8 Knowledge Base

Im letzten Schritt wird eine Datenbank mit identifizierten Privacy Issues aus Review-Texten eingerichtet. Diese soll eine Knowledge Base darstellen und einen schnellen und übersichtlichen Zugang zu den betroffenen Texten ermöglichen. Es wird eine rationale Datenbank in der Structured Query Language (SQL) Syntax verwendet, welche alle verfügbaren Metainformationen zu den identifizierten Privacy Issues beinhaltet. Dazu gehören neben dem betroffenen Text, der App Name, App ID, Zeit und Datum, Score, Herkunfts-Store und Store-Kategorie der App. Des Weiteren wird jedem Text der als Privacy Issue klassifiziert wird, eine oder mehrere Kategorien zugeordnet. Diese Kategorien sollen aus den gefundenen Privacy Issues abgeleitet und anschließend zugeordnet werden. Mehrfachzuordnungen von Kategorien sind möglich.

3.9 Überlegungen und Lösungsalternativen

3.9.1 Crawler

Da von den beiden App Store Betreibern, Google und Apple, keine definierte und frei zugängliche Schnittstelle zur Verfügung gestellt wird, müssen Bewertungen auf andere Weise heruntergeladen werden. Die kann bspw. durch Browsersimulatoren wie Selenium[7] erfolgen. Dieser simuliert den Aufruf der App Store Webseiten und lädt die dazugehörigen Bewertungen herunter. Daher wurde überlegt, diese Art Review-Crawler selbst zu entwickelt und zu implementieren oder einen öffentlich verfügbaren Crawler einzusetzen. Die Entscheidung fiel auf einen öffentlich verfügbaren Crawler und wurde aus zwei Hauptgründen getroffen:

1. Grund: Der Quellcode der beiden App Stores ändert sich sehr schnell, sodass ein eigener Crawler ständig angepasst werden müsste. Dies bedeutet unter anderem einen großen Aufwand, der bei der Verwendung eines öffentlichen Crawlers nicht anfällt.

2. Grund: Versuche einen eigenen Review-Crawler mittels Selenium zu entwickeln zeigten, dass die Geschwindigkeit, mit der Bewertungen

[7] Selenium: `https://selenium-python.readthedocs.io/` Zugriff: 04.05.2021

heruntergeladen werden können, nicht ausreichend war um die hohe Anzahl (ca. 82 Mio) Bewertungen herunterzuladen. Diese Einschränkung war bei den öffentlichen Crawlern nicht gegeben.

3.9.2 Sprache

Eine andere Überlegung galt der Wahl der Review-Sprache und damit auch der Wahl des länderspezifischen App Stores. Die Wahl fiel aus den folgenden Gründen auf Englisch und damit auf den amerikanischen App Store:

1. Grund: Die Verfügbarkeit und Kompatibilität von Tools im Bereich NLP ist für die englische Sprache größer, als für andere Sprachen wie Deutsch.

2. Grund: Die Vergleichbarkeit des Ansatzes und der damit erstellten ML-Modelle und deren Performance-Daten, ist auf Grund der hohen Anzahl an wissenschaftlichen Arbeiten, die sich auf die Verarbeitung von englischer Sprache beziehen, deutlich höher im Gegensatz zu anderen Sprachen.

3.9.3 Contrast Score

Beim Erarbeiten des Konzepts gab es Überlegungen bezüglich dem Einbeziehen positiver (Score 5) Bewertungen. In der Arbeit von Vu et al. [45] wurde ein „contrast score“ genutzt, um die Negativität eines Keywords zu bestimmen. Dies wurde in diesem Ansatz zunächst ebenfalls implementiert. Da im Verlauf der Implementierungsphase klar wurde, dass durch diese Selektion auch wichtige Datenschutz- und Privatsphäreverletzungen verloren gingen, die laut „contrast score“ von den Verfassern positiv gemeint waren, wurde diese Art der Filterung wieder deaktiviert. Auch sollte den eingesetzten Klassifizierungsalgorithmen die Möglichkeit gegeben werden, Negativität und Positivität selbst zu erlernen.

3.10 Zusammenfassung

Um den hier vorgestellten Ansatz realisieren zu können, ist Wissen aus vielen einzelnen Bereichen der Informationstechnologie nötig. Es werden unter anderem Praktiken aus den Bereichen NLP, ML, Deep Learning und Datenbanken verwendet. Aktuell existieren mehrere Ansätze um mittels

Algorithmen, Texte auf bestimmte Merkmale zu untersuchen. Jedoch existiert kein Verfahren, welches speziell auf die Thematik der Privatsphäreverletzungen in Apps aus den jeweiligen Stores abzielt. In diesem Bereich kann der hier beschriebene Ansatz verwendet und weiterentwickelt werden. Zusammenfassend besteht dieser Ansatz aus aufeinanderfolgenden Phasen, welche in Abbildung 3.3 dargestellt werden. In Phase 2 - Analyse können unterschiedliche Algorithmen verwendet werden um die Performance der Modelle zu verbessern. Die Algorithmen mit der besten Performance, werden in Kapitel 5 vorgestellt und besprochen.

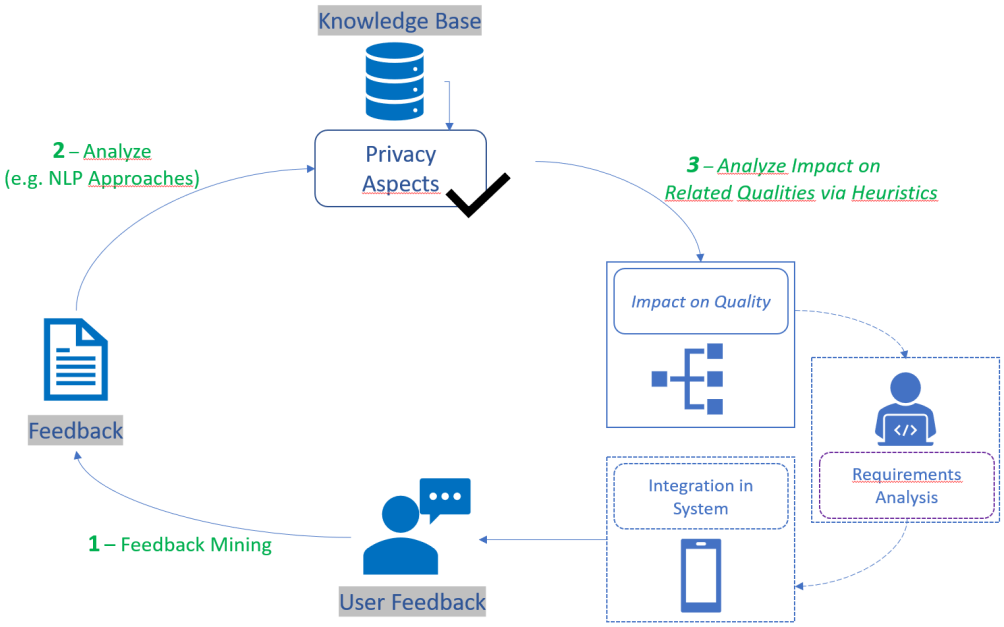


Abbildung 3.3: Schematischer Aufbau des Konzepts.

Kapitel 4

Implementierung

4.1 Allgemein

Um den Lösungsansatz aus Kapitel 1.3 und das damit verbundene Konzept aus Kapitel 3 umzusetzen, wurde ein Programm in der Programmiersprache Python v3 geschrieben. Python bietet eine sehr gute Auswahl an Bibliotheken die im Bereich NLP und ML für diese Arbeit benötigt werden. Anhang A.1 listet alle verwendeten Programmbibliotheken einschließlich der verwendeten Version und einer Beschreibung auf. Die folgenden Unterkapitel beschreiben die genaue Implementierung der einzelnen Module und geben einen Überblick über die Architektur des entwickelten Programms. Weiterhin werden einige Probleme sowie deren Lösung erläutert, die während der Implementierungsphase aufgetreten sind.

4.2 Architektur

Das entwickelte Programm ist in mehrere Module untergliedert. Abbildung 4.1 zeigt den Aufbau des entwickelten Programms nach Paketen und Klassen. Die verschiedenen Phasen sind dabei den Klassen farbig zugeordnet. Generell wird der gesamte Prozess in acht Phasen unterteilt, welche bereits in Kapitel 3.1 beschrieben wurden. In den Phasen 3 und 7 werden Modelle trainiert, die nicht bei jedem Phasendurchlauf erneut trainiert werden müssen. Faktisch können diese Modelle weiterverwendet werden, wenn ein neuer Phasendurchlauf ausgeführt wird. Die Rücksprungpfeile deuten an, dass die Phasen 8 und 5 wiederholt ausgeführt werden können. Das heißt, wenn in Phase 8 neue Bewertungen analysiert und gelabelt wurden, kann ein neues Modell in Phase 7 trainiert werden. Auch ist es möglich, die Review-Analyse aus Phase 5 erneut zu durchlaufen um bspw. neue Keywords in die Auswahl mit

aufzunehmen. Dabei muss kein neues word2vec Modell trainiert, sondern das bestehende Modell kann beibehalten und wiederverwendet werden. Die Ausführung des Programms beziehungsweise der einzelnen Module erfolgt nacheinander auf einer Textkonsole. Dabei können einzelne Phasen auch wiederholt ausgeführt werden und sind voneinander unabhängig, sofern die korrekte Reihenfolge eingehalten wird. Die Erstellung eines Graphical User Interface (GUI) war explizit nicht Bestandteil dieser Arbeit. Eine Möglichkeit die hier entwickelten Module durch eine GUI zu erweitern, wird im Kapitel 7.2 beschrieben.

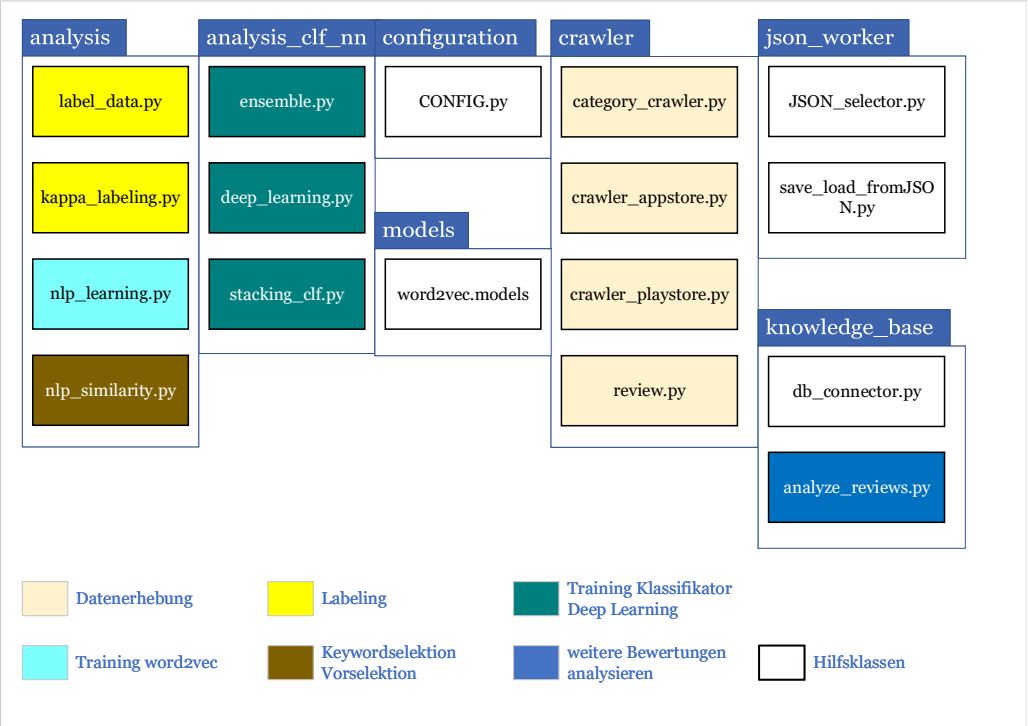


Abbildung 4.1: Übersicht der Modularchitektur mit farbiger Zuordnung der Phasen.

4.3 Phasenbeschreibungen

In den folgenden Unterkapiteln erfolgt die Beschreibung der einzelnen Phasen gemäß Abbildung 3.1. Jede Phase stellt dabei ein eigenes Modul des entwickelten Programms dar.

4.3.1 Datenerhebung

Um alle weiteren Phasen durchlaufen zu können, mussten zunächst Bewertungen vorhanden sein auf die schnell und einfach zugegriffen werden konnte. Da weder der Google Play Store noch der Apple App Store über definierte Schnittstellen für den Download von Bewertungen verfügen, wurden sogenannte Review Crawler verwendet. Zwei solcher Review Crawler (siehe Unterkapitel 3.2.2 und 3.2.3) wurden als externe Programmbibliothek genutzt und benötigen als Eingabe, lediglich einen App-Namen oder eine App-ID, um den Download der zur Verfügung stehenden Bewertungen zu initiieren. Weitere Parameter, wie die Menge der herunterzuladenden Bewertungen, oder die Angabe eines bestimmten Scores sind ebenfalls möglich. Die Ausgabe dieser Review Crawler besteht anschließend aus mindestens dem Bewertungstext, dem Erstellungsdatum, einem Score und dem App-Namen bzw. der App-ID. Diese Informationen wurden anschließend im JSON Format für jede App auf dem lokalen Datenträger abgespeichert. Um eine gemischte Auswahl an Apps zu erhalten, wurden alle Apps einer bestimmten Store-Kategorie heruntergeladen. Die Auswahl der Kategorien ist in Tabelle 4.1 zu sehen. Kategorien aus dem Google Play Store besitzen weiterhin Unterkategorien, bei denen es möglich ist, dass eine App auch mehrfach in den Unterkategorien oder Kategorien vorhanden ist. In diesem Fall wurde die App nicht mehrfach betrachtet und heruntergeladen, sondern nur bei der erstmaligen Ausführung berücksichtigt. So wurden Dopplungen von Bewertungen vermieden.

Jeder Eintrag in einer JSON-Datei entspricht folgender Datenstruktur:

```
1 {
2     appId = "",
3     score = "",
4     datetime = "",
5     content = "",
6     sentences[{sentence:"", isPrivacyIssue:0/1}],
7     privacy_issue = 0/1
8 }
```

Listing 4.1: Aufbau der Datenklasse „Review“.

Die Attribute „sentences“ und „privacy_issue“ benötigen einer genaueren Erläuterung. Beim Attribut „sentences“ handelt es sich um eine Liste einzelner Sätze aller vorkommenden Sätze aus dem Attribut „content“, welche automatisch beim Herunterladen der Bewertung erstellt wird. Zu jedem Satz existiert weiterhin das Attribut „isPrivacyIssue“ welches zunächst mit „None“ initialisiert wird. In Phase 6 dem Labeling, wird diesem Attribut

eine „1“ oder „0“ zugewiesen, je nachdem ob in dem zu labelnden Satz eine Privatsphäreverletzung angesprochen wurde oder nicht. Details werden in Unterkapitel 4.3.6 beschrieben. Nach Abschluss der Downloadvorgänge, sind alle Bewertungen zu einer App in der beschriebenen Datenstruktur in einem Ordner verfügbar. Die Unterordnerstruktur bildet dabei die Kategorien bzw. Unterkategorien der heruntergeladenen Apps ab.

Store	Kategorie	Anzahl Apps	Anzahl Bewertungen
AS	Biggest	5	82.701
	Education	239	143.306
	Finance	240	62.272
	Games	240	1.102.147
	Health Fitness	240	93.999
	Lifestyle	240	222.110
	Medical	240	35.916
	Navigation	240	20.191
	Productivity	240	130.545
	Shopping	239	228.690
	Travel	240	172.970
	Utilities	240	255.241
PS	Auto and Vehicles	103	502.217
	Communication	165	33.015.176
	Family	269	428.930
	Finance	254	15.019.197
	Game Educational	165	998.698
	Health and Fitness	126	3.260.293
	Maps and Navigation	259	4.094.925
	Medical	595	716.187
	Social	194	15.769.883
	Travel and Local	290	6.110.107

Tabelle 4.1: Übersicht der Store Kategorien aufgeteilt nach App Store (AS: Apple App Store, PS: Google Play Store).

In dieser Phase wurden insgesamt 82.465.848 Bewertungen aus 5.063 Apps heruntergeladen. Davon entfielen 2.420 Apps mit 79.915.613 Bewertungen auf den Google Play Store und 2.643 App mit 2.550.088 Bewertungen auf den Apple App Store. Abb. 4.2 zeigt die Verteilung der Apps auf die beiden App Stores und deren Bewertungen. Genauere Zahlen zur Verteilung der Anzahl von heruntergeladenen Bewertungen auf die einzelnen Kategorien, sind in Tabelle 4.1 zusammengefasst.

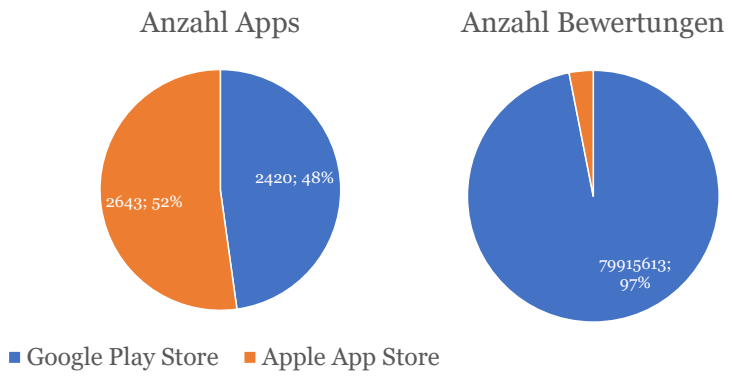


Abbildung 4.2: Übersicht heruntergeladener Apps bzw. Bewertungen. Links Anzahl heruntergeladener Apps, rechts Anzahl heruntergeladener Bewertungen.

Store	Name	Kategorie
AS	House of Restoration	Education
	Beyond Polish	Shopping
PS	Current - The Bank for Modern Life	Finance
	Supermarket	Game Educational
	SBB Mobile	Maps and Navigation
	JD	Travel and Local
	IRCTC Train Ticket	Travel and Local
	Lime	Travel and Local
	Wyndham Hotels	Travel and Local

Tabelle 4.2: Apps von denen, aus nicht bekannten Gründen, keine Bewertungen heruntergeladen werden konnten.

Einige App-Bewertungen konnten auch nach wiederholten Versuchen nicht heruntergeladen werden und sind in Tabelle 4.2 aufgeführt. Der Grund hierfür war nicht ermittelbar. Des Weiteren konnten auch die Bewertungen der Apps „Facebook“ und „WhatsApp“ aus dem Google Play Store nicht komplett heruntergeladen werden. Dies ist auf die verfügbare Anzahl an Bewertungen zurückzuführen, da diese mit Facebook: 109.165.668 Bewertungen[1] und WhatsApp: 133.826.116 Bewertungen[2] sehr groß ist. Der verwendete Review Crawler war nach wiederholten Verbindungsabbrüchen nicht in der Lage, zwischen bereits heruntergeladenen und noch herunterzuladenden Bewertungen zu unterscheiden und startete den Downloadprozess funktionsbedingt von vorn. Um nicht auf alle Bewertungen dieser beiden Apps verzichten zu müssen, wurde jeweils ein Teil der Bewertungen heruntergeladen (Facebook: 2.696.359 Bewertungen, WhatsApp: 4.386.248 Bewertungen).

Probleme

Im Google Play Store wird der geografische Standort der Nutzer, anhand der öffentlichen IP-Adresse ermittelt. So werden den Nutzern nur die Apps angezeigt, die in seiner Region zur Verfügung stehen. Speziell in Deutschland, ließ sich diese Einstellung der Lokation nicht manuell ändern, wie es in anderen Ländern möglich ist. Auch die Auswahl der verfügbaren Apps unterschied sich stark zwischen dem deutschen und amerikanischen App Store. Da bei dieser Arbeit nur englischsprachige Bewertungen untersucht werden sollten, stellte dies ein Problem dar, welches gelöst werden musste. Um dieses Geoblocking zu umgehen, wurde eine Software eines kommerziellen Virtual Private Network (VPN)-Anbieters verwendet, um einen VPN-Tunnel zu einem entfernten Server in den USA herzustellen. So konnte der Zugriff auf den amerikanischen Google Play Store hergestellt werden und die entsprechenden Bewertungen der ausgewählten Apps heruntergeladen werden.

4.3.2 Datenvorverarbeitung

Gemäß Kapitel 3.3 wurden alle Texte der heruntergeladenen Bewertungen in einer Preprocessing-Pipeline vorverarbeitet. Dazu wurde die Programmbibliothek spaCy[3] eingesetzt, da diese eine breite Funktionalität und gute Performance [42] für diesen Zweck bereitstellt. Dieser Schritt wurde durchgeführt, um die Eingabe für die im folgenden angewendeten ML-Algorithmen zu optimieren. Mit dieser Vorgehensweise wurde beabsichtigt, die Performance

[1] Stand: 18.04.2021
[2] Stand: 18.04.2021
[3] spaCy: `https://spacy.io/api/language` - Zugriff: 19.04.2021

der ML-Algorithmen zu verbessern. Chandrasekar et al. [6] zeigten in ihrer Arbeit den positiven Effekt dieses Verfahrens am Beispiel eines Naive-Bayes Klassifikators. Grundlegende Vorgehensweisen bei der Vorverarbeitung von Texten, erläuterten Kannan et al. [19] in ihrer Arbeit. Abbildung 4.3 zeigt die einzelnen Verarbeitungsschritte der implementierten Preprocessing-Pipeline.

Abbildung 4.3: Übersicht der Verarbeitungsschritte der implementierten Preprocessing-Pipeline.

In dieser Phase wurde außerdem die Auswirkung verschiedener Pipelines auf die Performance der in Phase 7 verwendeten Algorithmen verglichen. So wurde bspw. analysiert, ob die Verwendung der Rechtschreibkorrektur Einfluss auf die Performance der Algorithmen hat. Die Ergebnisse dieser Analyse werden in Kapitel 5.1.2 erläutert.

4.3.3 Training word2vec

Die Implementierung des word2vec [26][27] Algorithmus erfolgte unter Verwendung der Programmbibliothek Gensim[4] die unter Python frei verfügbar ist. Als Eingabe für den Algorithmus wurden alle verfügbaren Bewertungstexte aus der Datenerhebungsphase genutzt, welche anschließend in Vektor-Repräsentation als Eingabe dienten. Diese Eingabe umfasste insgesamt 29.578.550 Sätze mit 281.189.982 Wörtern, was eine durchschnittliche Satzlänge von 9,5 Wörtern ergab. Der bereinigte Wortkorpus umfasste 136.819 einzigartige englische Wörter, mit denen das Modell trainiert wurde.

Word2vec verfügt über zwei Lernalgorithmen, die zur Training eines Modells, eingesetzt werden können und in Kapitel 2.6.6 beschrieben wurden. Beim training des hier verwendeten Modells, wurde der CBOW Lernalgorithmus verwendet. Das Modell wurde über 30 Epochen trainiert und als Ausgabe eine Dimensionsgröße von 300 gewählt. Das so trainierte word2vec Modell war anschließend in der Lage, zu einem vorgegebenen Keyword, weitere ähnliche Wörter auszugeben. In den folgenden Phase wurde die Möglichkeit genutzt, eine bestimmte Anzahl ähnlicher Wörter, zu einem Eingabewort, ausgeben zu lassen.

[4]Gensim.models Word2vec: `https://radimrehurek.com/gensim/models/word2vec.html` Zugriff: 20.04.2021

4.3.4 Keyword-Selektion

Um eine Menge geeigneter Keywords zu erhalten, wurde das zuvor trainierte word2vec Modell verwendet. Hierzu war es nötig, den Parameter „similarity“ anzugeben, der einen bestimmten Vektorabstand (in diesem Fall der Kosinus Abstand) der Wortvektoren vorgibt. Dieser Parameter kann im Intervall von $[0; 1]$ gewählt werden, wobei der Wert 1 das Eingabe-Keyword selbst beschreibt und nur dieses Wort ausgegeben wird. Je weiter sich der Wert verkleinert, desto unähnlicher werden die Ausgabewörter in Relation zum Eingabewort. Um eine geeignete Auswahl an ähnlichen Wörtern zu erhalten, wurde dieser Parameter durch Versuche auf $0,7$ eingestellt. Als initiales Eingabewort, wurde das Bigramm „personal_information“ gewählt. Die daraus resultierenden Wörter sind in Abbildung 4.4 aufgeführt. Mit diesem Vorgehen können auch weitere Themenbereiche untersucht werden, indem ein anderes initiales Keyword genutzt wird. Soll bspw. der Bereich „Security“ in App Bewertungen untersucht werden, könnte das initiale Keyword „security“ genutzt werden und auf dieselbe Weise weiter verfahren werden, wie in den weiteren Kapiteln beschrieben.

Alle folgenden Keywords leiteten sich aus dem initialen Startwort „personal_information“ ab. So entstand eine Art Wörterbaum der verschiedene Begrifflichkeiten, im Kontext der zuvor genutzten Bewertungen für das word2vec Modell hervorbrachte. Für die nächste Verarbeitungsphase „Vorselektion der Bewertungen“ wurde eine eingegrenzte Menge an Keywords ausgewählt. Dabei dienten die nachfolgenden Begriffe als Startpunkt für weitere ähnliche Begriffe:

personal_information, disclosure, confidential_information, pii, security, privacy_concern, privacy

Zu diesen Begriffen wurden die 14 ähnlichsten Wörter ermitteln, was eine Auswahl von $7 * 14 = 98$ möglichen Wörtern ergab, welche detailliert in Anhang B.1 aufgelistet sind. Aus diesen 98 Begriffen zusätzlich zu den sieben initialen Keywords wurden 29 Begriffe, die in Tabelle 4.3 aufgelistet sind, ausgewählt um anschließend Bewertungen auf diese Begriffe zu analysieren. Des Weiteren wird die Wortherkunft in Abbildung 4.4 in einem Baumdiagramm dargestellt. Die Suche nach allen gefundenen 98 Keywords wäre nicht zielführend gewesen, da viele Begriffe doppelt vorkamen oder nicht der Definition eines Privacy Issues gemäß Kapitel 2.2.2 entsprachen. Ein interessanter Punkt der bei der Sichtung der Keywords auffiel, ist das Auftreten von Rechtschreibfehlern. So wurde das Wort „privacy“ von den Autoren der Bewertungstexte sehr häufig falsch geschrieben

(*privicy, privecy, pricacy*). Diese falsch geschriebenen Wörter wurden mit zu den finalen Keywords, dem Keyword-Katalog, aufgenommen. Eine weitere interessante Buchstabenkombination im Zusammenhang mit dem Keyword „personal_information“ war „ssn“. Diese Abkürzung steht im amerikanischen Raum für *Social Security Number* und ist mit der deutschen Sozialversicherungsnummer zu vergleichen. Dieser Begriff ist im Bezug auf Privatsphäre und Datenschutz sehr wichtig, da einige Apps diese schutzbedürftige Nummer verwenden möchten und viele Nutzer der Verwendung zustimmen, aber nicht wissen wie diese Daten anschließend verwendet und verarbeitet werden. Aus diesem Grund wurde auch dieser Begriff zum Keyword-Katalog hinzugefügt. Weitere wichtige Begriffe waren „pii“ was für *personal identifiable information* steht, „gdpr“ was für *General Data Protection Regulation* steht und die europäische Datenschutzverordnung darstellt und „hipaa“ was für *Health Insurance Portability and Accountability Act* steht und die Datenschutzvorgaben im Gesundheitsbereich regelt. In all diesen Bereichen, stellt Datenschutz und Privatsphäre eine zentrale Rolle dar, weshalb auch diese Suchbegriffe zum Keyword-Katalog hinzugefügt wurden. Der anschließende Analyseprozess der Bewertungen der unter Verwendung der ausgewählten finalen Keywords durchgeführt wurde, ist in Kapitel 4.3.5 erläutert.

personal_information	confidential_information	violate_privacy
pii	personally_identifiable	gdpr
respect_privacy	protect_privacy	privacy
privicy	privecy	pricacy
privacy_concern	privacy_protection	privacy_policy
privacy_violation	breach_privacy	permission
private_confidential	personally_identifiable	confidential
hipaa	identifiable_information	divulge
breach	ssn	steal_identity
unfettered_access	breeche	

Tabelle 4.3: Inhalt des finalen Keyword-Katalogs mit 29 ausgewählten Keywords zur Vorselektion von Bewertungen.

4.3.5 Vorselektion der Bewertungen

Ziel dieser Phase ist es, eine geeignete Vorauswahl aus den ca. 82 Millionen heruntergeladenen Bewertungen zu erhalten, die mit einer hohen Wahrscheinlichkeit Privacy Issues beinhalten. Die so erhaltenen Bewertungen sollen anschließend in der folgenden Phase manuell gelabelt werden. Zu diesem

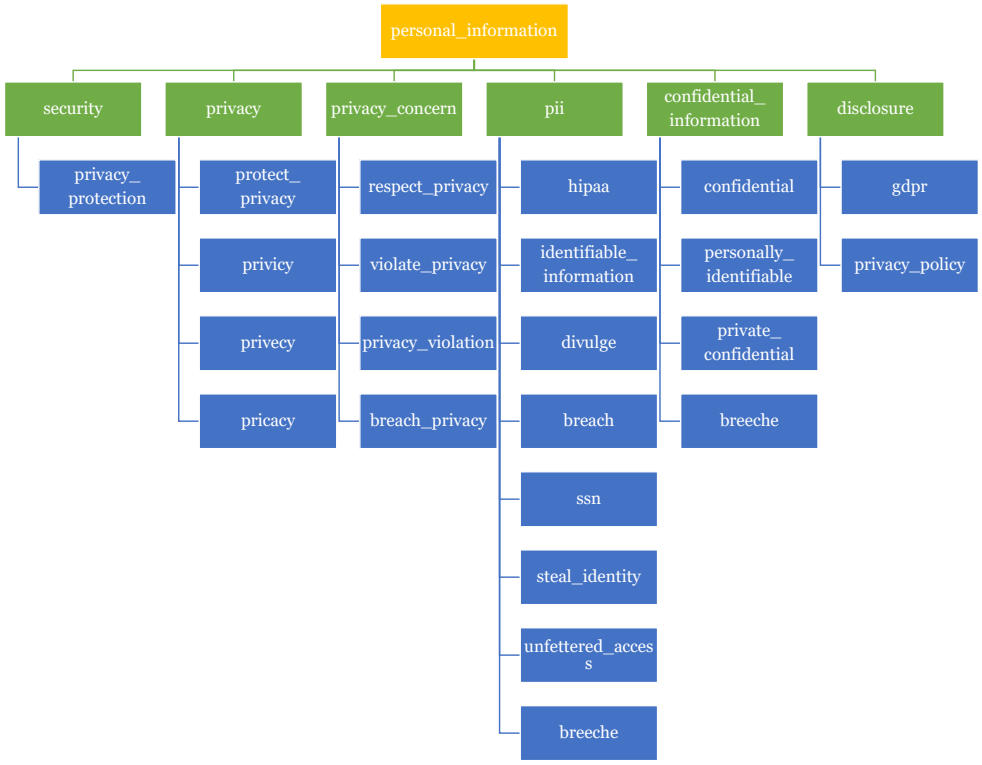


Abbildung 4.4: Darstellung der 29 ausgewählten Keywords in einer Baumstruktur. So kann erkannt werden, welche Wörter von welchen Wörtern abstammen beziehungsweise abgeleitet wurden.

Zweck wurde ein Algorithmus entworfen, der nicht nur nach den exakten Keywords aus Tabelle 4.3 mittels einfachem „Wortvergleich“ sucht, sondern vielmehr eingehende Bewertungstexte, Wort für Wort mit dem Keyword-Katalog vergleicht, indem immer auch ähnliche Begriffe zum Eingabewort betrachtet werden. Um diese ähnlichen Begriffe zu finden, wurde erneut das word2vec Modell aus Kapitel 4.3.3 verwendet, um eine intelligente Keyword-Suche zu implementieren, welche in Listing 4.2 verdeutlicht ist.

```
1
2 function getReviewsFromSimilarity(review,
    ↪ keyword_katalog, word2vec_model)
3     cleaned_reviewtext = clean(review.text)
4     for keyword in keyword_katalog:
5         for word in cleaned_reviewtext.token
6             if word2vec_model.similarity(keyword,
                ↪ word) >= 0.7 do
```

```
7                     return review, keyword
8                 else
9                     continue
```

Listing 4.2: Algorithmus zur intelligenten Suche von Bewertungen mit Hilfe des Keyword-Katalogs. Der Funktionsaufruf clean() führt das Preprocessing des Textes durch.

Diese Funktion benötigt als Parameter eine Bewertung, den Keyword-Katalog und das zuvor trainierte word2vec Modell. Der Parameter „similarity“ wurde, wie auch bei der Keyword-Suche aus Kapitel 4.3.4 erneut mit 0.7 festgelegt. Die Abbildung 4.5 stellt den entwickelten Algorithmus zur intelligenten Keyword-Suche grafisch dar. Der Eingabetext „*WhatsApp ´s new privacy policy is so bad it migh be illegal.*“ wird zunächst in der Preprocessing-Pipeline verarbeitet. Daraus entsteht der bereinigte Satz „*WhatsApp new privacy policy bad illegal*“ welcher in Token zerlegt ist. Anschließend wird die Keyword-Suche Wort für Wort durchgeführt, was hier am Beispiel des Wortes *privacy* beschrieben wird. Zu diesem Wort werden, im zuvor trainierten word2vec Modell, alle ähnlichen Begriff die eine Ähnlichkeit von ≥ 0.7 besitzen gesucht. Anschließend werden diese Worte mit dem Keyword-Katalog verglichen. Tritt bei diesem Vergleich ein Treffer auf, terminiert der Algorithmus und speichert die betreffende Bewertung als potenzielles Privacy Issue ab.

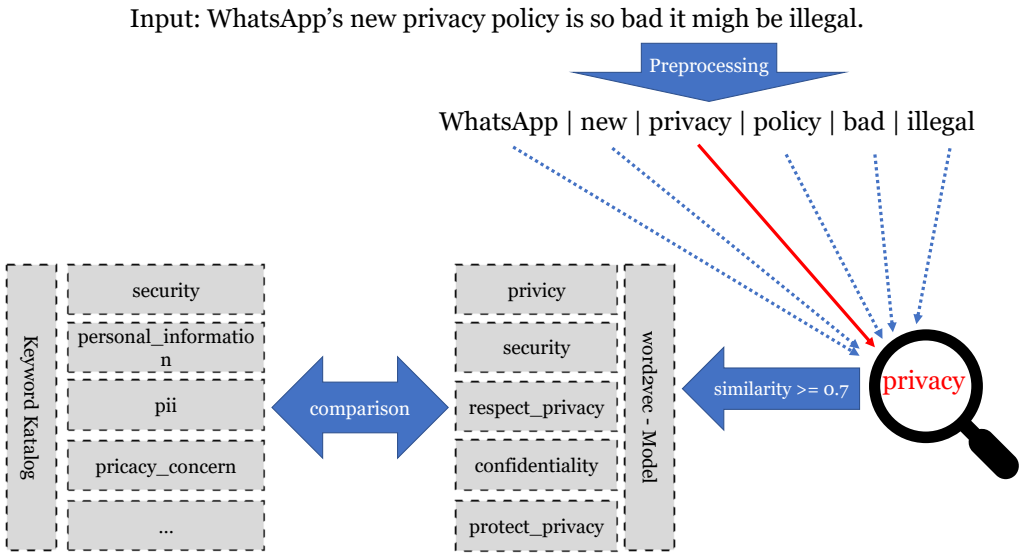


Abbildung 4.5: Modell der intelligenten Keyword-Suche.

Auf diese Weise wurden Bewertungen von 118 Apps analysiert und

es wurden 1.743 Bewertungen gefunden, die potentielle Privacy Issues darstellen. Diese 1.743 Bewertungen bestanden aus 5.967 Sätzen. Um ein Gleichgewicht zwischen Positiven und Negativen zu halten, wurden zusätzlich die gleiche Menge negative Funde hinzugefügt. Jeder potentiell positive Fund, wurde demnach mit einem potentiell negativen Fund ergänzt. Hintergründe zu diesem Vorgehen werden später ergänzt.

4.3.6 Labeling

Die vorselektierten Bewertungen aus der vorhergehenden Phase, wurden in dieser Phase manuell gelabelt. Dazu wurde ein Modul implementiert, welches die vorselektierten Bewertungen aus einer JSON Datei einliest, anschließend satzweise auf der Konsole ausgibt und abschließend wieder im JSON Format abspeichert. Dabei wurde sichergestellt, dass die angezeigten zu labelnden Sätze in der richtigen Reihenfolge ausgegeben werden, um den Kontext der Bewertung nicht zu verfälschen. Außerdem war es möglich, alle Sätze einer Bewertung anzeigen zu lassen, um den Kontext besser verstehen zu können. Der Rater musste zum labeln lediglich die Werte 0 oder 1 vergeben und diese bestätigen. Bei diesem Vorgang wurde nur das Datenfeld „isPrivacyIssue“ der Datenklasse „Review“ gemäß Listing 4.1 auf 1 (ist Privacy Issue) oder 0 (ist kein Privacy Issue) gesetzt. Das übergeordnete Datenfeld „privacy_issue“ wurde nach dem Abschluss eines Label-Durchgangs, automatisch durch das Modul vergeben. Dies wurde gemäß des Datenfeldes „isPrivacyIssue“ der einzelnen Sätze, als logische ODER Verknüpfung entschieden. Dieses Vorgehen hat den Vorteil, dass in den folgenden Phasen (Klassifizierung und Deep Learning) beim Training der Modelle anhand der gelabelten einzelnen Sätze oder ganzer Reviews unterschieden werden kann, welches bereits im Konzeptkapitel 3.1.1 angesprochen wurde und hier Anwendung findet.

Die aus der Phase „Vorselektion“ gefilterten 1.743 Bewertungen wurden in dieser Phase gelabelt. In 1.488 positiven vorselektierten Bewertungen, befanden sich 756 TP und 732 FP. In 255 negativen vorselektierten Bewertungen, befanden sich 249 TN und 6 FN. Dies ist in einer Konfusionsmatrix in Tabelle 4.4 abgebildet. Bewertet man nun die Performance dieser Vorselektion, gelangt man zu folgender Performance: *Accuracy: 0.5766, Precision: 0.5081, Recall: 0.9921 und F1-Score: 0.6720.* Dies zeigt sehr gut, dass der gewählte Ansatz der Vorselektion, gut funktioniert hat, da zum einen in den Negativen sehr wenig Positive (6) enthalten waren und zum anderen, in den Positiven viele (756) Privacy Issues gefunden werden konnten. Insgesamt waren unter 1.743 Bewertungen 762 Privacy Issues zu identifizieren, was einem Anteil von ca. 43% entspricht. Unter der Annahme, dass es sich ohne Vorselektion, bei maximal 2,4% der Bewertungen [25] um Privacy Issues handelt, müssten

ohne diesen Ansatz der Vorselektion mehrere tausend Bewertungen manuell gelabelt werden, um die gleiche Menge von 762 Privacy Issues zu erhalten. Somit ist dieser Ansatz der Vorselektion sehr hilfreich für die effiziente Suche nach Privacy Issues in großen Datenmengen von App Store Bewertungen.

		Actual class	
		P	**N**
Predicted	**P**	756	732
	N	6	249

Tabelle 4.4: Konfusionsmatrix zu Bewertungen aus der Vorselektion.

Wie bereits erwähnt, wurden nicht nur die Bewertungstexte als Ganzes gelabelt, es wurden auch die einzelnen Sätze der 1.743 Bewertungen gelabelt welche aus 5.967 Sätzen bestanden. Eine Übersicht der Labeling-Ergebnisse wird in Tabelle 4.5 gezeigt. Für die Performance-Analyse gilt: *Accuracy: 0.3550, Precision: 0.2503, Recall: 0.9930 und F1-Score: 0.3998*. Auch wenn die Werte Accuracy, Precision und F1-Score eher schlecht ausfallen, zeigt sich dennoch, dass der gewählte Ansatz gut funktioniert, da von den 5.967 zu labelnden Sätzen, 1.291 als Privacy Issue identifiziert werden konnten. Dies entspricht einer Trefferquote von ca. 22% welche ebenfalls höher ist als die angenommenen 2,4% ohne Vorselektion.

		Actual class	
		P	**N**
Predicted	**P**	1282	3840
	N	9	836

Tabelle 4.5: Konfusionsmatrix zu Sätzen aus der Vorselektion.

Mit diesen gelabelten Daten, konnten anschließend in der folgenden Phase, mehrere ML-Modelle trainiert werden. Somit wurden mehrere ML-Modell trainiert, welche in der Lage waren, Bewertungen auf Privacy Issues zu untersuchen.

4.3.7 Training Klassifikator / Deep Learning

In dieser Phase wurden verschiedene ML-Modelle mit den zuvor gelabelten Daten trainiert und miteinander verglichen. Grundsätzlich wurde hier in zwei Kategorien unterschieden, den Klassifizierungsalgorithmen und den Deep Learning Verfahren. Als Klassifizierungsalgorithmen wurden Logistische Regression, Support Vector Classification (SVC), Decision Tree und Random

Forest eingesetzt. Diese Algorithmen wurden jeweils in einem Ensemble und in einem Stacking-Classifier zusammengefasst und ausgewertet. Im Bereich Deep Learning wurden Feedforward-Netze, CNNs und RNNs genutzt. Abschließend wird ein Vergleich von Ensemble, Stacking-Classifier, Feedforward-Netz, CNN und RNN gegeben, sowie Vor- und Nachteile dargelegt. Weitere Vergleiche zwischen den Modellen bezüglich der beiden eingesetzten Trainingsmethoden (Training auf Sätzen oder ganzen Bewertungen) werden gezeigt.

Trainings-, Validierungs- und Testmengen

Tabelle 4.6 zeigt die verfügbaren und gelabelten Daten, die für das Training der Modelle zur Verfügung standen. Da NLP Verfahren am besten funktionieren, wenn die Anzahl der Positiv- bzw. Negativbeispiele gleich sind, wurde dies Anzahl der Bewertungen angeglichen. So wurden zum Training lediglich 762 positive und 762 negative Bewertungen, respektive 1.291 positive und 1.291 negative Sätze verwendet. Weiterführende Informationen zum Problem der schlecht ausbalancierten Datensets, analysierten bspw. Kotsiantis et al. [21] in ihrer Arbeit „Handling imbalanced datasets: A review“ oder Visa et al. [44] unter dem Titel „Issues in Mining Imbalanced Data Sets - A Review Paper“.

	Positive	Negative
Bewertungen	762	981
Sätze	1.291	4.676

Tabelle 4.6: Übersicht über die Anzahl gelabelter Daten. Positive: stellen ein Privacy Issue dar, Negative: stellen kein Privacy Issue dar.

Die Aufteilung der Daten in Trainings-, Validierungs- und Testmengen ist in Tabelle 4.7 dargestellt. Die Größe des Validierungsdatensatzes bei den eingesetzten Klassifikationsalgorithmen, konnte programmiertechnisch nicht explizit angegeben werden, da dafür keine Konfigurationsmöglichkeiten vorgesehen waren. Dies konnte lediglich bei den Deep Learning Verfahren explizit konfiguriert werden. Weiterhin wurden alle Verfahren mit 10-fach Cross Validation überprüft. Dazu wurde die Variante „Stratified k-folds“ verwendet, welches für ausbalancierte Folds sorgt. Alle folgenden Performance-Analysen beziehen sich daher auf die 10-fach Cross Validation der Verfahren. Weiterführende Literatur über die Wahl von k ist im Buch *Applied Predictive Modeling* [22, S. 70] oder in den Forschungsarbeiten von Molinaro et al. [28] und Kim [20] vorgestellt.

	Trainingssatz	**Testsatz**	**Validierungssatz**
Klassifikations-algorithmen	80%	20%	N/A
Deep Learning Verfahren	80%	20%	20% vom Trainingssatz

Tabelle 4.7: Trainings-, Validierungs- und Testmengen nach verwendeten Verfahren.

Allgemeiner Aufbau

In diesem Unterkapitel wird der allgemeine Aufbau der folgenden ML-Algorithmen erläutert. Dieser Aufbau ist grundsätzlich bei allen Verfahren gleich. Als Eingabe dienten die gelabelten Daten aus der vorhergehenden Phase (Labeling). Alle Daten wurden durch die, in Unterkapitel 4.3.2 beschriebene Preprocessing-Pipeline geführt. Um Performance-Vergleiche durchzuführen wurden einzelne Funktionen der Pipeline vorübergehend deaktiviert. Alle möglichen Testvarianten sind in Tabelle 4.8 aufgelistet. Bei den Klassifikationsalgorithmen besteht die Besonderheit, dass keine Word Embeddings eingesetzt werden konnte. Ein weiterer Punkt der angepasst werden konnte, betrifft die Trainingsdaten. Dort war es möglich die Modelle auf Grundlage der gesamten Bewertung mit dazugehörigem Label zu trainieren, oder ein Training auf Grundlage einzelner Sätze durchzuführen. Mit diesem Vorgehen konnten unterschiedliche Resultate bei der Verwendung der beiden Modelle erzielt werden. Ergebnisse zu diesem Vergleich werden in Kapitel 5.1.2 aufgeführt. Als Vectorizer wurde *TfidfVectorizer*[5] von sklearn verwendet, welcher mit einer N-Gramm Range von 1 bis 2 konfiguriert wurde.

Ensemble

Bei der Implementierung des Ensemble wurden vier Klassifikationsalgorithmen verwendet. Logistische Regression, Decision Trees, Random Forest und SVCs welche ausführlich in Kapitel 2.4.1 beschrieben wurden. Die Implementierung des Ensembles erfolgte dabei mit Hilfe der Programmbibliothek *sklearn.ensemble.VotingClassifier*[6]. Dieser VotingClassifier bietet die Möglichkeit, das Voting zu beeinflussen indem der Parameter „voting“ auf „soft“ oder „hard“ gesetzt wird. In dieser Implementierung wurde der

[5] TfidfVectorizer: `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html` Zugriff: 29.04.2021

[6] VotingClassifier: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html` Zugriff: 29.04.2021

Verfahren	Rechtschreibkorrektur	Cleaning	Word Embedding	Training auf Bewertungen (B) / Sätze (S)
Ensemble	On / Off	On / Off	N/A	B / S
Stacking-Clf	On / Off	On / Off	N/A	B / S
Feedforward	On / Off	On / Off	Google, word2vec_ground_truth	B / S
CNN	On / Off	On / Off	Google, word2vec_ground_truth	B / S
RNN	On / Off	On / Off	Google, word2vec_ground_truth	B / S

Tabelle 4.8: Überblick über die 88 analysierten Varianten aller verwendeten Verfahren.

Parameter auf *soft* konfiguriert was bedeutet, dass die Wahrscheinlichkeiten der Klassifizierung der verwendeten Algorithmen als Entscheidungsgrundlage genutzt werden. Dieser Parameter wurde bereits in Kapitel 2.4.2 beschrieben.

Die Implementierung der einzelnen Klassifikationsalgorithmen erfolgte mit Hilfe der Programmbibliotheken *sklearn.linear_ model.LogisticRegression*[7], *sklearn.tree.DecisionTreeClassifier*[8], *sklearn.ensemble. RandomForestClassifier*[9] *und sklearn.svm.SVC*[10]. Zur Verarbeitung der Daten wurde eine Pipeline[11] verwendet, welche die Eingabedaten in die Vorverarbeitungspipeline führt, sie anschließend mit dem TF-IDF Verfahren vektorisiert und abschließend das Klassifikationsmodell trainiert, welches die Klassifikation im VotingClassifier bestimmt. Die erzielten Ergebnisse sind in Kapitel 5.1.2 aufgeführt.

Stacking-Classifier

Der Stacking-Classifier wurde durch die Programmbibliothek *sklearn.ensemble.StackingClassifier*[12] von sklearn implementiert. Wie

[7]LogisitcRegression: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html` Zugriff: 29.04.2021

[8]DecisionTreeClassifier: `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html` Zugriff: 29.04.2021

[9]RandomForestClassifier: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html` Zugriff: 29.04.2021

[10]SVC: `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html` Zugriff: 29.04.2021

[11]Pipeline: `https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html` Zugriff: 29.04.2021

[12]StackingClassifier: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html` Zugriff: 29.04.2021

auch beim Ensemble, wurden die gleichen Klassifikationsalgorithmen verwendet: Logistische Regression, Decision Trees, Random Forest und SVCs. Die Implementierung dieser Algorithmen erfolgte analog zu der des Ensemble. Als Meta-Classifier wurde Logistische Regression eingesetzt. Die Ergebnisse des Stacking-Classifiers sind in Kapitel 5.1.2 vermerkt.

Aufbau der folgenden Deep Learning Verfahren

Die folgenden drei Deep Learning Verfahren unterscheiden sich lediglich in der Modellarchitektur. Daher wird in diesem Abschnitt zunächst der Teil beschrieben, der bei allen Verfahren gleich ist.

Word Embeddings: Um die Leistung der Klassifikation der Deep Learning Algorithmen zu verbessern, wurden vorinitialisierte Wortvektoren verwendet. Dies ist eine erfolgreiche Methode um die Leistung der eingesetzten Netze zu verbessern, wenn die Menge an Trainingsdaten gering ist (Collobert et al. [8]). Zu diesem Zweck wurden die frei zu verwendenden word2vec-Vektoren von Google[13] verwendet, welche auf 100 Milliarden Wörtern aus Google News trainiert wurden. Diese Vektoren haben eine Dimensionalität von 300 und wurden mit der Bag-of-Words Architektur trainiert. Wörter die nicht in der Menge der vortrainierten Wörter vorhanden waren, wurden mit zufälligen Werten initialisiert. Da beim erstellten Modell aus Phase 3 bereits ein eigenes word2vec Word Embedding trainiert wurde, konnte dieses eingesetzt und mit der Leistung des Google News Word Embeddings verglichen werden. Diese Word Embeddings verfügten ebenfalls über eine Dimensionalität von 300 und werden folgend mit „word2vec - Ground Truth“ bezeichnet. Außerdem gab es eine Variante (bezeichnet als *ohne Word Embedding*), bei der keine der beiden vortrainierten Word Embeddings genutzt wurden, stattdessen wurde ein Embedding Layer verwendet.

Overfitting: Um Overfitting zu vermeiden, wurde das Verfahren „Early Stopping“ eingesetzt, welches verschiedene Parameter wie „Validation Loss“ oder „Accuracy“ des Trainingsprozesses überwacht. Die Implementierung erfolgte mit Hilfe der Programmbibliothek tf.keras.callbacks.EarlyStopping[14]. Das Verfahren überwachte den Parameter „val_loss“ welcher minimiert werden sollte. Abschließend wurden die besten Parameter des Trainingsprozesses wiederhergestellt.

[13] word2vec: `https://code.google.com/archive/p/word2vec/` Zugriff: 02.05.2021

[14] EarlyStopping: `https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping` Zugriff: 02.05.2021

Feedforward Neural Network - FNN

Die Implementierung des Netzes erfolgte mit der Programmbibliothek „keras“[15], welche alle benötigten Komponenten für dir folgenden Netze zur Verfügung stellte. Als einfaches FNN wurde eine Netzarchitektur, die sich auf die Grundlagen von Collobert et al. [8] bezieht, verwendet. Diese ist in Tabelle 4.9 dargestellt. Die Größe des Eingabevektors (400) wurde anhand der längsten verfügbaren Bewertung in den Trainingsdaten ermittelt.

Layer	**Input / Output Dim**	**#Parameter**	**Berechnung**	**Activation**
Word Embedding[16]	400 / 300	991.200	#Wörter[17]$*300+300$	-
GlobalMaxPool1D	300 / 300	0	-	-
Dense	300 / 128	38.528	$300*128+128$	ReLU
Dense	128 / 1	129	$1*128+1$	Sigmoid

Tabelle 4.9: Layer Architektur des FNN mit Angabe der Eingabe- und Ausgabedimensionen, Anzahl der Parameter, der dazugehörigen Berechnungen und der genutzten Aktivierungsfunktionen. Anzahl trainierbarer Parameter: 38.657. Anzahl nicht trainierbarer Parameter: 991.200 (Input Layer).

Um Vergleiche durchführen zu können, wurde der Word Embedding Layer teils mit Google News Word Vektoren, teils mit selbst trainierten Word Embeddings aus Phase 3 des Prozesses initialisiert. Außerdem wurde ein Vergleichstest ohne Word Embeddings durchgeführt. Die Ergebnisse sind in Kapitel 5.1.2 dargestellt. Dieses Vorgehen ist auch bei den folgenden Netzen (CNN und RNN) angewandt worden.

Convolutional Neural Network - CNN

Die Netzarchitektur des CNNs ist in Tabelle 4.10 dargestellt. Der Convolutional Layer hat eine Filtergröße von 5 mit einem Stride von 1. Die Eingabegröße ist 300 mit einer Unit-Größe von 128. Als Aktivierungsfunktion wurde ReLU genutzt. Um Overfitting zu vermeiden, wurde ein Dropout Layer mit einer Dropoutrate von 0.5 hinzugefügt. Dies führt dazu, dass zufällig 50% der Neuronen in diesem Layer deaktiviert werden. Die Ergebnisse werden in Kapitel 5.1.2 erläutert.

[15] Keras Dokumentation: `https://keras.io/api/` Zugriff: 03.05.2021

[16] Varianten: Google News Vektor, word2vec - Ground Truth, ohne Word Embedding (Embedding Layer)

[17] Abhängig von der Anzahl der Wörter in den Eingabebewertungen. Hier: #Wörter= 3303

Layer	**Input / Output Dim**	**#Parameter**	**Berechnung**	**Activation**
Word Embedding	400 / 300	1.030.200	#Wörter(3433) * 300 + 300	-
Conv 1D	300 / 396	192.128	5*128*300+128	ReLU
GlobalMaxPool1D	396 / 128	0	-	-
Flatten	128 / 128	0	-	-
Dropout 0.5	128 / 128	0	-	-
Dense	128 / 1	129	1 * 128 + 1	Sigmoid

Tabelle 4.10: Layer Architektur des CNN mit Angabe der Eingabe- und Ausgabedimensionen, Anzahl der Parameter, der dazugehörigen Berechnungen und der genutzten Aktivierungsfunktionen. Anzahl trainierbarer Parameter: 192.257. Anzahl nicht trainierbarer Parameter: 1.030.200 (Input Layer).

Recurrent Neural Network - RNN

Für die Implementierung des RNNs wurde ein LSTM Netz eingesetzt. Die Architektur ist in Tabelle 4.11 dargestellt. Als Eingabevektoren wurden, wie schon zuvor, Word Embedding Vektoren mit einer Dimensionalität von 300 verwendet. Darauf folgen zwei bidirektionale LSTM Layer welche Sigmoid als Aktivierungsfunktion nutzen. Anschließend wird die Ausgabe an ein Dense Layer mit 32 Neuronen und ReLU Aktivierung weitergeleitet. Um Overfitting zu vermeiden, wurde auch hier ein Dropout Layer mit Dropoutrate von 0.5 eingefügt. Abschließend folgte ein weiterer Dense Layer mit Sigmoid Aktivierungsfunktion. Die erzielten Ergebnisse des Netzes, sind in Kapitel 5.1.2 erläutert.

Layer	**Input / Output Dim**	**#Parameter**	**Berechnung**	**Activation**
Word Embedding	400 / 300	1.030.200	#Wörter (3433) * 300 + 300	-
Bidirectional LSTM	300 / 64	186.880	$2*4*((300+64)*64*64)^{18}$	Sigmoid
Bidirectional LSTM	128 / 32	41.216	2*4*((128+32) *32+32))	Sigmoid
Dense	64 / 32	2080	64 * 32 + 32	ReLu
Dropout 0.5	32 / 32	0	-	-
Dense	32 / 1	33	1 * 32 + 1	Sigmoid

Tabelle 4.11: Layer Architektur des RNN mit Angabe der Eingabe- und Ausgabedimensionen, Anzahl der Parameter, der dazugehörigen Berechnungen und der genutzten Aktivierungsfunktionen. Anzahl trainierbarer Parameter: 230.209. Anzahl nicht trainierbarer Parameter: 1.030.200 (Input Layer).

4.3.8 Weitere Bewertungen analysieren

Die letzte Phase des Prozesses beschreibt das Vorgehen, wie mit Hilfe der zuvor erstellten Modelle, neue App Bewertungen auf Privacy Issues untersucht werden können. Alle trainierten ML-Verfahren erstellen ein Modell, mit dem im Anschluss unbekannte Daten klassifiziert werden können. Diese Modelle werden in dieser Phase genutzt um weitere Bewertungen zu analysieren. Diese neu identifizierten Bewertungen können durch die Labeling-Phase (siehe Kapitel 3.5.1) erneut in den Kreislauf eingebracht werden um den Trainingsdatenbestand zu erweitern. Das erstellte Modell kann somit weiterentwickelt und verbessert werden.

4.4 Knowledge Base

Um alle identifizierte Datenschutz- und Privatsphäreverletzungen in einem Archiv vorrätig zu haben, wurde eine Knowledge Base in Form einer Datenbank angelegt. Die Datenbank wurde als Docker Container mit dem Docker Hub Image *mariadb*[19] in der Version 10.5 implementiert. Die Schnittstelle ist im Modul „knowledge_base\db_connector.py“ entwickelt worden. Eine grafische Verwaltungsoberfläche wurde mittels *Adminer*[20] in der Version 4.8.0 bereitgestellt. Die detaillierte Konfiguration der Docker Container, ist in Anhang C.1 hinterlegt. Ein definiertes, auf dem lokalen Datenträger abgelegtes Speichervolume diente dazu, die Daten persistent zu speichern.

Schließlich können die identifizierten Privacy Issues einfacher untersucht werden, wenn Kategorien identifiziert und ebenfalls in einer Datenbanktabelle hinterlegt werden. Aus allen analysierten Privacy Issues konnten zehn Kategorien abgeleitet werden, welche in Tabelle 4.12 aufgelistet sind. Kategorisiert wurden alle einzelnen Sätze der identifizierten Privacy Issues. Dabei sind einzelnen Sätzen auch mehrere Kategorien zugewiesen worden. Eine Übersicht der Verteilung dieser Kategorien, ist detailliert in Kapitel 5.1.3 erläutert.

Das gewählte Datenbankschema ist in Abbildung 4.6 abgebildet. In der Tabelle *kb* wurden alle Bewertungen die Privacy Issues enthalten abgelegt. Um Bewertungen effizient finden zu können, wurde ein Attribut „content_hash_id“ erstellt. Dieses enthält einen Secure Hash Algorithm (SHA)-256 Hash, bestehend aus der *app_id* und dem *content* einer Bewertung. In der Tabelle *category_names* sind alle identifizierten Kategorien gem. Tabelle

[18] Berechnung: 2*(da Bidirectional) 4*(da 4 Dense Layer in einem LSTM enthalten sind)
[19] Docker Hub - mariadb: `https://hub.docker.com/_/mariadb` Zugriff: 29.04.2021
[20] Docker Hub - Adminer: `https://hub.docker.com/_/adminer` Zugriff: 29.04.2021

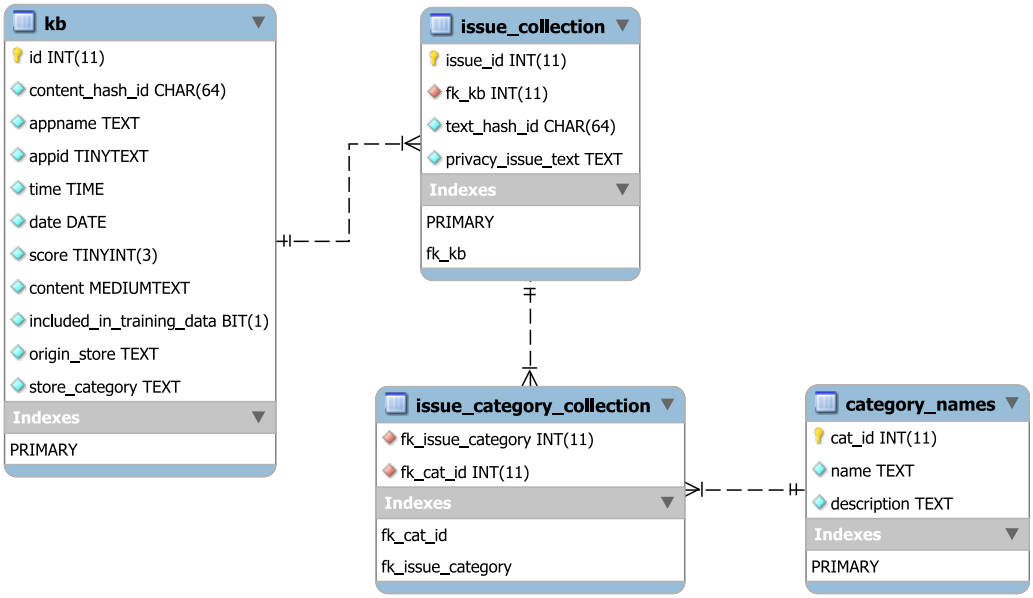


Abbildung 4.6: Datenbankschema der Knowledge Base.

4.12 abgespeichert. Die Tabelle *issue_ collection* verknüpft die Bewertungen aus der Tabelle *kb* mit der Tabelle *issue_ category_ collection* in der wiederum eine Verknüpfung zur Tabelle *category_ names* angelegt ist.

4.5 Metriken - Ground Truth

Die Daten die zum Training der zuvor vorgestellten Algorithmen verwendet wurden, wiesen die Metriken auf, die in Tabelle 4.13 vermerkt sind. Unterschieden wurde hierbei nach ganzen Bewertungen, bestehend aus mehreren Sätzen und nach allen einzelnen Sätzen aus den Bewertungen. Die Sätze sind somit eine Teilmenge der Bewertungen, weisen aber dennoch Unterschiede auf.

Abbildung 4.7 zeigt die Verteilung nach durchschnittlicher Wortlänge der Bewertungen, aufgeteilt nach vergebenem Score. Daraus lässt sich ableiten, dass Bewertungen mit einem Score von 3, die meisten Wörter (59,3 Wörter pro Bewertung) zum Verfassen einer Bewertung benötigen. Sehr gute Bewertungen mit einem Score von 5, sind hingegen eher kürzer (37,2 Wörter). Insgesamt bestehen negative Bewertung (Score 1 - 3) aus mehr Wörtern als positive Bewertungen. Abbildung 4.8 zeigt die Verteilung der Bewertungen nach Score. Auffällig ist hier die große Masse an negativen (Score 1) und positiven (Score 5) Bewertungen. Dies ist darauf zurückzuführen, da hier nicht nur Positivbeispiele für Privacy Issues enthalten sind, sondern auch

cat_id	Name	Beschreibung
1	other	Everything that doesn't fit into any other category.
2	security_breach	A privacy issue resulted from a security breach.
3	permissions	If too many or unneeded permissions are required by the app.
4	data_transfer_to_third	Some data will be passed on to third parties.
5	bad_privacy_policy_ or_user_agreement	User agreement or privacy policy are: bad / questionable / incomprehensible / not available.
6	violation_data_economy	Violation of data economy or data avoidance principle (not technical level).
7	request_privacy	Only ask for "more"privacy.
8	claim_privacy_issue	Pure conjecture / assertion / statement by a user: "violation of privacyör recommendation to use another app.
9	spy_and_steal_data	Collection of user data (geolocation, taking photos ...) (steal data, spy data).
10	advertising	Loss of privacy due to advertising.

Tabelle 4.12: Übersicht der Knowledge Base Kategorien mit vergebener ID und Beschreibung.

Type	Train Size	Test Size	#Klassen	Avg. Length	#unique words
Bewertungen	1.394	349	2	46,5 words	6.159
Sätze	4.774	1.193	2	13,5 words	6.159

Tabelle 4.13: Metriken zur Ground Truth

Negativbeispiele. Die Abbildung 4.9 zeigt diese Metriken aufgeteilt auf Positiv- und Negativbeispiele. Hier wird deutlich, dass mehr Privacy Issues in Bewertungen mit negativem Score (1-2) enthalten sind als in positiven Bewertungen (4-5). Ein Großteil (521 Bewertungen) der Privacy Issues lässt sich daher in Bewertungen mit einem Score von 1 finden. Allerdings lassen sich auch Privacy Issues in Bewertungen (241 Bewertungen) mit einem höheren Score (2-5) identifizieren. Dies bestätigt auch die Herangehensweise, die Bewertungen mit positivem Score nicht außer Acht zu lassen und diese dennoch in den Analyseprozess mit einzubeziehen.

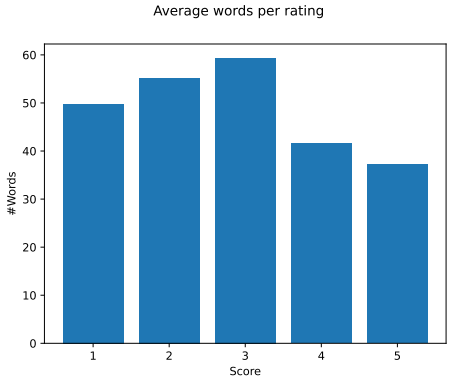


Abbildung 4.7: Übersicht über die durchschnittliche Wortlänge aller Bewertungen der Ground Truth.

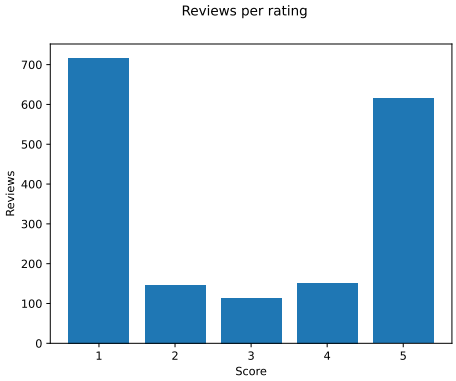
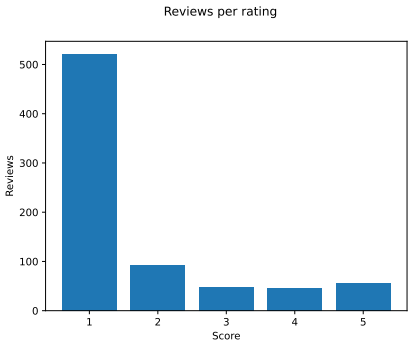
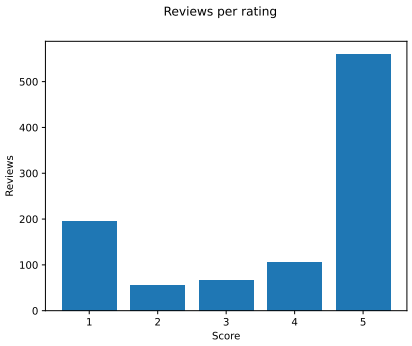


Abbildung 4.8: Übersicht über die Verteilung der Bewertungen nach Score aller Bewertungen der Ground Truth.



(a)



(b)

Abbildung 4.9: Übersicht nach Score, über die Anzahl der Bewertungen in denen a) Privacy Issues enthalten sind und b) keine Privacy Issues enthalten sind.

Word Cloud

In den Abbildungen 4.10 und 4.11 ist eine Word Cloud[21] abgebildet, welche häufig vorkommenden Wörter in den Bewertungen visuell darstellt. Häufig auftretende Wörter werden dabei größer dargestellt und weniger häufige Wörter werden klein dargestellt. Anhand der beiden Abbildungen, lässt sich gut die Schwierigkeit für die Klassifikationsalgorithmen erkennen, da bei beiden Klassen von Bewertungen sehr ähnliche Begrifflichkeiten enthalten sind. Dies kann die Klassifikation deutlich erschweren. Der Grund für diese Ähnlichkeit begründet sich darin, dass in der Ground Truth auch viele Bewertungen vorhanden sind, die bei der Keyword-Vorselektion gefunden wurden, dann aber beim manuellen Labeling als „Negativ“ identifiziert wurden. Dies führt dazu, dass sich Keywords auch in der Masse der Negativbeispiele wiederfinden.

Abbildung 4.10: In dieser Word Cloud werden die häufigsten Begriffe aus Bewertungen dargestellt, die **ein** Privacy Issue beinhalten.

[21] Ausschluss der Wörter: app, use, want, will, even

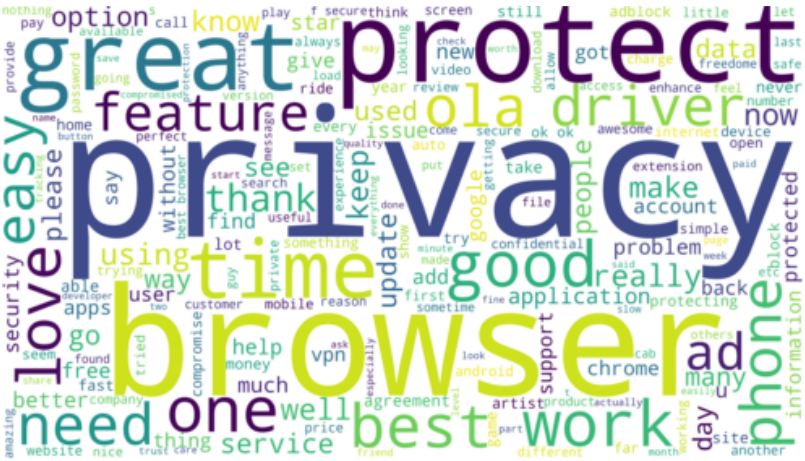


Abbildung 4.11: In dieser Word Cloud werden die häufigsten Begriffe aus Bewertungen dargestellt, die **kein** Privacy Issue beinhalten.

Kapitel 5

Evaluation

5.1 Ergebnisse

In diesem Kapitel werden alle Ergebnisse präsentiert, die der in dieser Arbeit beschriebene Ansatz hervorgebracht hat. Insbesondere ein Vergleich der eingesetzten Klassifizierungsverfahren und den damit verbundenen Vor- und Nachteilen wird beschrieben. Ebenfalls wird die Frage beantwortet, ob das Training der Klassifikationsalgorithmen mittels ganzer Bewertungen oder einzelnen Sätzen einen Vorteil bringt. Abschließend wird die beste Konfiguration bestehend aus bester Preprocessing Pipeline, eingesetztem Klassifikator und Trainingsstrategie vorgestellt.

5.1.1 Realdaten

Um die Performance-Werte, die in den folgenden Kapiteln angegeben werden, besser evaluieren zu können, und die ML-Modelle auch auf ihre Praxistauglichkeit analysieren zu können, wurden weitere Daten in Form von App Store Bewertungen erhoben. Die Daten stammen von Apps, die nicht in der Ground Truth, die zum Training der Klassifikationsverfahren genutzt wurden, vorhanden waren. Damit wurde dem entwickelten Ansatz, ein zusätzlicher Evaluationsschritt hinzugefügt um die Funktionsfähigkeit nochmals zu bestätigen. Die Tabelle 5.1 beschreibt Metriken der Realdaten.

Type	**Test Size**	**#Klassen**	**#unique words**
Bewertungen	70	2	606
Sätze	122	2	606

Tabelle 5.1: Metriken der Realdaten. Angaben enthalten Positiv- und Negativbeispiele zu gleichen Teilen. #: Anzahl

5.1.2 Klassifikationsverfahren

In diesem Unterkapitel werden die Performance-Werte der eingesetzten Klassifikationsverfahren vorgestellt. Dabei werden die Werte *Accuracy* und *F1-Score* (siehe Kapitel 2.4.4) zum jeweiligen Modell angegeben. Zusätzlich werden diese Werte für die beiden Realdatensets *Sentences* und *Reviews* angegeben. Detaillierte Daten wie Recall oder Precision sind im Anhang D.4 in den Tabellen D.2 und D.3 aufgeführt. Es folgt eine Erklärung der Werte in den Tabellenspalten *Model, Sentences und Reviews*:

- Model: Werte wurden ermittelt, anhand der Testdaten, welche aus den Daten stammen, mit denen das Modell trainiert wurden.
- Reviews: Werte wurden ermittelt, anhand von Realdaten, welche nicht in den Trainingsdaten vorhanden waren. Details zu den Realdaten werden in Unterkapitel 5.1.1 gegeben. Die Validierung erfolgte durch Eingabedaten die aus ganzen Bewertungen bestanden.
- Sentences: Werte wurden ermittelt, anhand von Realdaten. Als Eingabe dienten einzelne Sätze von Bewertungen.

Die Tabelle 5.2 zeigt die jeweils besten Ergebnisse für jedes Klassifikationsverfahren an, basierend auf dem Training mit einzelnen Sätzen. Dabei wurden die Deep Learning Algorithmen, nach verwendetem Word Embedding unterteilt. Wenn kein Word Embedding verwendet wurde (ohne Word Embedding) wurde stattdessen ein Embedding Layer eingesetzt. Bei den Verfahren Ensemble und Stacking-Clf war es nicht möglich Word Embeddings zu verwenden. Außerdem sind Performance-Werte bezogen auf das trainierte Modell, auf Bewertungen (Reviews) und Sätze (Sentences) angegeben. Die Werte des Modells beziehen sich auf dieselben Daten womit auch das Modell trainiert wurde. Die Werte für Reviews beziehen sich darauf, dass zum validieren des Modells ganze Bewertungen genutzt wurden. Dementsprechend wurden die Werte für Sentences durch die Validierung, einzelner Sätze berechnet. Tabelle 5.3 zeigt selbiges für das Training basierend auf ganzen Bewertungen.

Die Abbildungen 5.2, 5.3 und 5.4 zeigen die jeweils besten F1-Scores jedes Klassifikators. Dabei wurde auf die Unterteilung nach *Cleaning, Spell Correction* und *Word Embedding* aus Gründen der Übersichtlichkeit verzichtet. Es wird daher nur die beste Variante der Klassifikationsalgorithmen abgebildet. Die linke Hälfte der Diagramme zeigt die F1-Scores für das Trainings basierend auf ganzen Bewertungen, wohingegen die rechte Hälfte die Werte für das Training basierend auf einzelnen Sätzen darstellt. Die

				Model		Reviews		Sentences	
Classifier	cl	sp	embeding_vector	Acc	F1	Acc	F1	Acc	F1
Ensemble	T	T	-	0.875	0.873	0.943	0.944	0.91	0.909
Stacking Clf	T	T	-	0.872	0.875	0.914	0.919	0.885	0.889
FNN	T	T	ohne	0.877	0.872	0.943	0.946	0.918	0.918
	T	T	word2vec - Ground Truth	0.759	0.75	0.914	0.914	0.803	0.81
	T	T	GoogleNews-vectors-negative300	0.854	0.844	0.914	0.921	0.885	0.887
CNN	T	T	ohne	0.882	**0.878**	0.957	**0.958**	0.926	**0.924**
	T	F	word2vec - Ground Truth	0.84	0.839	0.914	0.919	0.902	0.905
	T	T	GoogleNews-vectors-negative300	0.887	0.876	0.914	0.917	0.869	0.873
RNN	T	T	ohne	0.869	0.864	0.914	0.919	0.893	0.896
	T	T	word2vec - Ground Truth	0.837	0.837	0.943	0.946	0.902	0.906
	T	F	GoogleNews-vectors-negative300	0.873	0.874	0.886	0.889	0.918	0.919
			Maximalwerte	**0.887**	**0.878**	**0.957**	**0.958**	**0.926**	**0.924**

Tabelle 5.2: Übersicht der besten Werte der Klassifizierungsverfahren, basieren auf dem **Training auf einzelnen Sätzen**. In Fett sind die globalen Höchstwerte dargestellt. Die letzte Zeile (MAX) zeigt die Maximalwerte der Spalten an. Abkürzungen: Classifier (Clf), Cleaning (cl), Accuracy (Acc), Spell correction (sp), Accuracy (Acc), True (T), False (F). Diese Abkürzungen gelten für alle folgenden Tabellen.

				Model		Reviews		Sentences	
Classifier	cl	sp	embeding_vector	Acc	F1	Acc	F1	Acc	F1
Ensemble	T	F	-	0.852	0.852	0.914	0.906	0.877	0.865
Stacking-Clf	T	F	-	0.868	**0.872**	0.886	0.895	0.82	0.828
FNN	F	T	ohne	0.861	0.862	0.9	0.907	0.86	0.86
	T	T	word2vec - Ground Truth	0.66	0.653	0.8	0.816	0.762	0.76
	T	T	GoogleNews-vectors-negative300	0.778	0.782	0.914	0.921	0.828	0.832
CNN	F	F	ohne	0.861	0.862	0.871	0.87	0.869	0.86
	T	F	word2vec - Ground Truth	0.741	0.755	0.814	0.835	0.713	0.748
	T	T	GoogleNews-vectors-negative300	0.851	0.852	0.9	0.909	0.787	0.783
RNN	T	F	ohne	0.861	0.864	0.866	0.889	0.852	0.842
	T	T	word2vec - Ground Truth	0.783	0.781	0.886	0.897	0.885	**0.889**
	T	T	GoogleNews-vectors-negative300	0.815	0.815	0.929	**0.93**	0.811	0.789
			Maximalwerte	**0.868**	**0.872**	**0.929**	**0.93**	**0.885**	**0.889**

Tabelle 5.3: Übersicht der besten Werte der Klassifizierungsverfahren, basieren auf dem **Training mit ganzen Bewertungen**. In Fett sind die globalen Höchstwerte dargestellt. Die letzte Zeile (MAX) zeigt die Maximalwerte der Spalten an. Abkürzungen: siehe Tabelle 5.2

verschiedenen Farben stellen die Klassifikationsalgorithmen dar. Es werden drei verschiedene Abbildungen gezeigt, die jeweils die Performance der Validierungssets welches auch zum Trainieren des Klassifikators genutzt wurde (Abb. 5.2), die Performance, wenn auf das trainierte Netz ganze Bewertungen aus den Realdaten angewendet werden (Abb. 5.3) und wenn einzelne Sätze zum validieren genutzt wurden (Abb. 5.4). Die Abbildung 5.5 zeigt alle Klassifikationsalgorithmen in der Gesamtübersicht.

Ergebnisse

Auf den Abbildung 5.2, 5.3 und 5.4 wird deutlich, dass alle Klassifikationsalgorithmen deutlich bessere F1-Scores aufweisen, wenn sie zuvor auf der Grundlage von einzelnen Sätzen (rechte Diagrammhälfte) trainiert wurden. So hatte die Performance des *Modells* im Durchschnitt $1,44\%$ bessere F1-Scores, die *Realdaten - Bewertungen* $3,34\%$ und die *Realdaten - Sätze* sogar $6,0\%$ bessere F1-Scores wenn das Modell zuvor auf einzelnen Sätzen trainiert wurde. Dies bestätigt auch die Annahme aus Kapitel 3.1.1, dass es sinnvoll ist, Bewertungen in einzelne Sätze zu zerlegen, um diese dann gezielter labeln zu können. Mit dieser Herangehensweise konnten deutlich bessere Performance-Werte erzielt werden. Betrachtet man den Einsatzzweck dieser ML-Modelle, dem Analysieren von unbekannten App-Bewertungen, ist gemäß Abbildung 5.5, das CNN welches auf einzelnen Sätzen trainiert wurde, die beste Wahl. Hier wurde ein F1-Score von $95,8\%$ erreicht um Bewertungen aus Realdaten zu klassifizieren. Weiterhin lässt sich in aus den Tabellen 5.3 und 5.2 ableiten, dass ein Cleaning der Eingabedaten in den meisten Fällen eine Performancesteigerung herbeiführt. Bei der Rechtschreibkorrektur kommt es auf das verwendete Klassifikationsverfahren bzw. das genutzte Word Embedding an. Hier kann keine abschließende Empfehlung gegeben werden. Detaillierte Informationen zu den Varianten sind in Anhang D.1 aufgeführt. Des Weiteren sind in diesem Anhang alle Lernkurven der Deep Learning Verfahren, aller getesteten Varianten angehängt. Bei den Verfahren FNN und CNN, ergab sich eine Verschlechterung der F1-Scores bei der Verwendung der Word Embeddings. Einzig bei dem RNN-Netz ergab sich eine leichte Verbesserung. Bewertet man die F1-Scores der beiden Word Embeddings: „word2vec - Ground Truth“ und des „GoogleNews-vectors-negative300“, so schneidet der Google News Vektor im Durchschnitt besser ab. Tabelle 5.4 zeigt die ermittelten F1-Scores der jeweiligen Verfahren, abhängig von der Art der verwendeten Trainingsdaten. Die ermittelten Werte ergaben sich aus den Varianten (cleaning, Spell Correction) der Klassifikatiosnverfahren, indem der Durchschnitt der jeweiligen vier Werte zur Berechnung genutzt wurden. In Abbildung 5.1 sind die durchschnittlichen F1-Scores nach verwendetem

Word Embedding und Trainingsdatenset dargestellt. Dort lässt sich erkennen, dass das eigens trainierte Word Embedding *word2vec - Ground Truth* am schlechtesten abschneidet, wohingegen das *Goolge News* Word Embedding bessere Ergebnisse liefert. Die ermittelten Werte sprechen gegen die Annahme, das Word Embeddings die Performance von Klassifikationsverfahren die zur Textklassifikation genutzt werden, verbessern [8]. Dies lässt sich damit erklären, dass wenn in der Architektur der tiefen neuronalen Netze kein vortrainiertes Word Embedding eingesetzt wurde, stattdessen ein Embedding Layer eingesetzt wurde der die Wortvektoren verarbeitete. In der Arbeit von Qi et al. [35] wird dieser Umstand detailliert beschrieben. Nach dieser Arbeit, hat ein vortrainiertes Word Embedding nur unter bestimmten Bedingungen Vorteile, nicht aber im Allgemeinen. Da das hier zu klassifizierende Problem sehr spezifisch ist und das Trainingsdatenset ausreichend groß ist, ist es demnach wahrscheinlich, dass ein vortrainiertes Word Embedding keinen signifikanten Vorteil bringt. Dies ist hier offensichtlich der Fall, wie die ermittelten Ergebnisse aus Abbildung 5.1 und Tabelle 5.4 beweisen.

		Train of Reviews	Train on Sentences
Classifier	embeding_vector	F1	F1
FNN	ohne	0.851	0.871
	word2vec - Ground Truth	0.63	0.669
	GoogleNews-vectors-negative300	0.748	0.823
CNN	ohne	0.855	0.876
	word2vec - Ground Truth	0.709	0.809
	GoogleNews-vectors-negative300	0.841	0.868
RNN	ohne	0.798	0.862
	word2vec - Ground Truth	0.737	0.828
	GoogleNews-vectors-negative300	0.806	0.854

Tabelle 5.4: Vergleich der eingesetzten vortrainierten Word Embeddings nach F1-Score und eingesetzter Trainingsstrategie.

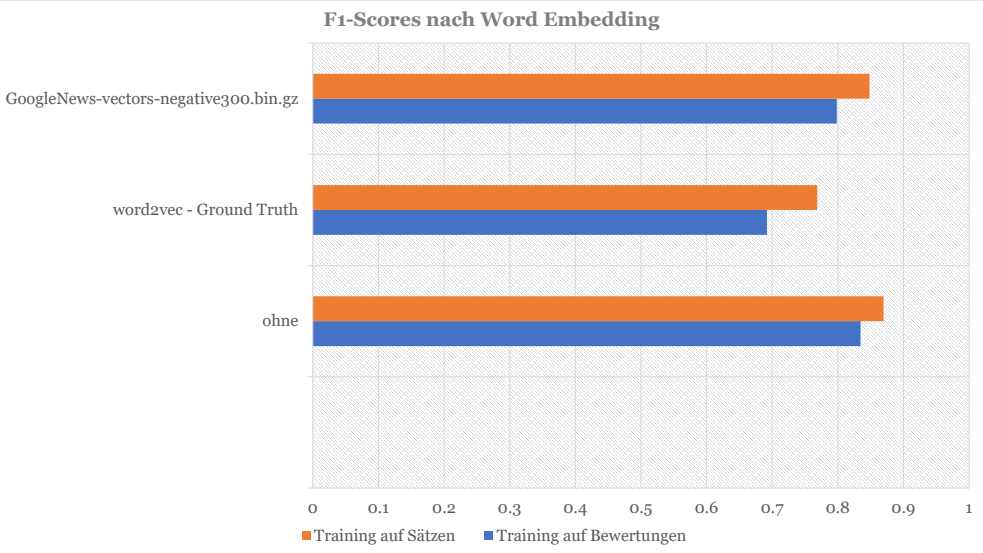


Abbildung 5.1: Vergleich vortrainierter Word Embeddings mit kummulierten F1-Scores unterteilt nach verwendeter Trainingsstrategie.

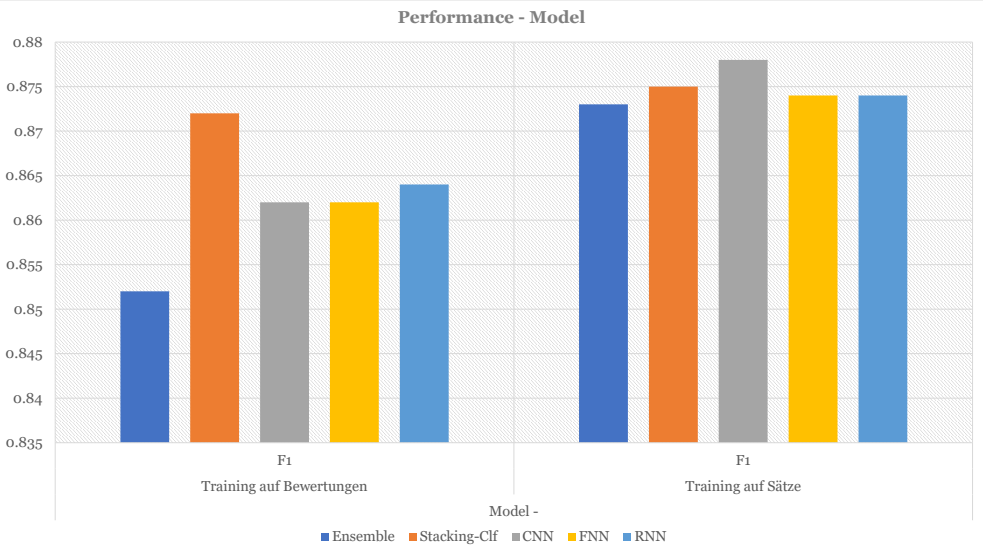


Abbildung 5.2: Links: F1-Scores für das Modell welches auf ganzen Bewertungen trainiert wurde. Rechts: F1-Scores für das Training auf einzelnen Sätzen.

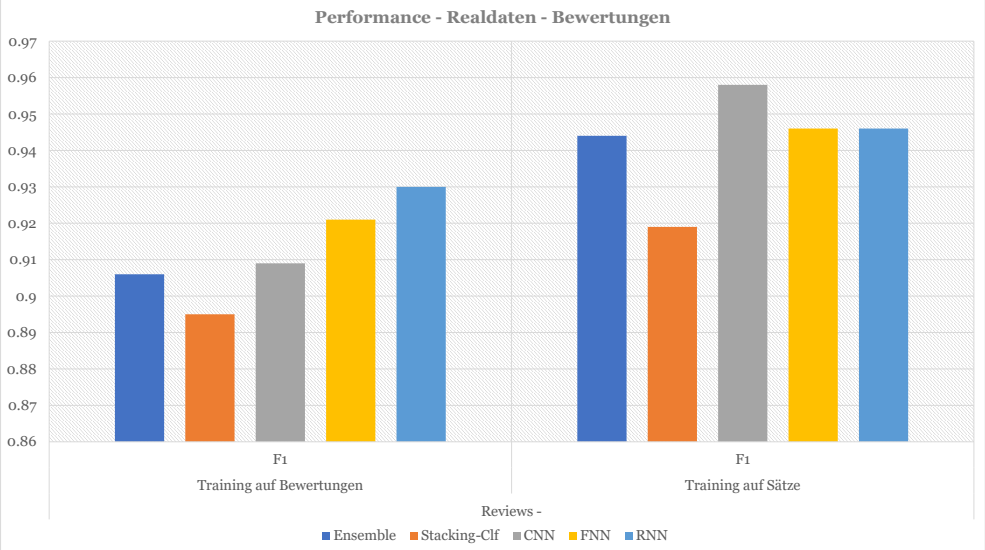


Abbildung 5.3: Validierung des Modells anhand von Realdaten welche aus *ganzen Bewertungen* bestanden. Links: F1-Scores für die Validierung des Modells welches basierend auf ganzen Bewertungen trainiert wurde. Rechts: F1-Scores für die Validierung des Modells welches basierend auf einzelnen Sätzen trainiert wurde.

Abbildung 5.4: Validierung des Modells anhand von Realdaten welche aus *einzelnen Sätzen* bestanden. Links: F1-Scores für die Validierung des Modells welches basierend auf ganzen Bewertungen trainiert wurde. Rechts: F1-Scores für die Validierung des Modells welches basierend auf einzelnen Sätzen trainiert wurde.

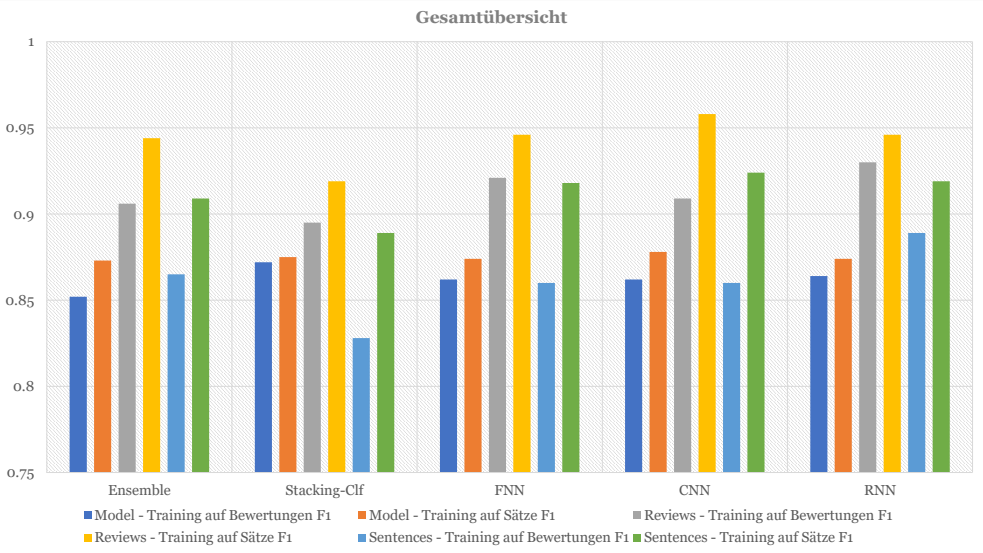


Abbildung 5.5: Gesamtübersicht zu allen Klassifikationsalgorithmen, farblich unterteilt nach genutztem Trainings- und Validierungssets. Es sind die F1-Scores abgebildet.

5.1.3 Knowledge Base - Kategorien

In Tabelle 5.5 ist die Aufteilung der 1.277 Sätze, auf die erkannten Kategorien dargestellt. Dabei war es möglich einen Satz mehrere Kategorien zuzuteilen. Es folgt eine Beschreibung der Kategorien:

1. other: Alles, was in keine andere Kategorie passt.
2. security_breach: Ein Datenschutzproblem resultierte aus einer Sicherheitslücke.
3. permissions: Wenn zu viele oder nicht benötigte Berechtigungen von der App angefordert werden.
4. data_transfer_to_third: Daten werden an Dritte weitergegeben.
5. bad_privacy_policy_or_user_agreement: Nutzungsvereinbarung oder Datenschutzrichtlinien sind: schlecht, fragwürdig, unverständlich oder nicht vorhanden.
6. violation_data_economy: Verstoß gegen das Prinzip der Datensparsamkeit oder Datenvermeidung (nicht technische Ebene).
7. request_privacy: Lediglich eine Anfrage nach „mehr“ Privatsphäre.
8. claim_privacy_issue: Reine Vermutung, Behauptung oder Aussage eines Nutzers: „Verletzung der Privatsphäre“ oder Empfehlung, eine andere App zu verwenden.
9. spy_and_steal_data: Sammlung von Benutzerdaten (Geolokalisierung, Aufnahme von Fotos ...) (Daten stehlen, Daten ausspionieren).
10. advertising: Verlust der Privatsphäre durch Werbung.

Diese zehn Kategorien wurden aus den erkannten 1.277 Privacy Issues eruiert. Die drei Kategorien mit den meisten erkannten Privacy Issues, waren *claim_privacy_issue (458), spy_and_steal_data(376) und permissions (301)*. Aber auch Kategorien mit weniger Vorkommen, sollten nicht außer Acht gelassen werden, da die Vorkommenshäufigkeit in keiner Relation mit dem Schadensausmaß zusammenhängt. So ist die Kategorie *data_transfer_to_third* mit 71 Vorkommen nicht sehr häufig vertreten, birgt aber ein erhebliches Schadenspotenzial da bspw. durch den Abfluss von Kreditkartendaten, Passwörtern oder Sozialversicherungsnummern an unberechtigte Dritte, sehr großer Schaden entstehen kann.

5.1.4 Beste Klassifikationsvariante

Auf Grundlage aller durchgeführten Untersuchungen der Klassifikationsalgorithmen, lässt sich eine Empfehlung für die Wahl einer passenden Variante

cat_id	Kategorie	Anzahl Vorkommen	Anteil
1	other	137	10.7%
2	security_breach	48	3.8%
3	permissions	301	23.6%
4	data_transfer_to_third	71	5.6%
5	bad_privacy_policy_or_user_agreement	167	13.1%
6	violation_data_economy	53	4.2%
7	request_privacy	31	2.4%
8	claim_privacy_issue	458	35.9%
9	spy_and_steal_data	376	29.4%
10	advertising	12	0.9%
	Insgesamt:	*1654*	

Tabelle 5.5: Übersicht über die Verteilung der 1.277 Sätze die Privacy Issues enthielten, auf die einzelnen Kategorien (Mehrfachauswahl möglich). Zusätzliche Angabe der Anteile bezogen auf 1.277 Sätze.

aussprechen. Wie gezeigt wurde, ist ein CNN mit Embedding Layer (ohne vortrainiertes Word Embedding) und bereinigten (cleaning) Eingabedaten und aktivierter Rechtschreibkorrektur (Spell_correction), mit einem F1-Score von $95,8\%$, die geeignetste Variante um Privacy Issues in App Bewertungen zu finden. Das Modell muss zuvor auf einzelnen Sätzen trainiert worden sein.

5.2 Inter Rater Agreement

5.2.1 Cohens Kappa Statistik

Die gelabelten Daten wurden mittels der Cohens Kappa Statistik auf Übereinstimmung geprüft. Dazu wurde eine randomisierte Stichprobe von 360 bereits gelabelten Bewertungen durch eine weitere Person gelabelt. Daraus ergaben sich teils unterschiedliche Label bei den zu bewertenden Bewertungen. In einer ersten Phase nach 60 gelabelten Bewertungen, wurden die Unterschiede gesichtet und anschließend unter den Ratern besprochen. Somit konnten Missverständnisse ausgeräumt und Differenzen besprochen werden und die Übereinstimmung verbessert werden. Anschließend wurden weitere 300 Bewertungen überprüft. Einige beispielhafte Unterschiede beim Labeling sind im Unterkapitel Beispiele aufgeführt.

Ergebnis

Nach dem Analysieren der 360 Proben der Bewertungen, wurde ein Cohens Kappa Score von 0.823 erreicht. Somit wurde das gesteckte Ziel aus Kapitel 3.5.2 von mindestens 0.8 erreicht. Diese 360 Bewertungen bestanden aus 1.280 Sätzen von denen 99 unterschiedlich gelabelte Sätze im Trainingsdatenset verbleiben.

Beispiele

In diesem Unterkapitel werden einige Beispiele zu Unterschieden beim Labeling gezeigt, bei denen auch nach einer Besprechung beider Rater kein Konsens gefunden werden konnte. Die gezeigten Beispiele stellen dabei nur einen kleinen Ausschnitt der unterschiedlichen Label dar.

1. *F-secure doesn't really collect anything except necessary things like your credit card if you buy it.*

    - Begründung Rater 1: Kein Privacy Issue, da der Nutzer selbst sagt, dass nur benötigte Daten gesammelt wurden.
    - Begründung Rater 2: Es handelt sich um ein Privacy Issue, da die benötigten Daten zu früh erhoben wurden. Die App ist anfangs kostenlos, es besteht aber die Möglichkeit durch einen In-App Kauf eine kostenpflichtige Version zu erwerben wozu dann auch die Kreditkartendaten benötigt würden. Somit wurde hier gegen das Gebot der Datensparsamkeit verstoßen.

2. *That actually looks like a security breach!*

    - Begründung Rater 1: Hier ist keine direkte Verbindung zu einer Datenschutz- oder Privatsphäreverletzung zu erkennen, lediglich die Aussage, dass es eine Sicherheitsverletzung unbekannter Art gibt. Da hier nicht näher spezifiziert ist, um welche Art Sicherheitslücke es sich handelt, sind hier zu wenig Informationen verfügbar um von einem Privacy Issue auszugehen.
    - Begründung Rater 2: Rater 2 begründet das vergebene Label damit, dass es sich bei einer Sicherheitslücke automatisch auch um ein Datenschutz- oder Privatsphäreproblem handeln muss.

3. *...if you think that somehow knowing the location of sex offenders will save your children from random acts of perversion.*

    - Begründung Rater 1: Auch hier ist Rater 1 der Bezug zu einem Privacy Issue zu ungenau formuliert. Es sind zu wenig Informationen im strittigen Satz vorhanden, als das von einem Verstoß gegen die Privatsphäre ausgegangen werden kann.
    - Begründung Rater 2: Hier wird nach Ansicht von Rater 2, klar die Privatsphäre der genannten Sexualstraftäter verletzt, da hier die geografische Lokalisation der Täter angesprochen wird. Somit werden personenbezogene Daten veröffentlicht was eine Verletzung der Privatsphäre darstellt.

Abschließend lässt sich festhalten, dass unterschiedliche Personen eine unterschiedliche Empfindlichkeit bei der Kategorisierung von Privacy Issues haben, obwohl eine genaue Definition vorgegeben wurde. Dies lässt sich unter anderem mit der bisherigen Erfahrung der Rater auf dem Gebiet des Privatsphäre- und Datenschutzes zurückführen, aber auch auf die Eigenschaft der Rater, dass Verstöße teils explizit genannt werden müssen wohingegen andere Rater bereits das Ankündigen eines Verstoßes genügt um ein positives Label festzulegen.

Kapitel 6

Diskussion

In diesem Kapitel werden die Ergebnisse dieser Arbeit diskutiert und Limitationen aufgezeigt. Dabei werden Erwartungen, Ursachen und Folgen im Bezug auf die Ergebnisse dargelegt. Abschließend werden einige Herausforderungen besprochen, die während der Implementierungsphase des Ansatzes bewältigt werden mussten.

6.1 Interpretation der Ergebnisse

In dieser Arbeit sollte ein Ansatz entwickelt werden, mit dem es möglich ist Privay Issues in App Store Bewertungen finden zu können. Dies wurde mit Hilfe von NLP-Verfahren und ML-Algorithmen umgesetzt, sodass abschließend ein Klassifikationsverfahren trainiert werden konnte. Mit dem daraus resultierenden Modell, konnten Privacy Issues in App Bewertungen gefunden und analysiert werden. Auch der gewählte Ansatz der Vorselektion von Bewertungen unter Verwendung eines word2vec Modells, bestätigte die Annahme, den manuellen Labeling-Vorgang effizienter gestalten zu können. So konnten Bewertungen mit deutlich geringerem Aufwand vorselektiert und anschließend manuell gelabelt werden. Die damit verbundene Zeitersparnis konnte genutzt werden, um weitere Analysen, bspw. im Bereich der Dopplungen von Bewertungstexten, durchführen zu können. Die entwickelte Knowledge Base soll weitere Forschungen erleichtern, indem identifizierte Privacy Issues leicht zugänglich gemacht wurden. Ergebnisse wurden in mehreren Bereichen erzielt. Dazu zählt die bereits erwähnte Knowledge Base, die trainierten ML-Modelle mit denen neue Privacy Issues in Bewertungen mit einem F1-Score von $95,8\%$ identifiziert werden können, aber auch das trainierte word2vec Modell, welches zukünftig genutzt werden kann um den Keyword-Katalog zu erweitern und den Suchbereich damit auszudehnen. Die-

se Ergebnisse können anschließend genutzt werden, um die anfangs erwähnte Filterfunktion in den App Stores, zu implementieren. Damit wären Nutzer in der Lage App-Bewertungen nach unterschiedlichen Themenschwerpunkten zu filtern und somit Aspekte der Softwarequalität zu verbessern. Diese Funktion könnte zukünftig als direkter Kommunikationskanal zwischen Nutzern und App-Entwicklern genutzt werden um einen Vorteil für beide Seiten zu schaffen.

Wie eingangs bereits erwähnt, ist in diesem speziellen Bereich des Privatsphäre- und Datenschutzes in App Store Bewertungen wenig Forschung durchgeführt worden. Bedenkt man wie die Nutzung von mobilen Telefonen zugenommen hat und vor allem wie sich das Nutzungsverhalten (Online-Banking, betriebliche Kommunikation per Email, WhatsApp, etc., oder auch Gesundheitsapps die bspw. den Insulinspiegel von Diabetikern überwachen und steuern) geändert hat, ist es an der Zeit den App-Nutzern mehr Optionen zur Kontrolle ihrer persönlichen Daten zu bieten, um so mehr Vertrauen zu schaffen.

6.2 Limitationen

In dieser Arbeit wurden für das Trainingsdatenset lediglich 118 Apps auf Privacy Issues untersucht. Da die zwei betrachteten App Stores mehrere 10.000 Apps beinhalten, würde die Analyse weit über den Rahmen dieser Arbeit hinausgehen. Die betrachteten Apps stellen daher lediglich einen kleinen Teil aller verfügbaren Apps dar. Auch der erstellte Keyword-Katalog mit seinen 29 festgelegten Begrifflichkeiten, stellt nur einen kleinen Ausschnitt der Möglichkeiten dieses Ansatzes dar. Auch hier mussten im Vorfeld sinnvolle Grenzen für die Größe und somit des Aufwandes gesetzt werden. Eine Erweiterung des Katalogs kann in folgenden Forschungen aber sehr einfach durchgeführt werden, sodass sich der Suchradius noch deutlich verfeinern lässt. Ein weiterer Punkt der nicht Bestandteil dieser Arbeit war, ist das Melden oder Weiterverfolgen aufgedeckter Privatsphäre- und Datenschutzverletzungen. Dies hätte zum einen an die App-Entwickler selbst und zum anderen an die App Store Betreiber erfolgen können. Auch das „Warnen“ von Nutzern durch bspw. selbst Verfassen von eigenen Bewertungen in den entsprechenden Apps, war nicht Bestandteil dieser Arbeit. Dies wäre ohnehin schwierig gewesen, da die gefunden Privacy Issues lediglich auf Aussagen Dritter basierten und nicht von mir auf Echtheit geprüft wurden. Genau dies ist noch ein weiterer essentieller Punkt der eine Limitation darstellt, da kein gefundenes Privacy Issue tatsächlich im Rahmen dieser Arbeit auf Echtheit überprüft wurde. Alle Privacy Issues

beziehen sich lediglich auf Aussagen von Nutzern der App Stores. Genauer gesagt müssen diese Nutzer eine App nicht einmal selbst installiert haben um eine Bewertung zu verfassen. Daher lässt sich auch die Echtheit aller Aussagen in den Bewertungen, nur schlecht nachweisen oder kontrollieren. Dies hat natürlich Auswirkungen auf die erzielten Ergebnisse dieser Arbeit, so sollte dieser Punkt jedem Nutzer bewusst sein, wenn Bewertungen gelesen werden. Die Ergebnisse speziell in der Knowledge Base, sind daher nur vermeintliche Privatsphäre- und Datenschutzverfehlungen. Bestätigen lässt sich die Echtheit im Rahmen dieser Arbeit nicht, wohl aber lassen sich weitere Forschungen in diesem Bereich anknüpfen.

6.3 Herausforderungen

Während der Implementierungsphase traten einige Herausforderungen auf, die in diesem Kapitel erläutert werden. Außerdem werden verschiedene Lösungsansätze dargelegt und die letztendlich gewählte Lösung präsentiert.

Geoblocking zu amerikanischem App Store

Beim Geoblocking handelt es sich um eine Sperre, die meist auf Grundlage von IP-Adressen eingerichtet wird. So können bspw. Inhalte im Internet auf bestimmte Regionen beschränkt werden. Ein solches Geoblocking setzt auch der Google Play Store ein, um Nutzern nur Apps aus ihrer Region anzuzeigen.

Da diese Arbeit auf der Grundlage von englischer Sprache und somit von englischsprachigen Bewertungen basieren sollte, war es nötig Apps aus den App Stores zu identifiziert, die aus dem amerikanischen App Store stammten. Dies war beim Google Play Store nicht ohne weiteres möglich, da ein Zugriff aus Deutschland, lediglich Zugriff auf den deutschen Play Store erlaubt. Ein manuelles ändern der Lokation ist aus Deutschland heraus nicht möglich und somit auch der Zugriff auf ausländische Play Store Inhalte. Da das Angebot der Apps stark vom länderspezifischen Store abhängt, musste der Zugriff auf den amerikanischen Store auf andere Weise erfolgen. Der Play Store verwendet die Quell-IP Adresse des Nutzers um die Landeszugehörigkeit zu überprüft. Um dies zu umgehen wurde ein VPN eingesetzt welches einen Tunnel zu einem amerikanischen VPN-Server aufbaute und somit eine amerikanische IP-Adresse vortäuschte. Mit dieser amerikanischen IP-Adresse war es anschließend möglich, auf alle Inhalte des amerikanischen App Stores zuzugreifen.

Fremdsprachige Bewertungen

Während der Überprüfung einiger heruntergeladener Bewertungen fiel auf, dass einige Bewertungen nicht in englischer Sprache verfasst wurden. Teilweise waren die Bewertungen auch in gemischten Sprachen, wie Englisch und Spanisch, verfasst. Daher wurde versucht diese Bewertungen aus den heruntergeladenen Daten zu entfernen. Versuche diese fremdsprachigen Texte zu entfernen, erweisen sich allerdings aus folgenden Gründen, als deutlich schwieriger als vorher angenommen:

1. 1. Variante: *Überprüfung des Textes auf englischsprachigen Inhalt.* Dieser Ansatz ist möglich, allerdings kann muss hier jedes Wort bspw. mittels Wörterbucheintrag überprüft werden. Dies ist nicht nur sehr zeitaufwändig, es besteht auch das Problem, dass in den Texten häufig Schreibfehler oder Slang enthalten ist, der sich so nicht überprüfen lässt. Es würden daher viele Bewertungen entfernt werden, die nicht der korrekten englischen Sprache entsprächen. Dies würde zu einem zu hohen Informationsverlust führen, weswegen diese Lösungsvariante nicht in Frage kam.

2. 2. Variante: *Verwendung von externen Programmbibliotheken um die Sprache einer Bewertung zu analysieren.* Ähnlicher Ansatz wie Variante 1, jedoch bessere Ausführung der Analyse, da die Programmbibliothek speziell für das Identifizieren von Sprachen entwickelt wurde. Hier ergabt sich jedoch auch der Nachteil, das gerade die mehrsprachigen Bewertungen verloren gingen, da diese von der verwendeten Programmbibliothek entfernt wurden. Auch diese Lösung hätte einen Informationsverlust zur Folge gehabt. Somit wurde diese Lösungsvariante ebenfalls nicht umgesetzt.

3. 3. Variante: *Keine Bereinigung durchführen.* Das Auftreten dieser anderssprachigen oder mehrsprachigen Bewertungen könnte auch außer Acht gelassen werden, da es nur wenige Bewertungen betraf. Dies war die eingesetzte Variante, da bei allen anderen Varianten der Aufwand und der Informationsverlust als zu groß eingeschätzt wurde. Allerdings wurde darauf geachtet, dass in den Trainingsdaten der Klassifikationsalgorithmen und in den Realdaten, keine mehrsprachigen oder anderssprachigen Bewertungen vorkamen. Damit wurde angestrebt, dass die trainierten ML-Modelle nur englischsprachige Privacy Issues finden können.

JSON Format ungeeignet

Für des Trainings des word2vec Modells, wurden alle 82 Millionen heruntergeladenen Bewertungen verwendet. Diese wurden im JSON-Format auf einem lokalen Datenträger gespeichert. Insgesamt waren dazu ca. 25 GB Speicher erforderlich. Diese Daten mussten vor dem Training des Modells geladen werden, was mit einem *Memory Error* terminiert wurde. Die Ursache dafür, liegt in der Art mit der eine JSON Datei geladen wird. Um eine JSON Datei zu verarbeiten, muss diese komplett in den Hauptspeicher geladen werden. Dabei wird noch zusätzlicher Speicher benötigt der über die eigentliche Größe der zu verarbeitenden JSON Datei hinausgeht. Der Computer der für das Training verwendet wurde, verfügte über 64 GB Arbeitsspeicher, was für die Verarbeitung der 25 GB JSON Daten nicht ausreichend war. Die Lösung des Problems, war ein anderen Dateiformat und somit eine andere Verarbeitungsstrategie der Daten. Somit wurden alle JSON Daten in das Comma-separated values (CSV)-Format konvertiert. Bei der Verwendung des CSV-Formats, ist es nicht notwendig, alle Daten zunächst in den Arbeitsspeicher zu laden um ihn zu verarbeiten. Vielmehr kann die Verarbeitung „on the fly“ durchgeführt werden, was zu einer hohen Ersparnis an Arbeitsspeicher führt. Des Weiteren ist der Speicherbedarf mit ca. 7,5 GB deutlich geringer als beim JSON-Format.

Kapitel 7

Verwandte Arbeiten

In diesem Kapitel werden Arbeiten vorgestellt, die sich inhaltlich mit ähnlichen Themen wie diese Arbeit, beschäftigen. Nachfolgend eine Auflistung ausgewählter Arbeiten:

- Sentiment Analysis of App Store Reviews [41]
- Mining User Opinions in Mobile App Reviews: A Keyword-based Approach [45]
- User Feedback in the AppStore: An Empirical Study [31]
- How can I improve my app? Classifying user reviews for software maintenance and evolution [32]
- Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews [25]
- A Natural Language Processing Based Approach Using Stochastic Petri Nets For Understanding Software Requirement Specifications [3]
- Polisis; Automated Analysis and Presentation of Privacy Policies Using Deep Learning [13]
- Using Natural Language Processing to Detect Privacy Violations in Online Contracts [42]
- Mining User Intentions from Medical Queries: A Neural Network Based Heterogeneous Jointly Modeling Approach [47]
- Deep Learning applied to NLP [24]
- Revealing the unrevealed: Mining smartphone users privacy perception on app markets [14]

Sangani et al. [41] beschäftigten sich mit der Sentimentanalyse von Bewertungen aus dem Google Play Store, um Entwicklern eine Möglichkeit zu geben positives und negatives Feedback von Nutzern auszuwerten. Dazu wurden Bewertungen von sieben Apps auf subjektive Wörter mit Hilfe des Verfahrens „Stochastic Gradient Descent“ untersucht. Anschließend identifizierten die Entwickler auf Grundlage dieser subjektiven Wörter, Themenbereiche die für Entwickler von Interesse sein könnten. Anhand dieser Themenbereiche in Kombination mit den gefundenen subjektiven Wörtern, wurden verschiedene Metriken wie dem durchschnittlichen Score, zur Analyse der Bewertungen genutzt.

Vu et al. [45] stellten ein Verfahren vor, welches durch die Eingabe von Keywords, passende Bewertungen aus einer Liste bestimmter Apps aus App Stores ausgeben konnte. Realisiert wurde dieses Verfahren unter anderem durch den word2vec Algorithmus, der auch in dieser Arbeit verwendet wurde. Weiterhin wurde eine Metrik „contrast score“ implementiert, welche angab ob ein Keyword häufiger in positiven oder in negativen Bewertungen vorkam. Auf diese Weise konnte ermittelt werden, ob das Keyword eine negative oder positive Bedeutung hat.

Pagano et al. [31] führten eine empirische Studie auf Grundlage von Bewertungen im Apple App Store durch. Eine zentrale Frage dieser Arbeit war, ob sich Bewertungen in App Stores, als Kommunikationskanal zwischen Entwicklern und Nutzern eigneten. Weiterhin wurde untersucht, zu welchem Zeitpunkt nach dem Release einer App, die meisten Bewertungen abgegeben wurden. Anschließend konnten alle verfügbaren Bewertungen thematisch sortiert und in vier Themenbereiche aufgegliedert werden.

Panichella et al. [32] stellten ein Verfahren vor, mit dessen Hilfe sich Bewertungen aus App Stores in Kategorien einordnen ließen. Dazu wurden die drei Verfahren NLP, Textanalyse und Sentiment Analyse kombiniert, um Bewertungen automatisch in eine Kategorie einzuordnen.

McIlroy et al. [25] untersuchten wie Bewertungen aus App Stores automatisch gelabelt werden können, um unter anderem ein besseres Verständnis für den Inhalt der Bewertungen zu erhalten. Für das automatische Labeling, wurden verschiedene Klassifikationsalgorithmen eingesetzt und deren Performance verglichen.

Ashtankar et al. [3] stellten eine Methode vor, die es ermöglicht die Mehrdeutigkeit von Texten in einem Graphen darzustellen. Zu diesem Zweck wurden Petri-Netze eingesetzt, die das Problem der Mehrdeutigkeit lösen sollten. Angewandt wurde die Methode auf Softwareanforderungen die in

natürlicher englischer Sprache verfasst waren. Gefundene Anforderungen wurden in einem Graphen dargestellt mit der Möglichkeit daraus im Anschluss Testfälle abzuleiten.

In der Arbeit von Harkous et al. [13] wird eine Methode vorgestellt, um Datenschutzrichtlinien in vorher definierte Datenschutzkategorien einzuordnen. Es wurden verschiedene Klassifikationsalgorithmen mit gelabelten Datensätzen von Datenschutzrichtlinien trainiert und im Anschluss deren Performance verglichen.

Silva et al. [42] setzen verschiedenen NLP-Tools ein, um Privatsphäreverletzungen in Verträgen zu finden. Es wurden Methoden analysiert, die personenbezogene Daten in Texten identifizieren konnten. Abschließend wurden verschiedene Einsatzmöglichkeiten eines solchen Verfahren beschrieben und welche Vorteile beim Einsatz solcher Methoden, in Bezug auf Privatsphäreverletzungen entstehen.

In der Arbeit von Zhang et al. [47] wird ein Verfahren vorgestellt, welches Nutzerintentionen aus medizinischen Textanfragen, mittels neuronalen Netzen extrahieren konnte. Dabei wurde speziell das Problem der Intentionsanalyse untersucht, welche sich von der Sentiment Analyse oder der Textklassifikation stark unterscheidet. Verfahren aus den Bereichen NLP und ML wurde eingesetzt um dieses Problem zu untersuchen. Abschließend wurde die Performance unterschiedlicher Netzarchitekturen anhand festgelegter Metriken verglichen.

Lopez et al. [24] beschreiben die Möglichkeit, ein CNN für Probleme aus dem Bereich NLP einzusetzen. Weiterhin wurden weitere Deep Learning Verfahren vorgestellt und deren Performance verglichen.

Hatamian et al. [14] beschrieben ein Verfahren, welches unter Zuhilfenahme von ML-Verfahren, NLP-Algorithmen und Sentimentanalyse, App Store Bewertungen auf Privacy Aspekte untersuchte. Dazu wurde aus 812.899 Bewertungen von 200 Apps ein Keyword-Katalog entwickelt, welcher anschließend in einem unüberwachten Lernverfahren eingesetzt wurde, um ein automatisiertes Labeling zu realisieren. Die Sentimentanalyse wurde genutzt um positive Bewertungen im Vorfeld zu filtern.

Zusammenfassung

Zusammenfassend existieren einige Arbeiten die sich mit der Analyse von App Bewertungen befassen sowie Arbeiten zum Thema Datenschutz und Privatsphäre. Untersuchungen die sich mit der Analyse von Datenschutz-

und Privatsphäreverletzungen in Bewertungen aus App Stores befassen, sind jedoch sehr selten. Daher setzt diese Arbeit ihren Schwerpunkt genau dort und kombiniert ML-Verfahren mit NLP-Problemen um speziell den Themenbereich des Datenschutzes und des Privatsphäreschutzes zu untersuchen.

7.1 Abgrenzung der Arbeit

Diese Arbeit zeigt einen neuartigen Ansatz, um Privatsphäreverletzungen in Bewertungen von Apps aus App Stores wie dem Google Play Store oder dem Apple App Store zu finden. Dabei wird ein spezielles Verfahren vorgestellt, um eine Vorselektion der Daten nach verschiedenen Merkmalen zu ermöglichen. Durch diese Vorselektion lässt sich das Erstellen von Trainingsdaten für ML-Algorithmen vereinfachen, da auf diese Weise geeignete Trainingsdaten in einer großen Anzahl an Bewertungen schneller gefunden werden können. Abschließend wird der Aufbau einer Knowledge Base beschrieben, in der identifizierte Privatsphäreverletzungen in verschiedene Kategorien einsortiert werden können, um so eine bessere Übersicht der gefundenen Bewertungen zu ermöglichen.

Eine Sentimentanalyse wie sie von Sangani et al. [41] durchgeführt wurde, kann bei meinem Verfahren nicht genutzt werden, da Privatsphäreverletzungen in Bewertungen nicht in Sentimenten ausgedrückt werden. Dennoch kann die Sentimentanalyse dabei helfen, die Stimmung einer identifizierten Privatsphäreverletzung zu analysieren. Aus dem Verfahren wie Vu et al. [45] es beschrieben haben, kann der Ansatz der Keyword-Suche mittels word2vec verwendet werden, um die Vorselektion von Bewertungen zu ermöglichen. Bei dieser Arbeit wurden die Bewertungen allerdings nicht auf Datenschutz- oder Privatsphäreverletzungen analysiert. Eine wichtige Erkenntnis lieferten Pagano et al. [31], die feststellten, dass sich Bewertungen in App Stores als Kommunikationskanal zwischen Entwicklern und Nutzern eignen. Diese Erkenntnis ist auch für diese Arbeit wichtig, da so davon ausgegangen werden kann, dass sich Nutzer und Entwickler über das Medium „Bewertungen“ austauschen können. Einen Ansatz um NLP-Verfahren und ML-Verfahren in Kombination auf Bewertungen anzuwendenden, stellten Panichella et al. [32] vor. Sie stellten verschiedene Methoden vor und verglichen diese. Die Erkenntnis von Panichella et al., dass Privatsphäre Themen nur sehr selten in Bewertungen enthalten sind, gab den Anstoß dazu eine Möglichkeit der Vorselektion von Bewertungen zu entwickeln um eine möglichst effiziente Suche zu ermöglichen. Nach diesen Untersuchungen wäre nur mit ungefähr zwei bis drei Privatsphäreverletzungen auf 100 Bewertungen zu rechnen gewesen. Um also 2.000 Bewertungen zu erhalten, die Privatsphä-

reverletzungen beinhalten, hätten ungefähr 100.000 Bewertungen manuell durchsucht werden müssen. Dies wäre aus Zeitgründen nicht durchführbar gewesen und somit wurde der Ansatz der Vorselektion mittels word2vec implementiert. Die Arbeit von Hatamian et al. [14] analysierten App Store Bewertungen auf Privatsphäreverletzungen. Diese Arbeit ist eine der wenigen wissenschaftlichen Versuche, Datenschutz- und Privatsphäreverletzungen in App Store Bewertungen zu untersuchen. Jedoch wurden bei dieser Methode lediglich 812.899 Bewertungen von 200 Apps untersucht bzw. in die Trainingsmasse aufgenommen. Der aus dieser Masse entstandenen Keyword-Katalog, ist demnach sehr auf die 200 Apps eingeschränkt. Dies stellt eine starke Einschränkung des Ansatzes dar, welcher in dieser Arbeit, durch die große Masse an analysierten Daten, nicht existiert. Auch wurde kein manuelles verifiziertes Labeling durchgeführt, was zu Fehlern in den Trainingsdaten führen kann. Die durchgeführte Sentimentanalyse um positive Bewertungen zu entfernen, kann dazu führen, dass relevante Privacy Issues in positiven Bewertungen verloren gehen. Auch dieser Punkt wurde in dieser Arbeit berücksichtigt, indem „positive Privay Issues“ in der Trainingsmasse existieren. Diese wurden dementsprechend als „kein Privacy Issue“ gelabelt. So wird dem ML-Algorithmus die Möglichkeit gegeben, auch diese Art von Privacy Issues richtig zu klassifizieren.

7.2 Schnittstellen und Anwendungen

In der Arbeit von Jacobsen [16] wird unter anderem eine Möglichkeit zur Visualisierung von App Bewertungen vorgestellt. Ein Teil der Arbeit beschäftigt sich mit einer Webapplikation zur Darstellung einzelner Bewertungen. In Absprache mit Jacobsen, besteht die Möglichkeit, die in dieser Arbeit verwendeten Bewertungen, durch eine Application Programming Interface (API) ebenfalls in die Web Application zu integrieren. Zu diesem Zweck definierte Jacobsen eine Schnittstelle zur Anbindung weiterer Analysetools, wie das hier vorgestellte. Auf diese Weise lassen sich mehrere Analysemodule kombinieren und visuell darstellen.

Kapitel 8

Zusammenfassung und Ausblick

8.1 Zusammenfassung

Ziel dieser Arbeit war es Datenschutz- und Privatsphäreverletzungen in App Store Bewertungen mit Hilfe von Maschinellen Lernen und Natural Language Processings zu finden. Dies wurde durch den Einsatz unterschiedlicher Klassifikations- und Deep Learning Verfahren erprobt um letztendlich die beste Möglichkeit aus einer Vielzahl von Möglichkeiten auszuwählen. Dazu mussten zunächst Bewertungen aus zwei App Stores heruntergeladen werden, um ein word2vec Modell trainieren zu können. Da es keinerlei Schnittstellen zum Download von Bewertungen gibt, mussten eigene Methoden entwickelt werden um Bewertungen herunterladen zu können. Nach der Downloadphase von über 82 Millionen Bewertungen, wurde ein word2vec Modell trainiert um einen Keyword-Katalog aufbauen zu können. Dazu wurden anhand eines initialen Keywords, weitere ähnliche Begriffe unter Zuhilfenahme des word2vec Modells, eruiert. Mit diesem Keyword-Katalog wurden, mit einer selbst entwickelten intelligenten Suche, 118 Apps analysiert. Die so erhaltenen vorselektierten Bewertungen konnten auf diese Weise schnell und effizient manuell gelabelt werden, um sie in einer anschließenden Phase als Trainingsdaten für ein überwachtes Lernverfahren verwenden zu können. So konnte die aufwändige Suche in einer großen Anzahl von Bewertungen deutlich vereinfacht werden, da diese spezielle Art der Vorselektion verwendet wurde. Eine weitere Schwierigkeit waren die häufigen Dopplungen von Bewertungstexten bei vielen Apps. Viele Bewertungen beinhalteten häufig denselben Text, was beim manuellen Labeling zu deutlich mehr Ermüdung führt als mit der entwickelten Vorselektion. Nachforschungen zu diesen Dopplungen ergaben, dass es teilweise nur einfache Sätze, wie „Good App!“ waren aber teilweise auch Bewertungstexte gab, die mehr als 120 Wörter

beinhalteten. Weitere Untersuchungen legten nahe, dass es sich bei diesen langen doppelten Bewertungstexten, vermutlich um Plagiate handeln musste. Diese Bewertungstexte traten häufig innerhalb einer App auf und dies sogar zum gleichen Zeitpunkt. Dies lässt vermuten, dass es sich nicht um „echte“ Bewertungen handelt, sondern um bspw. gekaufte Bewertungen. Dies wurde folgend nicht weiter betrachtet, da diese Dopplungen nicht in den Trainingsdaten vorkamen und deshalb keinen Einfluss auf die Klassifikationsverfahren hatten. Abschließend wurde ein Deep Learning Verfahren eruiert, welches die beste Performance aller untersuchten Verfahren aufwies, um Privacy Issues in App Store Bewertungen finden zu können. Nach der Performancemessung von insgesamt 88 Testvarianten (siehe Tabelle 4.8), wurde ein CNN mit einem F1-Score von $95,8\%$ als bestes Verfahren ausgewählt. Unter anderem bestätigt diese Performance, den gewählten Ansatz dieser Arbeit und kann als bspw. Filtermodul in App Stores eingesetzt werden um Nutzern eine Möglichkeit zu bieten, Bewertungen nach Privacy Issues zu durchsuchen. So können Nutzer und Entwickler verschiedene Aspekte der Softwarequalität gemeinschaftlich verbessern und so für mehr Vertrauen sorgen. Außerdem wurden alle bis zu diesem Zeitpunkt identifizierten 1.277 Privacy Issues in einer Knowledge Base in Form einer SQL-Datenbank abgespeichert und in zehn Privacy Kategorien eingeteilt. So wurde ein einfacher Zugriff für weitere Forschungs- oder Weiterentwicklungszwecke ermöglicht. An dieser Stelle möchte ich dringend auf die Notwendigkeit der Weiterentwicklung und Forschung in diesem Bereich hinweisen. In meiner Arbeit entdeckte ich teils gravierende Verstöße gegen die Privatsphäre und den Schutz persönlicher Daten. Bis zum heutigen Tag existieren Apps in den App Stores, welche bspw. alle Tastatureingaben samt Nutzernamen, Passwörter, Kontodaten, etc. des Nutzers an Dritte weiterleiten. Apps die versprechen Werbung in Browser PopUps zu blockieren aber in der Realität sämtliche aufgerufene URLs an Werbetreibende verkaufen. Apps die Nutzern die Möglichkeit bieten, Fahrdienste von anderen Nutzern in Anspruch zu nehmen, dazu aber persönliche Nutzerdaten wie Telefonnummer oder Namen, an die Fahrdienstleister weiterleiten ohne ein nutzerseitiges Einverständnis. Dies sind lediglich einige Verstöße die ich hier hervorheben möchte, bei der Vielzahl an Apps in den App Stores lässt sich allerdings vermuten, dass es noch deutlich mehr solcher Verstöße gibt. Ich würde mich daher sehr freuen wenn diese Arbeit den Anstoß zu weiteren Forschungsarbeiten in diesem Bereich gibt.

8.2 Ausblick

Großes Verbesserungspotenzial liegt, wie bei vielen ML-Algorithmen, bei der Menge der Trainingsdaten. Hier kann die Trainingsmenge noch deutlich vergrößert werden um zum einen mehr Diversität, und zum anderen auch eine noch bessere Leistung der Klassifizierung zu erreichen. Eine geeignete Möglichkeit stellte Jacobsen [16] in seiner Arbeit bereits vor, und implementierte zu diesem Zweck eine Schnittstelle zu seiner grafischen Oberfläche. Mit dieser ist es möglich, komfortabel weitere Analysen von App-Bewertungen durchzuführen und gleichzeitig die Anzahl der Trainingsdaten zu vergrößern. Eine weitere mögliche Erweiterung wäre, eine Untersuchung von Bewertungen nicht nur auf Privacy Issues zu beschränken, sondern bspw. auch auf andere wichtige Qualitätsaspekte einer Software. Durch den Keyword-Based Ansatz aus Phase 4 und 5 des hier vorgestellten Ansatzes, kann der Analyseprozess leicht auf weitere Qualitätsaspekte erweitert werden. Die auf diese Weise trainierten ML-Modelle können dann in unterschiedlichen Analysemodulen verwendet werden, um App-Bewertungen auf spezielle Aspekte hin zu analysieren. Auch der Bereich der Hyperparameter Optimierung birgt noch Verbesserungspotenzial. So könnten mit einer Optimierung der Hyperparameter der eingesetzten ML-Algorithmen, die trainierten Modelle in ihrer Performance verbessert werden. Da hier eine Vielzahl von Möglichkeiten existiert, wird häufig auf spezielle Verfahren zurückgegriffen, die dabei helfen die optimalen Parameter zu finden. Zwei geläufige Optimierungsverfahren sind *Grid Search* und *Random Search*. Mehr zu diesem Thema und den zwei genannten Verfahren, kann in der Arbeit *Random Search for Hyper-Parameter Optimization* von Bergstra und Bengio [4] nachgelesen werden.

Anhang A

A.1 Verwendete Bibliotheken

In Tabelle A.1 sind die wichtigsten externen Bibliotheken aufgelistet, die bei der Implementierung der Module verwendet wurden.

Name	**Version**	**Beschreibung**
app-store-scraper	0.3.4	Review-Crawler für den Apple App Store
beautifulsoup4	4.9.3	Bibliothek zum Auslesen von Daten aus HTML- und XML-Dateien
colorama	0.4.4	Tool um Textausgaben auf der Konsole farbig auszugeben
en-core-web-lg	2.3.1	Zusatzmodul für spaCy
gensim	3.8.3	Bibliothek zum Training von Vektorembeddings
google-play -scraper	0.1.1	Review-Crawler für den Google Play Store
googletrans	4.0.0.rc1	Übersetzungstool für die Konsole
Keras	2.4.3	Deep-Learning-Bibliothek für Python
Keras-Pre-processing	1.1.2	Dienstprogramme für die Vorverarbeitung von Keras-Datensätzen
mariadb	1.0.6	Connector für Python und Datenbanken
mathplot	0.1	Bibliothek zum Erstellen von Plots
matplotlib	3.3.2	Bibliothek zum Erstellen von Plots
nltk	3.5	Natural Language Toolkit für Python
numpy	1.19.4	Bibliothek zur Handhabung von Vektoren, Arrays, etc.
pandas	1.1.4	Werkzeug zur Datenanalyse und Datenmanipulation

pyspellchecker	0.6.0	Rechtschreibkorrektur gem. Norvig [33]
scikit-learn	0.23.2	Softwarebibliothek zum Maschinelles Lernen
scipy	1.5.4	Analysesoftwareumgebung
seaborn	0.11.1	Datenvisualisierungsbibliothek basierend auf matplotlib
selenium	3.141.0	Tool für Webseitenaufruf aus der Konsole heraus
sklearn	0.0	Softwarebibliothek zum Maschinelles Lernen
spacy	2.3.2	Bibliothek für Natural Language Processing
tensorflow	2.4.1	End-to-End Open-Source-Plattform für maschinelles Lernen
wordcloud	1.8.1	Ausgabe von Bildern mittels Eingaben von Texten

Tabelle A.1: Übersicht der verwendeten Bibliotheken bei der Implementierung der Module.

A.2 Stop Words

Gemäß des Pythonpakets: spacy.lang.en.stop_words.STOP_WORDS
Dokumentation: `https://spacy.io/api/language`

done, herein, front, ca, each, this, n‘t, all, rather, anyone, say, even, noone, him, ’d, as, hereupon, us, whom, been, without, yourself, everywhere, also, her, anything, its, becomes, any, see, almost, both, often, another, mine, well, could, through, everyone, below, last, ‘ve, hence, eight, ‘ll, became, put, ‘d, what, ’m, serious, nowhere, behind, n’t, when, does, afterwards, some, seemed, onto, whither, whose, seems, me, being, get, the, meanwhile, yourselves, seem, amount, herself, besides, have, please, again, of, might, really, then, neither, there, has, you, seeming, none„m“, due, already, down, thereafter, were, will, own, beyond, nothing, be, few, more, former, since, call, those, for, thus, he, become, otherwise, an, empty, mostly, anywhere, wherein, indeed, whereafter, latterly, towards, show, five, than, whoever, although, move, least, once, go, moreover, third, themselves, toward, i, about, nine, sixty, from, ‘m, such, while, formerly, during, ’ve, beforehand, here, over, between, would, nobody, hers, nor, myself, re, out, via, our, one, thereby, they, every, „nt“, must, too, „re“, „s“, whenever, twelve, eleven, around, whatever, make, alone, after, other, whereby, do, a, his, who, else, whereas, why, ’ll, it, somehow, still, part, perhaps, sometimes, latter, throughout, them, very, quite, so, into, ‘re, two, thereupon, namely, under, twenty, never, beside, many, next, now, above, yet, along, bottom, to, am, yours, used, nevertheless, within, per, only, in, but, whereupon, among, others, ten, give, everything, full, first, at, therefore, because, where, that, upon, with, is, their, „ve“, however, further, thence, amongst, my, may, always, elsewhere, should, cannot, we, whence, did, whole, which, anyhow, six, until, unless, several, using, therein, name, not, ourselves, ‘s, most, across, „d“, fifteen, keep, your, „ll“, much, whether, had, various, himself, can, ’re, ours, something, ’s, four, regarding, anyway, made, she, hereafter, off, same, hereby, up, sometime, was, just, and, except, ever, top, together, if, either, on, take, though, enough, itself, fifty, becoming, or, somewhere, someone, doing, forty, less, back, wherever, are, before, by, three, hundred, these, no, against, side, how, thru

Anhang B

B.1 Keywordselektion

Keyword	**Similarity**
personal	0.6920167207717896
information	0.6812109351158142
invade_privacy	0.6740520596504211
info	0.6662514209747314
confidential_information	0.651781439781189
violate_privacy	0.6428519487380981
pii	0.6391447186470032
confidential	0.6363919377326965
permission	0.6222036480903625
identity	0.6206094622612
invasion_privacy	0.6175373792648315
privacy	0.6140189170837402
spy	0.6053483486175537
personally_identifiable	0.605155348777771

Tabelle B.1: Die 14 ähnlichsten Begriffe zum initialen Keyword: **personal_information**

Keyword	**Similarity**
privacy_policy	0.5983387231826782
agreement	0.561232328414917
disclaimer	0.5584768652915955
disclose	0.5497713088989258
tos	0.5299844741821289
eula	0.5096613168716431
term_agreement	0.48776379227638245
clause	0.46899354457855225
pii	0.4658811390399933
personal_information	0.4607158899307251
gdpr	0.4588368535041809
breach	0.4580553472042084
term_condition	0.4569736123085022
arbitration	0.4528177082538605

Tabelle B.2: Die 14 ähnlichsten Begriffe zum initialen Keyword: **disclosure**

Keyword	**Similarity**
confidential	0.6671245098114014
personal_information	0.651781439781189
private_confidential	0.5412140488624573
permission	0.5154746770858765
confidentiality	0.5078724026679993
consent	0.5047850012779236
personal	0.5022889375686646
pii	0.49581241607666016
personally_identifiable	0.48921507596969604
calender_event	0.48422184586524963
privacy_breach	0.4765622913837433
information	0.4653681218624115
gross_invasion	0.4644950032234192
camera_microphone	0.46381109952926636

Tabelle B.3: Die 14 ähnlichsten Begriffe zum initialen Keyword: **confidential_information**

Keyword	**Similarity**
personally_identifiable	0.6536592245101929
personal_information	0.6391447186470032
confidential	0.5226943492889404
hipaa	0.5141285061836243
identifiable_information	0.5140661597251892
divulge	0.5136538743972778
info	0.5099493265151978
breach	0.5021291375160217
confidential_information	0.4958125054836273
ssn	0.4944862127304077
personal	0.4848081171512604
steal_identity	0.48246172070503235
unfettered_access	0.4792608618736267
breeche	0.4776982367038727

Tabelle B.4: Die 14 ähnlichsten Begriffe zum initialen Keyword: **pii**

Keyword	**Similarity**
security_measure	0.7232373356819153
safety	0.7012678384780884
protection	0.6993065476417542
privacy	0.6957027316093445
secure	0.6608012914657593
secuirty	0.646793007850647
securty	0.6357026100158691
sequrity	0.6196225881576538
encryption	0.6195058822631836
privacy_protection	0.6132329106330872
protect	0.5947636365890503
sercurity	0.5852968692779541
secutity	0.5695582628250122
security_protocol	0.5626903176307678

Tabelle B.5: Die 14 ähnlichsten Begriffe zum initialen Keyword: **security**

Keyword	**Similarity**
privacy	0.6621456146240234
privacy_violation	0.6467101573944092
security_risk	0.6453513503074646
breach_privacy	0.6418044567108154
invasion_privacy	0.6364349722862244
privacy_breach	0.6347256898880005
concerned_privacy	0.6059290766716003
security_breach	0.599212646484375
respect_privacy	0.5941228270530701
privacy_invasion	0.5864307880401611
intrusion_privacy	0.5831705927848816
breach	0.5725035071372986
violation_privacy	0.5722947120666504
violate_privacy	0.5607143640518188

Tabelle B.6: Die 14 ähnlichsten Begriffe zum initialen Keyword: **privacy_concern**

Keyword	**Similarity**
privicy	0.7050511240959167
security	0.6957027316093445
respect_privacy	0.6753281354904175
confidentiality	0.674375593662262
protect_privacy	0.6708014011383057
privecy	0.6701926589012146
pricacy	0.6632412075996399
privacy_concern	0.6621456146240234
safety	0.6612159013748169
private	0.627245306968689
privacy_protection	0.6202348470687866
personal_information	0.6140189170837402
privacy_policy	0.6102060675621033
protection	0.6063618063926697

Tabelle B.7: Die 14 ähnlichsten Begriffe zum initialen Keyword: **privacy**

Anhang C

C.1 Docker Konfiguration

```
version: '3.1'
services:
  db:
    image: mariadb
    restart: always
    volumes:
      - db_data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: 12Qwer34
      MYSQL_DATABASE: knowledge_base
      MYSQL_USER: kb
      MYSQL_PASSWORD: 12Qwer34
    ports:
      - 3306:3306
  adminer:
    image: adminer
    restart: always
    ports:
      - 8080:8080
volumes:
  db_data: {}
```

Listing C.1: YAML [1]Konfigurationsdatei zum Erstellen der Docker Container *mariadb* und *adminer*.

[1]YAML ist eine vereinfachte Auszeichnungssprache zur Datenserialisierung.

Anhang D

D.1 Lernkurven

In diesem Anhang werden alle Lernkurven zu den verwendeten Deep Learning Algorithmen aufgeführt, dabei wurde eine Unterteilung nach verwendetem Trainingsdatenset vorgenommen. In Kapitel D.2 sind alle Daten zu den Modellen aufgeführt, welche mit dem Trainingsdatensatz *ganzer Bewertungen* trainiert wurden. In Kapitel D.3 sind alle Daten zu den Modellen aufgeführt, welche mit dem Trainingsdatensatz *einzelner Sätze* trainiert wurden.

Zu jeder Variante gemäß Tabelle 4.8 existieren je zwei Diagramme. Das linke Diagramm zeigt die Accuracy des Trainingssets (blau) und des Validierungssets (rot). Auf der Abszisse werden die Trainingsepochen dargestellt und die Ordinate zeigt die Accuracy im Bereich von 0.0 bis 1.0. Das rechte Diagramm zeigt den Loss auf dem Trainingsset (blau) und dem Validierungsset (rot). Die Werte der Abszisse und Ordinate bleiben unverändert. An den Abbildungsbeschreibungen kann die verwendete Variante ebenfalls abgelesen werden. Es gelten die Abkürzungen gem. Tabelle D.1.

Clf	Cl	Sp	Word Embedding	Abbildung Reviews / Sentences
FNN	F	F	ohne	D.1 / D.37
	T	F		D.2 / D.38
	F	T		D.3 / D.39
	T	T		D.4 / D.40
	F	F	word2vec - Ground Truth	D.5 / D.41
	T	F		D.6 / D.42
	F	T		D.7 / D.43
	T	T		D.8 / D.44
	F	F	GoogleNews-vectors-negative300	D.9 / D.45
	T	F		D.10 / D.46
	F	T		D.11 / D.47
	T	T		D.12 / D.48
CNN	F	F	ohne	D.13 / D.49
	T	F		D.14 / D.50
	F	T		D.15 / D.51
	T	T		D.16 / D.52
	F	F	word2vec - Ground Truth	D.17 / D.53
	T	F		D.18 / D.54
	F	T		D.19 / D.55
	T	T		D.20 / D.56
	F	F	GoogleNews-vectors-negative300	D.21 / D.57
	T	F		D.22 / D.58
	F	T		D.23 / D.59
	T	T		D.24 / D.60
RNN	F	F	ohne	D.25 / D.61
	T	F		D.26 / D.62
	F	T		D.27 / D.63
	T	T		D.28 / D.64
	F	F	word2vec - Ground Truth	D.29 / D.65
	T	F		D.30 / D.66
	F	T		D.31 / D.67
	T	T		D.32 / D.68
	F	F	GoogleNews-vectors-negative300	D.33 / D.69
	T	F		D.34 / D.70
	F	T		D.35 / D.71
	T	T		D.36 / D.72

Tabelle D.1: Übersicht zu allen Lernkurven aller Deep Learning Algorithmen mit Angabe der Abbildung. Aufteilung der Abbildungen nach Training auf Bewertungen oder Sätzen (siehe Spalte Abbildung Reviews / Sentences). *Abkürzungen: Clf: Classifier, Cl: Cleaning active, Sp: Rechtschreibkorrektur, T: True, F: False, EMB: genutztes Word Embedding*

D.2 Lernkurven - Bewertungen

In diesem Anhang Kapitel sind alle Lernkurven zu den Deep Learning Algorithmen aufgeführt, welche auf Grundlage von ganzen Bewertungen trainiert wurden.

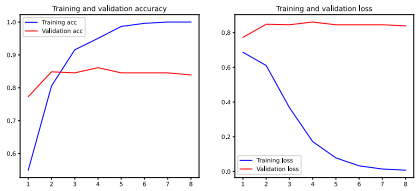


Abbildung D.1: FNN - Cl: False Sp: False EMB: no

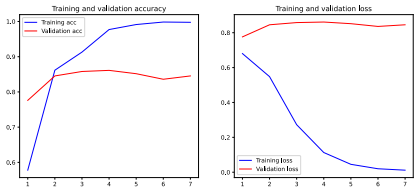


Abbildung D.2: FNN - Cl: True Sp: False EMB: no

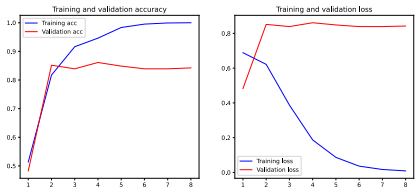


Abbildung D.3: FNN - Cl: False Sp: True EMB: no

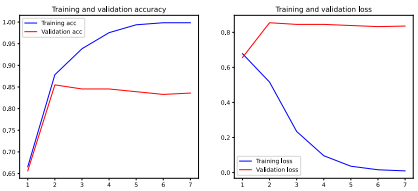


Abbildung D.4: FNN - Cl: True Sp: True EMB: no

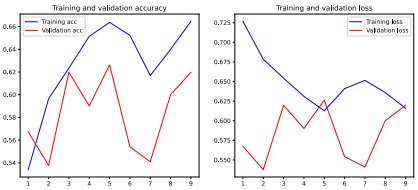


Abbildung D.5: FNN - Cl: False Sp: False EMB: Ground Truth

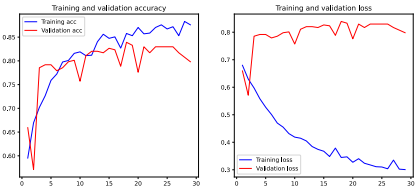


Abbildung D.6: FNN - Cl: True Sp: False EMB: Ground Truth

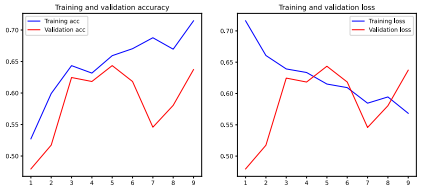


Abbildung D.7: FNN - Cl: False Sp: True EMB: Ground Truth

Abbildung D.8: FNN - Cl: True Sp: True EMB: Ground Truth

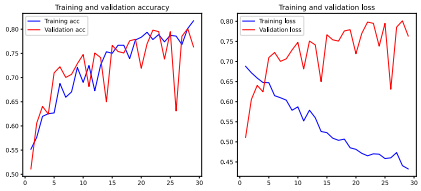


Abbildung D.9: FNN - Cl: False Sp: False EMB: Google

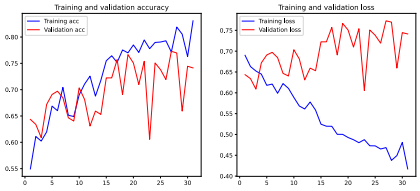


Abbildung D.10: FNN - Cl: False Sp: True EMB: Google

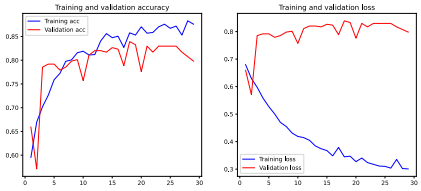


Abbildung D.11: FNN - Cl: True Sp: False EMB: Google

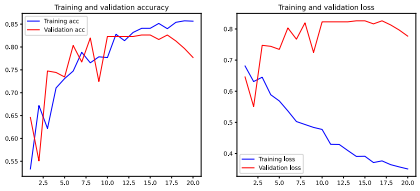


Abbildung D.12: FNN - Cl: True Sp: True EMB: Google

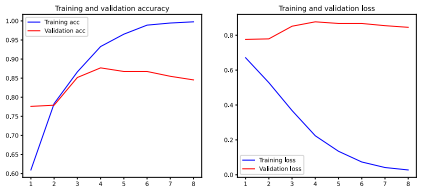


Abbildung D.13: CNN - Cl: False Sp: False EMB: no

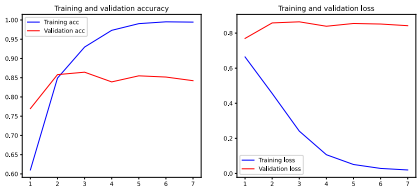


Abbildung D.14: CNN - Cl: True Sp: False EMB: no

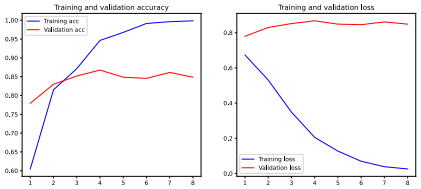


Abbildung D.15: CNN - Cl: False Sp: True EMB: no

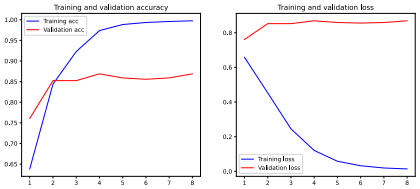


Abbildung D.16: CNN - Cl: True Sp: True EMB: no

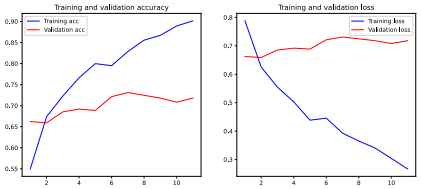


Abbildung D.17: CNN - Cl: False Sp: False EMB: Ground Truth

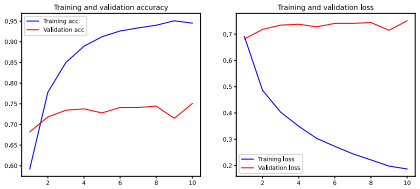


Abbildung D.18: CNN - Cl: True Sp: False EMB: Ground Truth

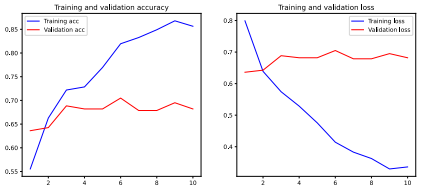


Abbildung D.19: CNN - Cl: False Sp: True EMB: Ground Truth

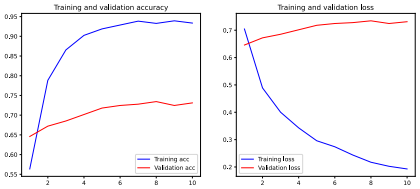


Abbildung D.20: CNN - Cl: True Sp: True EMB: Ground Truth

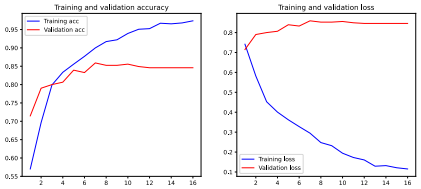


Abbildung D.21: CNN - Cl: False Sp: False EMB: Goolge

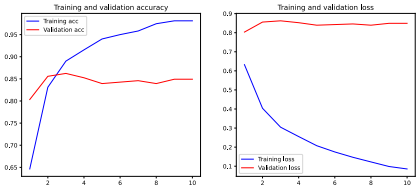


Abbildung D.22: CNN - Cl: True Sp: False EMB: Google

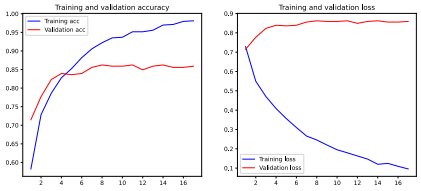


Abbildung D.23: CNN - Cl: False Sp: True EMB: Google

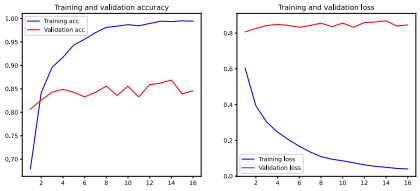


Abbildung D.24: CNN - Cl: True Sp: True EMB: Google

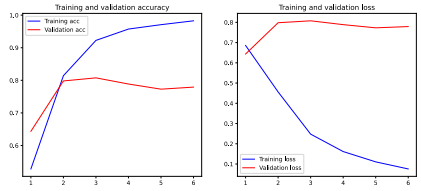


Abbildung D.25: RNN - Cl: False Sp: False EMB: no

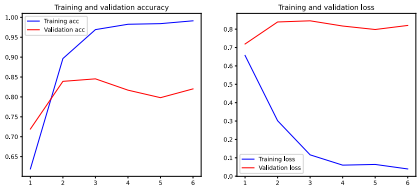


Abbildung D.26: RNN - Cl: True Sp: False EMB: no

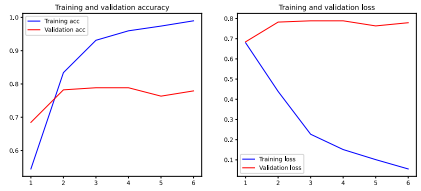


Abbildung D.27: RNN - Cl: False Sp: True EMB: no

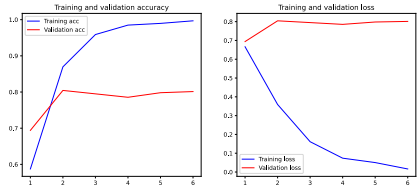


Abbildung D.28: RNN - Cl: True Sp: True EMB: no

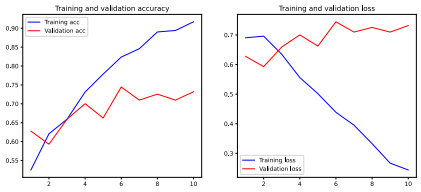


Abbildung D.29: RNN - Cl: False Sp: False EMB: Ground Truth

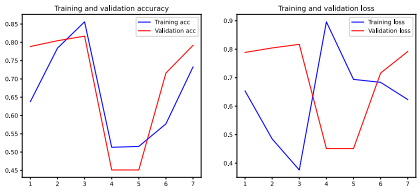


Abbildung D.30: RNN - Cl: True Sp: False EMB: Ground Truth

Abbildung D.31: RNN - Cl: False Sp: True EMB: Ground Truth

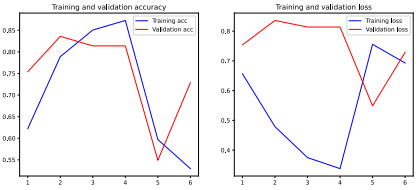


Abbildung D.32: RNN - Cl: True Sp: True EMB: Ground Truth

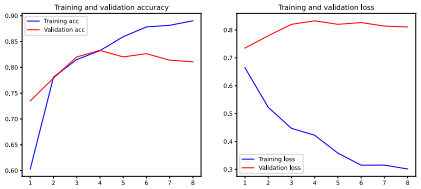


Abbildung D.33: RNN - Cl: False Sp: False EMB: Goolge

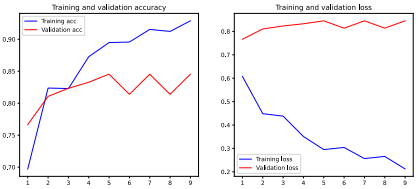


Abbildung D.34: RNN - Cl: True Sp: False EMB: Google

Abbildung D.35: RNN - Cl: False Sp: True EMB: Google

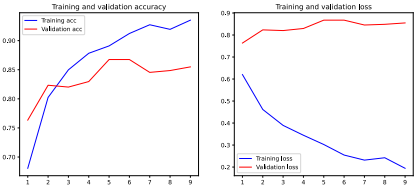


Abbildung D.36: RNN - Cl: True Sp: True EMB: Google

D.3 Lernkurven - Sätze

In diesem Anhang Kapitel sind alle Lernkurven zu den Deep Learning Algorithmen aufgeführt, welche auf Grundlage von einzelnen Sätzen trainiert wurden.

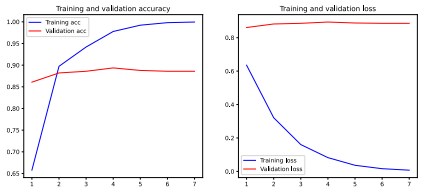


Abbildung D.37: FNN - Cl: False Sp: False EMB: no

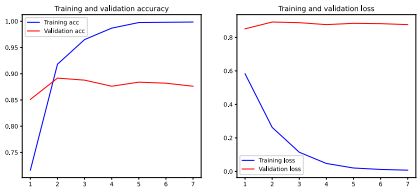


Abbildung D.38: FNN - Cl: True Sp: False EMB: no

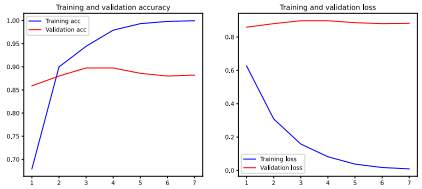


Abbildung D.39: FNN - Cl: False Sp: True EMB: no

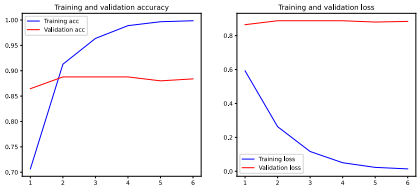


Abbildung D.40: FNN - Cl: True Sp: True EMB: no

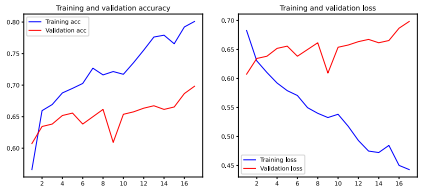


Abbildung D.41: FNN - Cl: False Sp: False EMB: Ground Truth

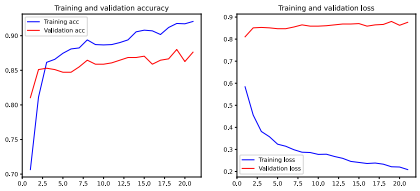


Abbildung D.42: FNN - Cl: True Sp: False EMB: Ground Truth

Abbildung D.43: FNN - Cl: False Sp: True EMB: Ground Truth

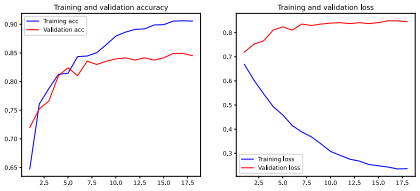


Abbildung D.44: FNN - Cl: True Sp: True EMB: Ground Truth

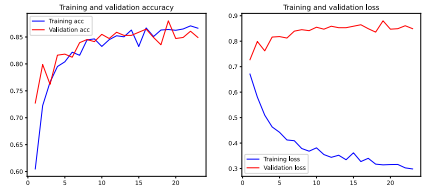


Abbildung D.45: FNN - Cl: False Sp: False EMB: Google

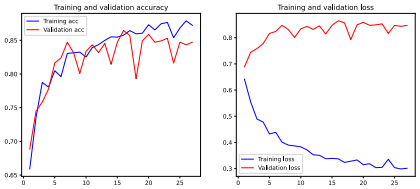


Abbildung D.46: FNN - Cl: False Sp: True EMB: Google

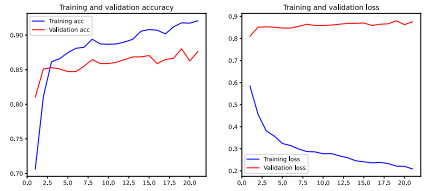


Abbildung D.47: FNN - Cl: True Sp: False EMB: Google

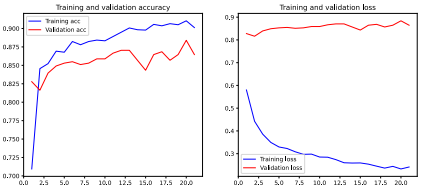


Abbildung D.48: FNN - Cl: True Sp: True EMB: Google

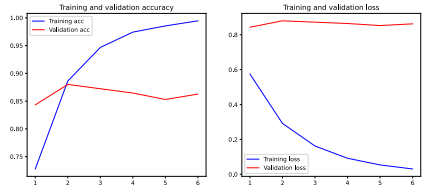


Abbildung D.49: CNN - Cl: False Sp: False EMB: no

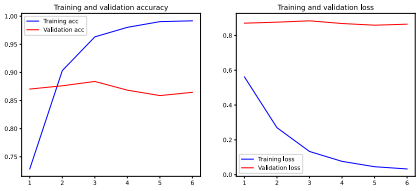


Abbildung D.50: CNN - Cl: True Sp: False EMB: no

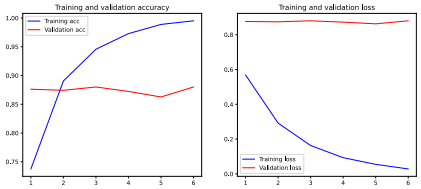


Abbildung D.51: CNN - Cl: False Sp: True EMB: no

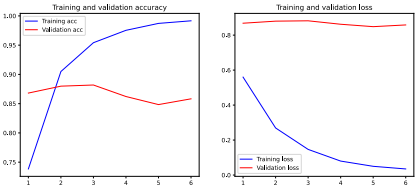


Abbildung D.52: CNN - Cl: True Sp: True EMB: no

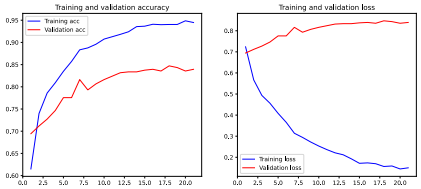


Abbildung D.53: CNN - Cl: False Sp: False EMB: Ground Truth

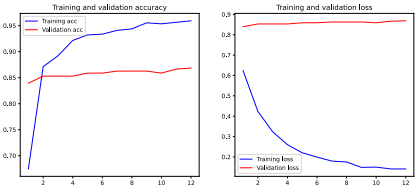


Abbildung D.54: CNN - Cl: True Sp: False EMB: Ground Truth

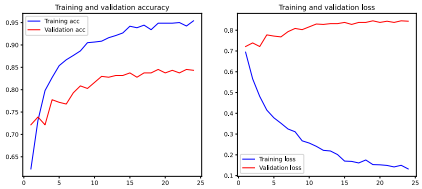


Abbildung D.55: CNN - Cl: False Sp: True EMB: Ground Truth

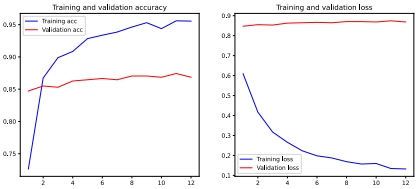


Abbildung D.56: CNN - Cl: True Sp: True EMB: Ground Truth

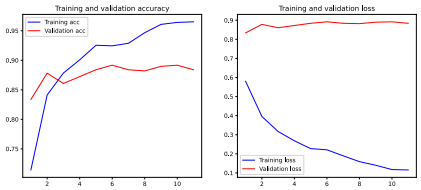


Abbildung D.57: CNN - Cl: False Sp: False EMB: Goolge

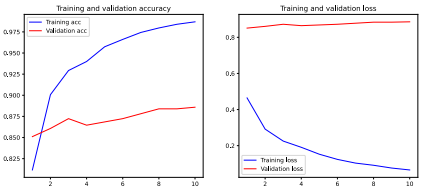


Abbildung D.58: CNN - Cl: True Sp: False EMB: Google

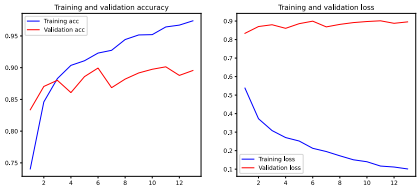


Abbildung D.59: CNN - Cl: False Sp: True EMB: Google

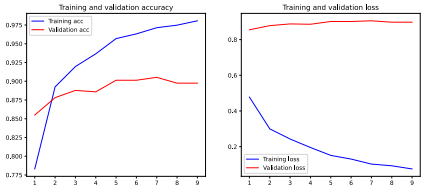


Abbildung D.60: CNN - Cl: True Sp: True EMB: Google

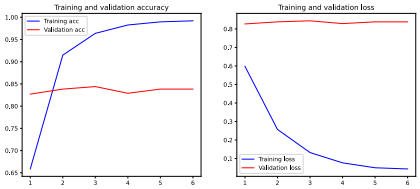


Abbildung D.61: RNN - Cl: False Sp: False EMB: no

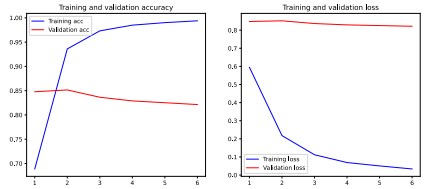


Abbildung D.62: RNN - Cl: True Sp: False EMB: no

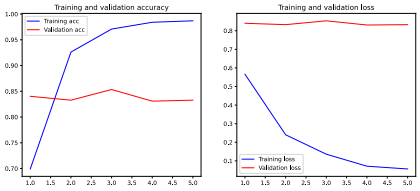


Abbildung D.63: RNN - Cl: False Sp: True EMB: no

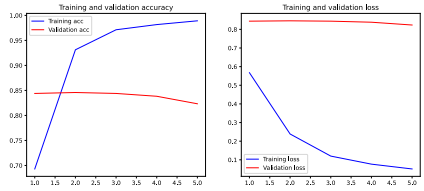


Abbildung D.64: RNN - Cl: True Sp: True EMB: no

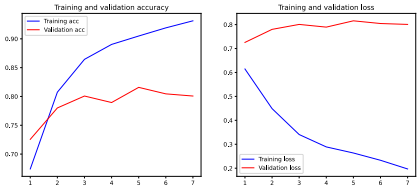


Abbildung D.65: RNN - Cl: False Sp: False EMB: Ground Truth

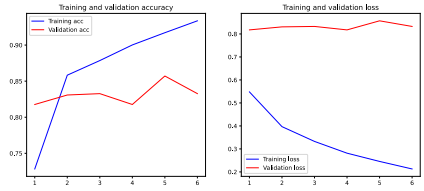


Abbildung D.66: RNN - Cl: True Sp: False EMB: Ground Truth

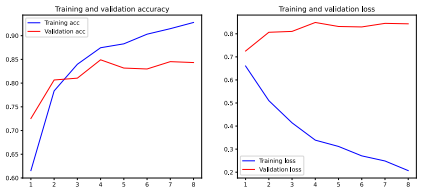


Abbildung D.67: RNN - Cl: False Sp: True EMB: Ground Truth

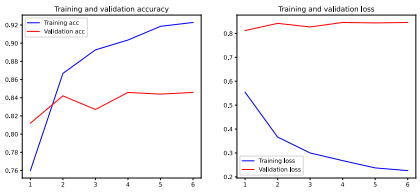


Abbildung D.68: RNN - Cl: True Sp: True EMB: Ground Truth

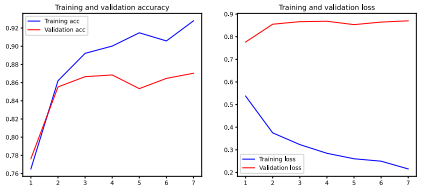


Abbildung D.69: RNN - Cl: False Sp: False EMB: Goolge

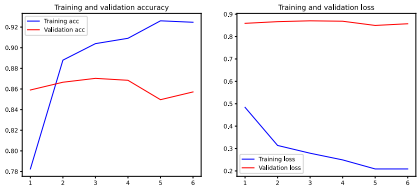


Abbildung D.70: RNN - Cl: True Sp: False EMB: Google

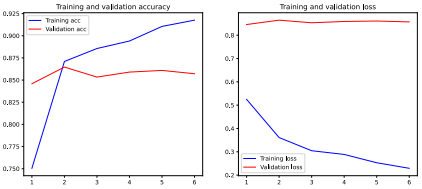


Abbildung D.71: RNN - Cl: False Sp: True EMB: Google

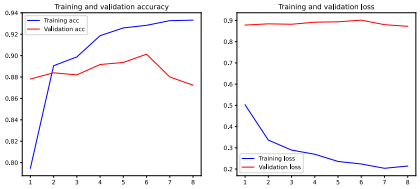


Abbildung D.72: RNN - Cl: True Sp: True EMB: Google

D.4 Tabellarische Ergebnisse

In Tabelle D.2 sind alle Performance-Werte aller Klassifikationsalgorithmen aufgelistet, die basierend auf ganzen Bewertungen trainiert wurden. In Tabelle D.3 sind alle Performance-Werte aller Klassifikationsalgorithmen aufgelistet, die basierend auf einzelnen Sätzen trainiert wurden. Die letzte Zeile der beiden Tabellen zeigt die Maximalwerte der jeweiligen Spalte an.

				Model				Reviews				Sentences			
Classifier	cleaning	spell_corrector	embeding_vector	Acc	Precision	Recall	F1	Acc	Precision	Recall	F1	Acc	Precision	Recall	F1
Ensemble	F	F	-	0.833	0.813	0.871	0.84	0.857	0.821	0.914	0.865	0.828	0.87	0.77	0.817
	T	F	-	0.852	0.861	0.845	0.852	0.914	1	0.828	0.906	0.877	0.96	0.787	0.865
	F	T	-	0.836	0.831	0.851	0.84	0.857	0.838	0.886	0.861	0.803	0.825	0.77	0.797
	T	T	-	0.837	0.845	0.83	0.837	0.871	0.861	0.886	0.873	0.861	0.907	0.803	0.852
Stacking-Clf	F	F	-	0.845	0.832	0.872	0.851	0.871	0.81	0.971	0.883	0.811	0.788	0.852	0.819
	T	F	-	0.868	0.858	0.888	0.872	0.886	0.846	0.943	0.895	0.82	0.791	0.869	0.828
	F	T	-	0.847	0.831	0.88	0.854	0.886	0.829	0.971	0.894	0.82	0.8	0.852	0.826
	T	T	-	0.867	0.86	0.885	0.871	0.857	0.791	0.971	0.872	0.803	0.761	0.886	0.819
FNN	F	F	ohne	0.847	0.851	0.848	0.849	0.9	0.868	0.943	0.904	0.861	0.867	0.852	0.86
	T	F		0.843	0.861	0.828	0.843	0.857	0.821	0.914	0.865	0.828	0.833	0.82	0.827
	F	T		0.861	0.866	0.859	0.862	0.9	0.85	0.971	0.907	0.86	0.87	0.852	0.86
	T	T		0.851	0.859	0.845	0.851	0.886	0.865	0.714	0.889	0.844	0.875	0.803	0.838
	F	F	word2vec - Ground Truth	0.624	0.642	0.612	0.617	0.443	0.639	0.657	0.648	0.688	0.717	0.623	0.667
	T	F		0.644	0.639	0.696	0.661	0.743	0.73	0.771	0.75	0.656	0.673	0.607	0.638
	F	T		0.596	0.611	0.574	0.589	0.671	0.667	0.686	0.676	0.705	0.745	0.623	0.679
	T	T		0.66	0.676	0.641	0.653	0.8	0.756	0.886	0.816	0.762	0.767	0.754	0.76
	F	F	GoogleNews-vectors-negative300	0.698	0.7	0.702	0.694	0.829	0.848	0.8	0.824	0.721	0.846	0.541	0.66
	T	F		0.747	0.752	0.765	0.753	0.9	0.912	0.886	0.899	0.811	0.865	0.735	0.796
	F	T		0.765	0.772	0.761	0.764	0.814	0.806	0.829	0.817	0.746	0.826	0.623	0.71
	T	T		0.778	0.78	0.786	0.782	0.914	0.854	1	0.921	0.828	0.812	0.852	0.832
CNN	F	F	ohne	0.861	0.866	0.859	0.862	0.871	0.882	0.857	0.87	0.869	0.925	0.803	0.86
	T	F		0.853	0.852	0.862	0.856	0.843	0.833	0.857	0.845	0.828	0.9	0.738	0.811
	F	T		0.854	0.865	0.849	0.855	0.83	0.53	0.829	0.841	0.861	0.893	0.82	0.855
	T	T		0.845	0.851	0.843	0.846	0.829	0.811	0.857	0.833	0.82	0.898	0.721	0.8
	F	F	word2vec - Ground Truth	0.665	0.675	0.675	0.67	0.814	0.789	0.857	0.822	0.713	0.681	0.803	0.737
	T	F		0.741	0.726	0.788	0.755	0.814	0.75	0.943	0.835	0.713	0.667	0.852	0.748
	F	T		0.677	0.685	0.68	0.68	0.829	0.871	0.771	0.818	0.738	0.716	0.787	0.75
	T	T		0.706	0.682	0.79	0.732	0.814	0.762	0.914	0.831	0.779	0.75	0.836	0.791
	F	F	GoogleNews-vectors-negative300	0.84	0.851	0.832	0.84	0.857	0.805	0.943	0.868	0.803	0.803	0.803	0.803
	T	F		0.834	0.844	0.827	0.834	0.871	0.825	0.943	0.88	0.746	0.75	0.738	0.744
	F	T		0.836	0.841	0.835	0.838	0.871	0.825	0.943	0.88	0.82	0.842	0.787	0.814
	T	T		0.851	0.856	0.851	0.852	0.9	0.833	1	0.909	0.787	0.797	0.77	0.783
RNN	F	F	ohne	0.794	0.824	0.749	0.778	0.743	0.73	0.771	0.75	0.73	0.78	0.639	0.703
	T	F		0.861	0.862	0.867	0.864	0.866	0.865	0.914	0.889	0.852	0.906	0.787	0.842
	F	T		0.775	0.828	0.696	0.748	0.857	0.791	0.971	0.872	0.836	0.789	0.918	0.848
	T	T		0.806	0.802	0.807	0.802	0.886	0.846	0.943	0.892	0.877	0.848	0.918	0.882
	F	F	word2vec - Ground Truth	0.718	0.731	0.69	0.7	0.757	0.75	0.771	0.761	0.746	0.747	0.705	0.735
	T	F		0.782	0.775	0.786	0.779	0.871	0.825	0.943	0.88	0.828	0.812	0.852	0.832
	F	T		0.725	0.769	0.642	0.689	0.829	0.78	0.914	0.842	0.77	0.77	0.77	0.77
	T	T		0.783	0.774	0.796	0.781	0.886	0.814	1	0.897	0.885	0.862	0.918	0.889
	F	F	GoogleNews-vectors-negative300	0.801	0.809	0.801	0.802	0.757	0.781	0.714	0.746	0.754	0.816	0.656	0.727
	T	F		0.803	0.807	0.789	0.796	0.929	0.917	0.943	0.93	0.811	0.839	0.77	0.803
	F	T		0.811	0.808	0.822	0.81	0.886	0.886	0.886	0.886	0.803	0.836	0.754	0.793
	T	T		0.815	0.805	0.832	0.815	0.929	0.917	0.943	0.93	0.811	0.896	0.705	0.789
			Maximalwerte	0.868	0.866	0.888	0.872	0.929	1	1	0.93	0.885	0.96	0.918	0.889

Tabelle D.2: Ergebnisse zu allen Klassifikationsalgorithmen, die basierend auf ganzen Bewertungen trainiert wurden.

				Model				Reviews				Sentences			
Classifier	cleaning	spell_corrector	embeding_vector	Acc	Precision	Recall	F1	Acc	Precision	Recall	F1	Acc	Precision	Recall	F1
Ensemble	F	F	-	0.864	0.86	0.867	0.863	0.843	0.8	0.914	0.853	0.828	0.823	0.836	0.829
	T	F	-	0.873	0.869	0.874	0.871	0.929	0.917	0.943	0.929	0.902	0.915	0.885	0.9
	F	T	-	0.862	0.866	0.853	0.86	0.914	0.872	0.971	0.919	0.869	0.869	0.869	0.869
	T	T	-	0.875	0.874	0.873	0.873	0.943	0.919	0.971	0.944	0.91	0.917	0.902	0.909
Stacking-Clf	F	F	-	0.867	0.846	0.895	0.87	0.843	0.8	0.914	0.853	0.852	0.841	0.869	0.855
	T	F	-	0.871	0.85	0.899	0.873	0.9	0.868	0.943	0.904	0.877	0.859	0.902	0.88
	F	T	-	0.867	0.85	0.886	0.868	0.871	0.81	0.971	0.88	0.861	0.843	0.885	0.864
	T	T	-	0.872	0.849	0.902	0.875	0.914	0.871	0.971	0.919	0.885	0.862	0.918	0.889
FNN	F	F	ohne	0.876	0.887	0.861	0.873	0.9	0.889	0.914	0.901	0.852	0.864	0.836	0.85
	T	F		0.87	0.88	0.855	0.867	0.914	0.892	0.943	0.917	0.877	0.897	0.852	0.874
	F	T		0.877	0.887	0.863	0.874	0.929	0.875	1	0.933	0.861	0.833	0.902	0.866
	T	T		0.877	0.897	0.849	0.872	0.943	0.897	1	0.946	0.918	0.918	0.918	0.918
	F	F	word2vec - Ground Truth	0.649	0.659	0.601	0.629	0.714	0.674	0.829	0.744	0.672	0.667	0.689	0.677
	T	F		0.721	0.766	0.639	0.689	0.914	0.872	0.971	0.919	0.861	0.844	0.885	0.864
	F	T		0.649	0.679	0.55	0.608	0.7	0.667	0.8	0.727	0.721	0.721	0.721	0.721
	T	T		0.759	0.772	0.73	0.75	0.914	0.914	0.914	0.914	0.803	0.785	0.836	0.81
	F	F	GoogleNews-vectors-negative300	0.817	0.848	0.769	0.806	0.843	0.786	0.943	0.857	0.861	0.867	0.852	0.86
	T	F		0.843	0.868	0.804	0.835	0.914	0.892	0.943	0.917	0.877	0.897	0.852	0.874
	F	T		0.815	0.831	0.783	0.806	0.857	0.791	0.971	0.872	0.844	0.828	0.869	0.848
	T	T		0.854	0.892	0.801	0.844	0.914	0.854	1	0.921	0.885	0.873	0.902	0.887
CNN	F	F	ohne	0.881	0.893	0.863	0.877	0.929	0.941	0.914	0.928	0.91	0.946	0.869	0.906
	T	F		0.877	0.895	0.854	0.873	0.929	0.917	0.943	0.93	0.902	0.915	0.885	0.9
	F	T		0.881	0.9	0.855	0.876	0.943	0.919	0.971	0.944	0.91	0.931	0.885	0.908
	T	T		0.882	0.898	0.863	0.878	0.957	0.944	0.971	0.958	0.926	0.948	0.902	0.924
	F	F	word2vec - Ground Truth	0.779	0.776	0.779	0.777	0.886	0.829	0.971	0.895	0.852	0.841	0.869	0.855
	T	F		0.84	0.836	0.844	0.839	0.914	0.872	0.971	0.919	0.902	0.877	0.934	0.905
	F	T		0.779	0.782	0.772	0.776	0.9	0.883	1	0.909	0.811	0.788	0.852	0.819
	T	T		0.844	0.843	0.841	0.842	0.914	0.872	0.971	0.919	0.877	0.848	0.918	0.882
	F	F	GoogleNews-vectors-negative300	0.871	0.891	0.841	0.865	0.871	0.825	0.943	0.88	0.877	0.883	0.869	0.876
	T	F		0.87	0.887	0.846	0.865	0.886	0.865	0.914	0.889	0.869	0.857	0.885	0.871
	F	T		0.872	0.896	0.839	0.866	0.943	0.919	0.971	0.944	0.885	0.873	0.902	0.887
	T	T		0.887	0.899	0.861	0.876	0.914	0.892	0.943	0.917	0.869	0.846	0.902	0.873
RNN	F	F	ohne	0.856	0.861	0.855	0.856	0.843	0.833	0.857	0.845	0.885	0.855	0.855	0.855
	T	F		0.971	0.876	0.865	0.869	0.871	0.825	0.943	0.88	0.91	0.903	0.918	0.911
	F	T		0.863	0.883	0.842	0.859	0.9	0.833	1	0.909	0.91	0.903	0.918	0.911
	T	T		0.869	0.887	0.845	0.864	0.914	0.872	0.971	0.919	0.893	0.875	0.918	0.896
	F	F	word2vec - Ground Truth	0.817	0.812	0.833	0.819	0.843	0.786	0.943	0.857	0.811	0.797	0.836	0.816
	T	F		0.847	0.845	0.853	0.848	0.857	0.805	0.943	0.868	0.893	0.9	0.885	0.893
	F	T		0.805	0.796	0.822	0.807	0.829	0.745	1	0.854	0.836	0.806	0.885	0.844
	T	T		0.837	0.835	0.84	0.837	0.943	0.897	1	0.946	0.902	0.866	0.951	0.906
	F	F	GoogleNews-vectors-negative300	0.872	0.863	0.888	0.873	0.857	0.821	0.914	0.865	0.885	0.898	0.869	0.883
	T	F		0.873	0.866	0.877	0.874	0.886	0.865	0.914	0.889	0.918	0.905	0.934	0.919
	F	T		0.087	0.868	0.873	0.8	0.857	0.791	0.971	0.872	0.91	0.917	0.902	0.909
	T	T		0.867	0.868	0.869	0.867	0.857	0.791	0.971	0.895	0.893	0.875	0.918	0.896
			Maximalwerte	0.971	0.9	0.902	0.878	0.957	0.944	1	0.958	0.926	0.948	0.951	0.924

Tabelle D.3: Ergebnisse zu allen Klassifikationsalgorithmen, die basierend auf einzelnen Sätzen trainiert wurden.

Abbildungsverzeichnis

Tabellenverzeichnis

Akronyme

API Application Programming Interface 89

CBOW Continuous Bag of Words 23, 26, 27, 47

CNN Convolutional Neural Network 18, 19, 20, 38, 54, 58, 59, 70, 76, 87, 92

CSV Comma-separated values 83

FN False Negatives 15, 52

FNN Feedforward Neural Network 18, 38, 58, 70

FP False Positives 15, 52

GUI Graphical User Interface 42

JSON JavaScript Object Notation 33, 43, 52, 83

LSTM Long short-term memory 21, 59

ML Machine Learning 31, 32, 35, 37, 39, 41, 46, 47, 53, 55, 60, 67, 70, 79, 82, 87, 88, 89, 93

NLP Natural Language Processing 22, 25, 39, 41, 54, 79, 86, 87, 88

ReLU Rectified Linear Unit 20, 58, 59

RNN Recurrent Neural Network 20, 21, 38, 54, 58, 59, 70

SHA Secure Hash Algorithm 60

SQL Structured Query Language 38, 92

SVC Support Vector Classification 53, 55, 57

SVM Support Vector Machine 11

TF-IDF Term Frequency-Inverse Document Frequency 28, 29, 56

TN True Negatives 15, 52

TP True Positives 15, 52

VC Voting Classifier 13

VPN Virtual Private Network 46, 81

Literaturverzeichnis

[1] AppBrain - Statista Inc. Number of Android apps on Google Play. https://www.appbrain.com/stats/number-of-android-apps, 2021. Zugriff: 30.04.2021.

[2] AppBrain - Statista Inc. Number of available applications in the Google Play Store from December 2009 to December 2020. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, 2021. Zugriff: 30.04.2021.

[3] R. S. Ashtankar and W. M. Choudhari. A natural language processing based approach using stochastic petri nets for understanding software requirement specifications. *International Journal of Computer Sciences and Engineering*, page 107, 2016.

[4] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012.

[5] F. Ceballos. Stacking classifiers for higher predictive performance. https://towardsdatascience.com/stacking-classifiers-for-higher-predictive-performance-566f963e4840, 2019. Zugriff: 28.04.2021.

[6] P. Chandrasekar and K. Qian. The impact of data preprocessing on the performance of a naive bayes classifier. In *40th IEEE Annual Computer Software and Applications Conference, COMPSAC Workshops 2016, Atlanta, GA, USA, June 10-14, 2016*, pages 618–619. IEEE Computer Society, 2016.

[7] J. Cohen. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, 2011.

[9] A. Corral and I. Serra. The brevity law as a scaling law, and a possible origin of zipf's law for word frequencies. *Entropy*, 22(2):224, 2020.

[10] ExplosionAI GmbH. Industrial-Strength Natural Language Processing in Python. `https://spacy.io/`, 2021. Zugriff: 29.03.2021.

[11] A. Faul. *A concise introduction to machine learning, A Chapman & Hall book, Chapman & Hall/CRC machine learning & pattern recognition*. CRC Press, Boca Raton, 2020.

[12] E. Haddi, X. Liu, and Y. Shi. The role of text pre-processing in sentiment analysis. In *Proceedings of the First International Conference on Information Technology and Quantitative Management, ITQM 2013, Dushu Lake Hotel, Sushou, China, 16-18 May, 2013*, volume 17 of *Procedia Computer Science*, pages 26–32. Elsevier, 2013.

[13] H. Harkous, K. Fawaz, R. Lebret, F. Schaub, K. G. Shin, and K. Aberer. Polisis: Automated analysis and presentation of privacy policies using deep learning. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 531–548. USENIX Association, 2018.

[14] M. Hatamian, J. Serna, and K. Rannenberg. Revealing the unrevealed: Mining smartphone users privacy perception on app markets. *Computers and Security*, 83:332–353, 2019.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[16] B. L. Jacobsen. *Entwicklung eines Tools zur Feedback-Visualisierung*. Bachelor´s thesis, Leibniz Universität Hannover, 2021.

[17] M. Janani and S. Sumathi. *Neural networks for natural language processing, Advances in computer and electrical engineering ACEE book series*. IGI Global, Hershey, Pennsylvania, 2020.

[18] U. Kamath, J. Liu, and J. Whitaker. *Deep Learning for NLP and Speech Recognition*. Springer, 2019.

[19] S. Kannan, V. Gurusamy, S. Vijayarani, J. Ilamathi, and M. Nithya. Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2014.

[20] J. Kim. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Comput. Stat. Data Anal.*, 53(11):3735–3745, 2009.

[21] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[22] M. Kuhn and K. Johnson. *Applied predictive modeling*, volume 26. New York: Springer, 2013.

[23] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[24] M. M. Lopez and J. Kalita. Deep learning applied to NLP. *CoRR*, abs/1703.03091, 2017.

[25] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empir. Softw. Eng.*, 21(3):1067–1106, 2016.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

[28] A. M. Molinaro, R. Simon, and R. M. Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinform.*, 21(15):3301–3307, 2005.

[29] C. Olah. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. Zugriff: 10.05.2021.

[30] Oxford University Press. Oxford advanced learner's dictionary of current English. `https://www.oxfordlearnersdictionaries.com/definition/english/privacy?q=privacy`, 2021. Zugriff: 24.03.2021.

[31] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *21st IEEE International Requirements Engineering Conference, RE 2013, Rio de Janeiro-RJ, Brazil, July 15-19, 2013*, pages 125–134. IEEE Computer Society, 2013.

[32] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. How can i improve my app? classifying user reviews for software maintenance and evolution. In *2015 IEEE International Conference on Software Maintenance and Evolution, ICSME 2015, Bremen, Germany, September 29 - October 1, 2015*, pages 281–290. IEEE Computer Society, 2015.

[33] Peter Norvig. Peter Norvig - How to Write a Spelling Corrector. `http://norvig.com/spell-correct.html`, 2016. Zugriff: 05.05.2021.

[34] M. F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[35] Y. Qi, D. S. Sachan, M. Felix, S. J. Padmanabhan, and G. Neubig. When and why are pre-trained word embeddings useful for neural machine translation? *CoRR*, abs/1804.06323, 2018.

[36] J. Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

[37] K. Renaud and D. Galvez-Cruz. Privacy: Aspects, definitions and a multi-faceted privacy preservation approach. In *Information Security South Africa Conference 2010, Sandton Convention Centre, Sandton, South Africa, August 2-4, 2010. Proceedings ISSA 2010*. ISSA, Pretoria, South Africa, 2010.

[38] J. W. Richards, H. Brink, M. Fetherolf, and B. Cronin. *Real-world machine learning*. Manning, Shelter Island, 2017.

[39] S. Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *J. Documentation*, 60(5):503–520, 2004.

[40] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[41] C. Sangani and S. Ananthanarayanan. Sentiment analysis of app store reviews. *Methodology*, 4(1):153–162, 2013.

[42] P. Silva, C. Gonçalves, C. Godinho, N. Antunes, and M. Curado. Using natural language processing to detect privacy violations in online contracts. In *SAC '20: The 35th ACM/SIGAPP Symposium on Applied Computing, online event, [Brno, Czech Republic], March 30 - April 3, 2020*, pages 1305–1307. ACM, 2020.

[43] D. K. Van Bogaert and G. Ogunbanjo. Confidentiality and privacy: What is the difference? *South African Family Practice*, 51(3):194–195, 2009.

[44] S. Visa and A. Ralescu. Issues in mining imbalanced data sets-a review paper. *Proc. 16th Midwest Artificial Intelligence and Cognitive Science Conference*, 2005:67–73, 2005.

[45] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen. Mining user opinions in mobile app reviews: A keyword-based approach (T). In *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*, pages 749–759. IEEE Computer Society, 2015.

[46] Y. Yamashita. Convolutional neural networks - basics - an introduction to cnns and deep learning. `https://mlnotebook.github.io/post/CNN1/`, 2017. Zugriff: 13.05.2021.

[47] C. Zhang, W. Fan, N. Du, and P. S. Yu. Mining user intentions from medical queries: A neural network based heterogeneous jointly modeling approach. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 1373–1384. ACM, 2016.