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

Entwicklung einer Software zur Extrahierung und Analyse von Reviews aus App Stores

Development of Software for Extracting and Analyzing Reviews from App Stores

Bachelorarbeit

im Studiengang Informatik

von

Timo Kurtz

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Dr. Jil Klünder
Betreuer: M. Sc. Martin Obaidi

Hannover, 28.08.2023

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig 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 28.08.2023

Timo Kurtz

Zusammenfassung

Die Analyse von App-Reviews ist ein bedeutendes Forschungsgebiet. Aufgrund der großen Menge an Reviews, die für eine beliebte App im Google Play Store oder Apple App Store verfasst werden, gelangen App-Analysten bei einer manuellen Auswertung häufig an ihre Kapazitätsgrenzen. Daher besteht ein wachsender Bedarf an automatisierten Methoden zur Analyse dieser Reviews. Zur Unterstützung von App-Analysten stellt diese Arbeit die neu implementierte Anwendung *feelio* vor. Sie verfügt über eine benutzerfreundliche Oberfläche, die es dem Nutzer ermöglicht, englischsprachige Reviews einer App aus den beiden führenden App Stores, dem Google Play Store und dem Apple App Store zu erfassen oder mittels CSV-Datei zu importieren. Jedes Review erhält eine Zuordnung hinsichtlich verschiedener Merkmale: eine Sentimentpolarität, eine Emotion gemäß den Ekman'schen Basisemotionen, eine sprachliche Qualität, ein potenzieller Erklärungsbedarf sowie spezifische Charakteristika, wie beispielsweise die Klassifizierung als Funktionsanforderung („feature request"). Für die Sentimentanalyse wurde speziell der Transfer-Learning-Ansatz angewendet. Dabei kam ein bestehendes Large Language Model (LLM) zum Einsatz, das mithilfe von eigens annotierten Reviews auf die Domäne der App Stores im Rahmen dieser Arbeit trainiert wurde. Gemeinsam tragen alle Klassifizierungsmerkmale zur umfassenden Analyse von App-Reviews bei und bieten dem Nutzer Zugang zu einer Vielzahl von Review-Statistiken, einschließlich Filteroptionen, die speziell für die ausgewählte App bereitstehen. Zusätzlich kann ein Vergleich der App-Reviews und deren Klassifizierungen vorgenommen werden, um zu ermitteln, inwiefern sich die Statistiken der Reviews zwischen den beiden App Stores unterscheiden.

Abstract

Development of Software for Extracting and Analyzing Reviews from App Stores

The analysis of app reviews represents a significant field of research. Due to the large volume of reviews generated for a popular app on platforms such as the Google Play Store or Apple App Store, manual evaluation often pushes app analysts to their capacity limits. Therefore, there is a growing need for automated methods to analyze these reviews. To assist app analysts, this paper introduces the newly implemented application named *feelio*. It features a user-friendly interface that not only enables users to extract reviews from both the Google Play Store and Apple App Store but also allows importing reviews via a CSV file. Each review is categorized based on various attributes: sentiment polarity, emotion according to Ekman's basic emotions, linguistic quality, potential need for explanation and specific characteristics like being identified as a „feature request". For sentiment analysis, the transfer learning approach was specifically applied. An existing Large Language Model (LLM) was utilized, which was trained specifically on app reviews that were annotated for this study. Collectively, all classification features allow an in-depth analysis of app reviews, granting users access to a wide range of review statistics, including customized filtering options for a chosen app. Moreover, users can compare reviews and their classifications to identify disparities in review trends between the two major app stores.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Distribution von Applikationen, ob auf dem Smartphone oder der Smartwatch erfolgt zum großen Teil über die zwei Plattformen Google Play Store[1] und Apple App Store[2] [2]. Der Markt für mobile Applikationen hat sich zu einem rentablen Markt entwickelt, bei dem sich der Markteinstieg besonders für neue Entwickler leicht darstellt und eine schnelle finanzielle Rentabilität verspricht [6]. Dabei verzeichnet allein der Apple App Store im Quartal 2022 einen Umsatz von 21.2 Milliarden US-Dollar für den Vertrieb seiner Apps [82]. Um dem Nutzer des App Stores möglichst hochwertige Apps anzubieten, ist die Klassifizierung der Apps mithilfe von Reviews eine Möglichkeit, hervorragende Apps von schlechten Apps sofort zu unterscheiden. Für die Entwickler stellt es hingegen eine Möglichkeit dar, auf Verbesserungsvorschläge, Fehlerberichte und die allgemeine Unzufriedenheit von App-Nutzern, die über die Reviews den Entwicklern übermittelt werden, einzugehen [22, 26, 64, 66]. Die gewonnenen Erkenntnisse lassen sich effektiv in den Prozess der Anforderungsanalyse einbeziehen, was dazu beiträgt, die App-Entwicklung weiter zu optimieren [50, 80]. Darüber hinaus ist für die Unternehmen eine gute Gesamtbewertung der App wichtig, um eine professionelle Präsenz und ein gutes Image des Unternehmens im Zeitalter des Internets aufrechtzuerhalten [49]. Dabei spielt der Kontakt zum Kunden (Nutzer der App) und das Beantworten der Reviews im App Store eine wichtige Rolle.

1.1 Problemstellung

Besonders gefragte Apps, die aufgrund ihrer hohen Downloadzahlen die Ranglisten der App Stores dominieren, erhalten eine sehr große Anzahl an täglichen Reviews [28]. Eine manuelle Auswertung der Reviews auf täglicher Basis würde viel Zeit und Kapazitäten der Unternehmen in Anspruch

[1] `https://play.google.com/store/games?hl=en&gl=US`
[2] `https://www.apple.com/app-store/`

nehmen. Es stellt sich die Frage, wie die Auswertung der Reviews effizienter und automatisierter erfolgen kann.

1.2 Lösungsansatz

Im Rahmen dieser Arbeit wurde eine Anwendung zur Review-Analyse in englischer Sprache entwickelt. Sie hilft App-Analysten, Reviews aus den zwei führenden App Stores, dem Apple App Store und dem Google Play Store, zu erfassen oder mittels einer CSV-Datei zu importieren. Nach dem Import werden die Reviews hinsichtlich ihres Sentiments, ihrer Emotionen, ihres Sprachstils, spezifischer Merkmale und des Erklärungsbedarfs klassifiziert. Eine intuitive Benutzeroberfläche präsentiert Statistiken zu den klassifizierten Reviews und bietet Analysemöglichkeiten mithilfe verschiedener Filteroptionen. Auch ein Vergleich der Reviews zwischen den beiden App Stores sowie mit potenziellen Reviews aus einer CSV-Datei ist möglich.

Speziell für die Sentimentanalyse wurden 1080 Reviews annotiert, um ein Modell zu trainieren, das speziell auf die Domäne der App-Reviews zugeschnitten ist. Unabhängig von der tatsächlichen Software-Implementierung nahm die Auswertung der Performance der Klassifizierer einen wichtigen Teil der Arbeit ein, um zu bewerten, inwieweit eine solche Klassifizierung auch in der Praxis ihre Anwendung finden kann.

1.3 Struktur der Arbeit

Zu Beginn vermittelt das Kapitel 2 die wesentlichen Grundlagen zum Verständnis der nachfolgenden Kapitel. Das Kapitel 3 befasst sich mit den grundlegenden Konzepten der Arbeit. Anschließend nimmt das Kapitel 4 Bezug zur Implementierung der Anwendung. Kapitel 5 diskutiert die hervorgebrachten Ergebnisse in Hinsicht der Performance der KI-Methoden und die Einordnung der Grenzen und Gefahren der Validität. Bezug zu verwandten wissenschaftlichen Arbeiten wird in Kapitel 6 vermittelt. Auf Grundlage der behandelten Nachforschungen in diesem Kapitel wird die Einzigartigkeit dieser Arbeit hervorgehoben. Zu guter Letzt bietet das Kapitel 7 eine Übersicht über die behandelten Themen in dieser Arbeit und gibt eine Perspektive auf mögliche zukünftige Verbesserungen.

Kapitel 2

Grundlagen

Das folgende Kapitel vermittelt die Grundlagen für ein besseres Verständnis der zugrunde liegenden Arbeit. Zunächst wird in Abschnitt 2.1 auf das Requirements Engineering eingegangen, welches für ein erfolgreiches Softwareprojekt unverzichtbar ist. Anschließend wird im nächsten Abschnitt 2.2 der Aspekt der Usability einer Software behandelt. Eine Einführung in das Themengebiet künstliche Intelligenz folgt in den Abschnitten 2.3 und 2.4. Im Anschluss daran beschäftigt sich der Abschnitt 2.5 mit dem Gebiet des Natural Language Processing. Schließlich stellt der letzte Abschnitt 2.6 verschiedene Metriken vor, die für die spätere Auswertung in dieser Arbeit erforderlich sind.

2.1 Requirements Engineering

In der Softwareentwicklung existieren insbesondere zwei unabhängige Parteien: die Entwickler, welche die Software erschaffen, und die Nutzer, die diese Software letztendlich verwenden möchten [68]. Das Entwicklungsteam strebt danach, dem Endbenutzer ein möglichst ausgereiftes und fertiges Produkt zu präsentieren. Es ergibt sich jedoch eine komplexe Herausforderung, da Softwareentwickler und Kunden bzw. Endbenutzer unterschiedliche Fachgebiete und Perspektiven haben. Während die Entwickler die Fachkenntnisse zur Erstellung der Software besitzen und davon ausgehen, die Kundenanforderungen richtig zu verstehen, hat der Kunde das spezifische Verständnis für sein Geschäft und weiß, was die Software leisten soll. Allerdings fehlt ihm oft das Wissen, was die Software im Allgemeinen leisten kann. Diese gegenseitige Unkenntnis des Wissensstandes des anderen führt zu einem Unverständnis auf beiden Seiten, das auch als „Symmetry of Ignorance“ bezeichnet wird [21]. Ein Ansatz zur Lösung dieses Problems ist die Anwendung des Prinzips des Requirements Engineering (RE) [36]. RE beruht auf einem systematischen Vorgehen, um die Anforderungen des Kunden zu erheben und kontinuierlich zu verfolgen. Die Entwicklung

beginnt mit der Erhebung der Rohanforderungen, die anschließend interpretiert, priorisiert und konkretisiert werden müssen [8]. Während dieser ersten Phase können widersprüchliche Anforderungen auftreten, die es zu identifizieren gilt. Um eventuelle Inkonsistenzen zu beseitigen, müssen diese identifizierten Widersprüche besprochen und verhandelt werden. In diesem Prozess werden die Anforderungen in funktionale und nicht-funktionale Kategorien unterteilt. Die funktionalen Anforderungen beschreiben dabei die Kernfunktionen der Software [8, 36]. Die nicht-funktionalen Anforderungen hingegen beziehen sich auf die Art und Weise, wie die Software diese Kernfunktionen umsetzen soll. Innerhalb des Kontextes dieser Arbeit bestehen die funktionalen Anforderungen darin, Reviews zu erfassen (crawlen) und verschiedene Aspekte hinsichtlich der Analyse der Reviews zu berücksichtigen. Die Benutzerfreundlichkeit (Usability) und die Modularität der Software zählen zu den Qualitätsanforderungen und sind somit Teil der nicht-funktionalen Anforderungen.

2.2 Usability

Ein User Interface sollte nicht nur funktional zuverlässig sein, sondern auch eine benutzerfreundliche Bedienung gewährleisten. Zur Verbesserung der Usability kann auf eine der bekanntesten Methoden zur Optimierung von User Interfaces zurückgegriffen werden: die „10 Usability Heuristics for User Interface Design“ nach Jakob Nielsen [56]. Diese Heuristiken, die seit 1990 Bestand haben, wurden im Jahr 2020 lediglich im Wortlaut angepasst und optimiert [55].

1. **Visibility of system status**: Dem Benutzer der Software sollte zu jeder Zeit klar sein, in welchem Zustand sich die Software befindet, beispielsweise bei längeren Ladezeiten, die durch eine Fortschrittsanzeige dargestellt werden.

2. **Match between system and the real world**: Die Software spricht die Sprache des Anwenders und vermeidet den Gebrauch von domänenspezifischen Begriffen und Konventionen. Der Aufbau und die Funktionsweise müssen dem Anwender logisch erscheinen, wie beispielsweise das Verwenden eines „Burger“-Icons für ein eingeklapptes Menü.

3. **User control and freedom**: Es ist wichtig für eine Software, dass Benutzer ihre Aktionen widerrufen können. Nur so wird gewährleistet, dass Benutzer neue Funktionen testen und verwenden möchten, ohne sich vorher Sorgen machen zu müssen, dass ihr Systemstatus womöglich verloren gehen könnte (konsistenter Zustand bei Verwendung des Zurück-Buttons).

4. **Consistency and standards**: Die Software sollte eine einheitliche Struktur aufweisen, etwa durch die Verwendung eines konsistenten Design-Patterns. Dies trägt dazu bei, dass der Benutzer nicht verwirrt wird und Missverständnisse vermieden werden. Zudem sollten allgemein vertraute Standards befolgt werden, die dem Benutzer bereits bekannt sind, wie etwa die Platzierung einer Navigationsleiste im oberen Bereich einer Webseite.

5. **Error prevention**: Es sollte sichergestellt werden, dass der Benutzer keine unbeabsichtigten Aktionen ausführen kann.

6. **Recognition rather than recall**: Innerhalb der Benutzeroberfläche sollten Hinweise angeboten werden, sodass der Benutzer Funktionen eher erkennt, als sich an sie erinnern zu müssen.

7. **Flexibility and efficiency of use**: Die Software sollte in einer Weise entwickelt werden, dass sie sowohl von Experten als auch von Anfängern effizient genutzt werden kann. Dabei könnten Kurzbefehle für erfahrene Nutzer integriert werden, die jedoch für durchschnittliche oder unerfahrene Benutzer nicht zwingend erforderlich sind.

8. **Aesthetic and minimalist design**: Um den Benutzer nicht zu überfordern, sollte das User Interface einer Software ansprechend und minimalistisch gestaltet sein. Hierbei sollte besonderer Wert darauf gelegt werden, dass das Design nicht von der Hauptfunktion der Software ablenkt.

9. **Help users recognize, diagnose, and recover from errors**: Fehler sollten dem Benutzer deutlich angezeigt werden. Dabei sollte klar kommuniziert werden, welches Problem vorliegt und wie es auf einfachste Weise behoben werden kann.

10. **Help and documentation**: Während der Benutzung einer Software können kleine Hilfestellungen für den Benutzer äußerst nützlich sein. Beispielsweise könnte ein Tooltip erscheinen, wenn der Benutzer den Cursor über ein Element bewegt, um einen Hinweis auf dessen Funktion zu geben.

Angesichts der ständig fortschreitenden Technologien bedarf es einer kontinuierlichen Anpassung und Weiterentwicklung der Heuristiken [23]. Andernfalls könnten selbst Nielsens Heuristiken [55] bald an Relevanz verlieren [23]. Eine stärkere empirische Basis ist nur einer von vielen Aspekten, die es zu optimieren gilt, um Heuristiken zeitgemäß zu halten. Aktuell sind Nielsens Heuristiken [55] jedoch gut an die gegenwärtige Technik angepasst [23].

2.3 Lernverfahren in der künstlichen Intelligenz

Die künstliche Intelligenz (KI) gilt als eines der aufstrebenden Forschungsgebiete der Informatik in jüngster Zeit [8]. Um zu bewerten, inwieweit eine KI der menschlichen Intelligenz ähnelt oder diese sogar übertreffen kann, wird allgemein der Grad einer KI in drei Kategorien unterteilt: Narrow AI (NAI), General AI (GAI) und Super AI (SAI) [5]. NAI bezeichnet eine eingeschränkte KI, die nur für ein bestimmtes Anwendungsgebiet ausgelegt ist. GAI repräsentiert die allgemeine KI, die zu jedem Thema eine Antwort finden kann. SAI kann sogar der menschlichen Intelligenz überlegen sein. Derzeit befindet sich die industrielle Anwendung von KI hauptsächlich im Bereich der NAI, da Projekte wie DeepMind [89] spezialisierte Anwendungen verfolgen und auf bestimmte Ziele ausgerichtet sind. GAI und SAI bleiben noch weit entfernte Zukunftsvisionen [5]. Während an GAI bereits intensiv geforscht wird, löst die Vorstellung von SAI philosophische Diskussionen aus.

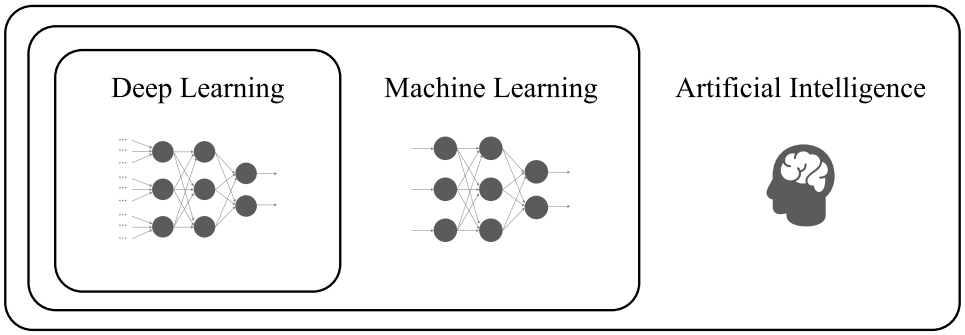


Abbildung 2.1: Venn-Diagramm, das die Beziehungen zwischen künstlicher Intelligenz (Artificial Intelligence), Machine Learning und Deep Learning darstellt

Die KI lässt sich auch technisch in verschiedene Kategorien unterteilen [54]. Ein Untergebiet der KI ist das maschinelle Lernen, das seinerseits das spezialisierte Teilgebiet des Deep Learning umfasst (siehe Abbildung 2.1). Methoden des maschinellen Lernens haben die Fähigkeit, Zusammenhänge in vorhandenen Datensätzen zu erkennen und darauf basierend Vorhersagen zu treffen. Es existieren drei Arten des maschinellen Lernens:

2.3.1 Supervised Learning

Supervised Learning bezeichnet eine Methode, bei der Algorithmen anhand einer umfangreichen Menge an annotierten (engl.: labeled) Daten trainiert werden [54]. Nach Abschluss der Trainingsphase wird in einer Testphase die Leistung des trainierten Algorithmus bewertet. Der Trainingsdatensatz nimmt im Lernprozess eine entscheidende Rolle ein, wobei sowohl der Testdatensatz als auch der Trainingsdatensatz vollständig voneinander unabhängig

sein müssen. Supervised Learning findet aufgrund seiner hohen Flexibilität und der Verfügbarkeit zahlreicher Open-Source-Bibliotheken weitverbreitete Anwendung. Die häufigsten Einsatzbereiche sind die Klassifizierung und die Regression. Während bei der Klassifizierung beschriftete Daten in diskrete Kategorien eingeteilt werden, versucht die Regression, kontinuierliche Werte vorherzusagen. Beim Bestreben, ein kompaktes System zu erreichen, könnten gegebenenfalls Kompromisse hinsichtlich der Fehlerbehandlung erforderlich sein.

2.3.2 Unsupervised Learning

Unsupervised Learning befasst sich mit dem Erkennen von Mustern in unbeschrifteten Daten [54]. Der Algorithmus identifiziert selbstständig Unterschiede und Zusammenhänge innerhalb der Daten. Diese Autonomie des Algorithmus kann als Vor- oder Nachteil betrachtet werden. Aus diesem Grund wird dieser Ansatz oft eingesetzt, um die Struktur oder Besonderheiten von Daten zu ergründen, besonders in der Phase der Voranalyse.

2.3.3 Reinforcement Learning

Reinforcement Learning stellt im Bereich vordefinierter Probleme eine effektive Strategie dar [54]. Dabei basiert es auf einer Anreiz- oder Belohnungsfunktion, die maximiert werden soll. Der Algorithmus ist darauf angewiesen, selbst herauszufinden, wie er diese maximale Belohnung erzielen kann, ohne dabei auf vorgegebene Wege oder Methoden zurückzugreifen.

2.4 Künstliche Intelligenz im operativen Einsatz

Nach der vorherigen groben Einordnung der verschiedenen Arten des maschinellen Lernens wird im Folgenden das Teilgebiet Deep Learning kurz eingeführt und allgemeine Herausforderungen im Umgang mit KI diskutiert.

2.4.1 Deep Learning und Transfer Learning

Deep Learning bezieht sich auf neuronale Netzwerke, die eine Reihe zusätzlicher Techniken nutzen, um extrem tiefe neuronale Netzwerke zu realisieren [54]. Transfer Learning, ein Ansatz des Deep Learning, setzt Ausgangsmodelle ein, die bereits umfangreich in einem bestimmten Bereich vortrainiert wurden. Es wird angenommen, dass solche Modelle, die bereits für einen bestimmten Bereich trainiert wurden, auch in weiteren oder spezifischeren Bereichen eingesetzt werden können. Sie werden mit zusätzlichen Daten aus einem anderen oder spezifischeren Bereich feinabgestimmt, um ihre Leistung in diesem speziellen Kontext zu verbessern. Ein wesentlicher

Vorteil dieser Technik ist, dass nicht ein komplett neues neuronales Modell entwickelt werden muss. Stattdessen kann ein bereits existierendes Modell verwendet werden, was eine erhebliche Einsparung an Trainingszeit, oft Stunden oder sogar Tage, bedeutet.

2.4.2 Overfitting und Underfitting

Die Entwicklung und das Training einer KI sind mit erheblichen technischen und theoretischen Herausforderungen verbunden. Eine solche Herausforderung stellt das Phänomen der Überanpassung (Overfitting) dar, welches bei den meisten der bisher erwähnten Modelle auftreten kann [54]. Wenn ein Modell zu sehr an die Trainingsdaten angepasst ist, kann es in Bezug auf andere Daten ungenau oder sogar unbrauchbar werden. Das Ziel muss daher sein, eine Modellparametrisierung zu finden, die weder zur Unteranpassung (Underfitting) noch zur Überanpassung führt.

2.5 Natural Language Processing

Natural Language Processing (NLP) ist ein Teilbereich der künstlichen Intelligenz und gleichzeitig ein Fachgebiet der Computerlinguistik [3]. Es beschäftigt sich mit der Herausforderung, menschenlesbaren Text zu verstehen und zu generieren. Die Computerlinguistik bildet die Schnittstelle zwischen Informatik und Linguistik. Ihr Ziel besteht darin, sprachliche Informationen zu entschlüsseln und zu interpretieren oder umgekehrt in eine verständliche Sprachform zu übertragen. Eine praktische Anwendung von NLP wird im Abschnitt 2.5.1 zur Sentimentanalyse sowie in den Abschnitten 2.5.2 und 2.5.3 zu den Modellen BERT [12] und RoBERTa [47] dargestellt.

2.5.1 Sentimentanalyse

Die Sentimentanalyse für Texte zielt darauf ab, den emotionalen Zustand des Autors zum Zeitpunkt der Texterstellung zu bewerten [41]. Hierbei kann das Sentiment in eine Polarität eingeteilt werden: positiv, neutral oder negativ. Ebenfalls ist es möglich das Sentiment anhand einer Vielzahl von Gefühlen zu klassifizieren, wie etwa den Grundemotionen nach Ekman (Furcht, Freude, Überraschung etc.) [17]. Zur Bestimmung des Sentiments eines Textes existieren verschiedene Ansätze [53]. Einerseits kann der eigentliche Inhalt oder das Thema des Textes betrachtet werden, andererseits spielt die Ausdrucksweise und der Ton des Verfassers eine entscheidende Rolle, um den emotionalen Gehalt zu erfassen. Eine umfassende Sentimentanalyse berücksichtigt sowohl den Inhalt als auch die Ausdrucksweise eines Textes, um dessen Sentiment zu klassifizieren.

Inhaltliche Bestimmung

Diese Analysemethode konzentriert sich auf den eigentlichen Inhalt oder das zentrale Thema des Textes. Ein Beispiel hierfür wäre ein App-Nutzer, der in einem Review äußert: „Ich mag die Werbung nicht“. Das Hauptthema dieses Satzes ist die „Werbung“. Daher spiegelt der Satz eine Meinung zum Inhalt wider, in diesem Fall eine negative Haltung gegenüber der Werbung.

Bestimmung durch Ausdrucksweise

Der Fokus liegt hier auf dem Ton oder dem emotionalen Gehalt der Worte, unabhängig vom eigentlichen Thema des Textes. Ein Beispiel für dieses Szenario wäre der Satz „Schon wieder das Gleiche...“ in einem Review. Dieser Satz könnte in verschiedenen Kontexten vorkommen, beispielsweise in Bezug auf eine Funktionserweiterung oder das Wetter. Jedoch vermittelt die Ausdrucksweise, insbesondere durch „Schon wieder“ und die „...“ am Ende, ein negatives Sentiment.

2.5.2 BERT

BERT [12] („Bidirectional Encoder Representations Transformers“) wurde im Jahr 2018 von Google eingeführt und setzte einen neuen Maßstab für die Leistung in der NLP. Bei BERT handelt es sich um ein „pre-trained language model“, das heißt, dass ein Modell zu Beginn auf eine umfassende Menge nicht annotierter Textdaten trainiert wurde, um zunächst ein allgemeines Verständnis der natürlichen Sprache zu entwickeln. Nach diesem sogenannten Vortraining („pre-training“) kann BERT für die unterschiedlichsten NLP-Aufgaben eingesetzt werden, indem eine Feinabstimmung mithilfe annotierter Daten zu dem betreffenden NLP-Problem bereitgestellt wird, um BERT auf diese Daten weiter zu trainieren. Diese Feinabstimmung („fine-tuning“) ermöglicht es, das BERT-Modell auf spezifische Aufgaben anzupassen, um somit eine bessere Performance als das Basismodell BERT für die Aufgabe zu erzielen. Beispiele für solche spezifischen Aufgaben sind die Textklassifizierung, das Zusammenfassen oder Übersetzen von Texten. Frühere Ansätze, wie z.B. OpenAI GPT [73], beschränkten sich darauf, entweder auf den vorherigen oder den nachfolgenden Kontext eines Wortes Bezug zu nehmen. Im Gegensatz dazu bietet BERT die Möglichkeit, sowohl auf den vorherigen als auch auf den nachfolgenden Kontext zuzugreifen, um die Bedeutung eines Wortes im Kontext besser zu erfassen und zu verstehen. Für das Vortrainieren des BERT-Modells wurden zwei unsupervided Aufgaben verwendet: „Masked Language Model“ (MLM) und „Next Sentence Prediction“ (NSP). Für die Aufgabe MLM werden wenige Prozente der Eingabe maskiert, welche daraufhin von dem Modell vorhergesagt werden sollen. NSP beschäftigt sich mit der Aufgabe, bei der Eingabe von zwei Sätzen zu entscheiden, welcher Satz vor dem

anderen Satz aufgetreten ist. Die Daten für das Pre-Training stammen von zwei unterschiedlichen Korpora: dem „BooksCorpus“ [90] (800M Wörter) und der englischen Variante von Wikipedia (2.500M Wörter), bei denen die Textpassagen herausextrahiert und unter anderen Listen, Tabellen ignoriert wurden. Google stellte zwei Modellgrößen für BERT vor: $BERT_{BASE}$ und $BERT_{LARGE}$. $BERT_{BASE}$ stellt eine kleinere Variante von BERT dar und benötigt somit weniger Rechenaufwand. $BERT_{LARGE}$ hingegen ist die größere Variante von BERT, welche mehr Kontextinformationen erfassen kann und oft in mehreren NLP-Aufgaben bessere Performance als $BERT_{BASE}$ besitzt. Jedoch ist der Rechenaufwand größer und die Trainingsphase nimmt mehr Zeit für die Verwendung von $BERT_{LARGE}$ in Anspruch.

2.5.3 RoBERTa

RoBERTa [47] („A Robustly Optimized BERT Pretraining Approach“) ist eine etablierte Weiterentwicklung von BERT, die von Meta[1] (ehemals Facebook) veröffentlicht wurde. Die Änderungen im Vergleich zum Ausgangsmodell BERT umfassen eine längere Trainingszeit, größere Batches während des Trainings mit einem höheren Datenvolumen, das Entfernen der „Next Sentence Prediction“ (NSP), das Training auf längere Sequenzen sowie das dynamische Verändern der Maskenmuster (MLM). RoBERTa wurde durch das Training mit fünf englischsprachigen Korpora (insgesamt 160 GB unkomprimierten Textes) unterschiedlicher Größe und Domäne entwickelt. Es stellt eine signifikante Verbesserung im Vergleich zum ursprünglichen Modell $BERT_{LARGE}$ dar. Dies unterstreicht die Bedeutung der Datengröße und Vielfalt während des Pre-Trainings.

2.6 Evaluation

Die Evaluation dient dazu, die Funktionsweise eines Modells oder eines Verfahrens zu bewerten und zu optimieren. Ein Instrument für die systematische Analyse einer KI stellt die Konfusionsmatrix mit den Werten True-Positive (TP), True-Negative (TN), False-Positive (FP) und False-Negative (FN) dar [71].

Angenommen, es wurde eine KI entwickelt, die automatisch ermittelt, ob ein Programmcode hinsichtlich Laufzeit oder Speicherbedarf effizient programmiert wurde. Für die Evaluation der KI kommt ein binäres Klassifikationssystem zum Einsatz: Die KI stuft den Code entweder als „effizient“ oder „nicht effizient“ ein.

Die in Abbildung 2.2 dargestellte Konfusionsmatrix, die eine übersichtliche Darstellung der Klassifikationsergebnisse bietet, umfasst folgende Werte:

[1] `https://about.meta.com`

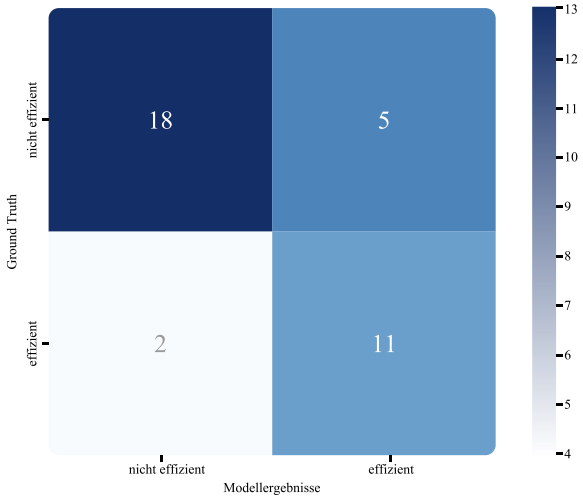


Abbildung 2.2: Konfusionsmatrix zur Evaluation einer KI im Kontext der Effizienzerkennung von Programmcode

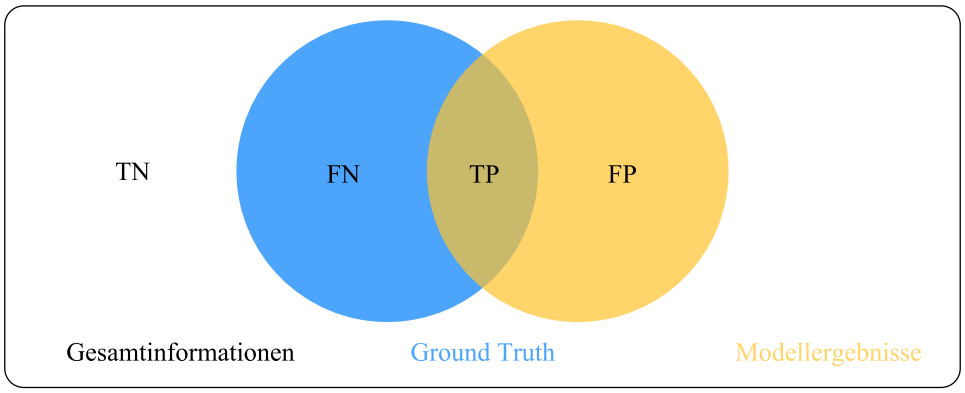
TP: Der Code ist effizient und die KI erkennt diesen Zustand korrekt (Summe: 11).

TN: Der Code ist nicht effizient und die KI erkennt diesen Zustand korrekt (Summe: 18).

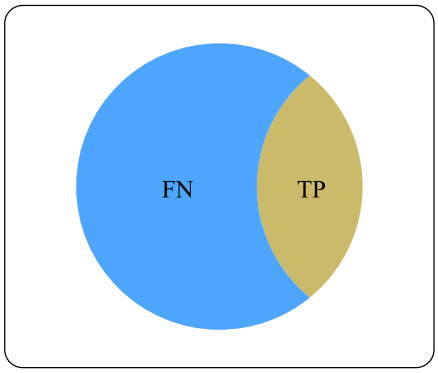
FP: Obwohl der Code ineffizient ist, stuft die KI ihn fälschlicherweise als effizient ein (Summe: 5).

FN: Trotz eines effizienten Codes klassifiziert die KI diesen fälschlicherweise als ineffizient (Summe: 2).

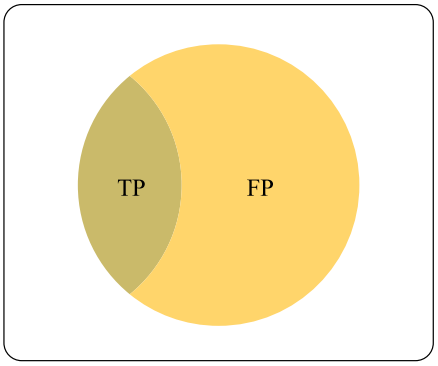
Die Abbildung 2.3 gibt einen anschaulichen Überblick über die Bestandteile einer Konfusionsmatrix und erläutert die Begriffe Gesamtinformationen („Ground Truth“) und Modellergebnisse. Die Gesamtinformationen umfassen alle Informationen, die während des Dokumentationsprozesses gesammelt wurden [51]. Die „Ground Truth“ bezeichnet die Daten, welche als korrekt angesehen werden. Die Modellergebnisse sind die Daten, die vom KI-System vorhergesagt werden.



(a) Übersicht über den Anteil des „Ground Truth“ und der Modellergebnisse



(b) Recall



(c) Precision

Abbildung 2.3: Venn-Diagramme zur Darstellung der Verteilung von TP, TN, FP, FN - adaptiert von C. Manning und H. Schütze [51]

Metriken

Basierend auf der Konfusionsmatrix lassen sich die Metriken Precision, Recall und F1-Score ableiten [51, 74]. Die Precision gibt die Genauigkeit eines Modells an, indem sie das Verhältnis der korrekt als positiv vorhergesagten Instanzen ($|TP|$) zur Gesamtanzahl der positiven Vorhersagen ($|TP|+|FP|$) misst (siehe Abbildung 2.3c).

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

Der Recall gibt die Vollständigkeit eines Modells an, indem er das Verhältnis der korrekt als positiv vorhergesagten Instanzen ($|TP|$) zur Gesamtanzahl

tatsächlich positiver Instanzen ($|TP|+|FN|$) misst (siehe Abbildung 2.3b).

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

Der F1-Score gibt den harmonischen Durchschnitt von Precision und Recall an und bietet somit eine ausgewogene Bewertung der Modellleistung. Er ist besonders nützlich, wenn sowohl Precision als auch Recall wichtig sind und ein Kompromiss zwischen beiden angestrebt wird.

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

Zusätzlich lässt sich die Accuracy als weitere Metrik nennen, welche die Korrektheit eines Modells darstellt. Sie setzt sich aus dem Verhältnis der korrekt klassifizierten Instanzen zur Gesamtanzahl klassifizierter Instanzen zusammen. Anhand dieser Metrik kann eine allumfassende Modellleistung abgeleitet werden.

$$\text{Accuracy}_{binary} = \frac{|\text{TP}|+|\text{TN}|}{|\text{TP}|+|\text{TN}|+|\text{FP}|+|\text{FN}|}$$

Zu jeder zuvor genannten Metrik gibt es mindestens eine weitere Metrik, die für den Fall eines Multiklassenproblems gilt [51]. Jede Klasse kann in einem Multiklassenproblem auf binäre Weise ausgewertet werden. Zu den Metriken Precision, Recall und F1-Score gibt es jeweils zusätzliche Durchschnittstypen: der Macro- und Micro-Durchschnitt. Der Micro-Durchschnitt einer Metrik berücksichtigt die Anzahl der Instanzen in jeder Klasse, wodurch größere Klassen eine größere Gewichtung erhalten [41]. Der Macro-Durchschnitt einer Metrik berechnet den Durchschnitt über alle Klassen, sodass jede Klasse die gleiche Gewichtung besitzt.

$$\text{Micro-Precision} = \frac{\sum_{i=1}^{k} |\text{TP}_i|}{\sum_{i=1}^{k} |\text{TP}_i| + \sum_{i=1}^{k} |\text{FP}_i|}$$

$$\text{Macro-Precision} = \frac{\sum_{i=1}^{k} \text{Precision}_i}{k}$$

$$\text{Micro-Recall} = \frac{\sum_{i=1}^{k} |\text{TP}_i|}{\sum_{i=1}^{k} |\text{TP}_i| + \sum_{i=1}^{k} |\text{FN}_i|}$$

$$\text{Macro-Recall} = \frac{\sum_{i=1}^{k} \text{Recall}_i}{k}$$

$$\text{Micro-F1-Score} = \frac{2 \cdot \text{Micro-Precision} \cdot \text{Micro-Recall}}{\text{Micro-Precision} + \text{Micro-Recall}}$$

$$\text{Macro-F1-Score} = \frac{\sum_{i=1}^{k} \text{F1-Score}_i}{k}$$

$$\text{Accuracy}_{\text{multi}} = \frac{\sum_{i=1}^{k} |\text{TP}_i|}{\sum_{i=1}^{k} |\text{TP}_i| + \sum_{i=1}^{k} |\text{FP}_i|}$$

Goldstandard und Baseline

Um zu überprüfen, ob das entwickelte KI-Modell anderen Ansätzen überlegen ist, wird es häufig mit dem Goldstandard oder der Baseline verglichen [86]. Der Goldstandard definiert die obere Grenze für die Effektivität eines Ansatzes und repräsentiert das optimal erreichbare Ergebnis für eine bestimmte Aufgabe. Sollte die Ermittlung des Goldstandards nicht möglich sein, wird häufig die sogenannte „Ground Truth“ als Referenzwert herangezogen. In vielen Fällen wird deshalb der Mensch als das beste Maß betrachtet.

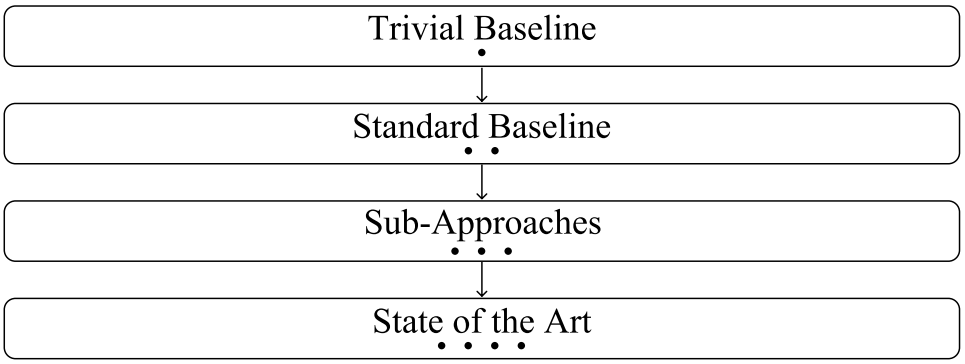


Abbildung 2.4: Hierarchische Einteilung einer Baseline

Die Baseline ist ein weiterer Ansatz zur Bewertung der Effektivität eines Modells [86]. Sie stellt die untere Grenze der Effektivität eines Ansatzes dar. Es gilt, dass ein neuer Ansatz immer besser sein sollte als jede aufgestellte Baseline. Es existiert eine Unterteilung von Baselines (siehe Abbildung 2.4). Diese beginnt mit der trivialen Baseline, die einfach von der gegebenen Aufgabe abgeleitet werden kann. Sie wird verwendet, um zu überprüfen, ob der neue Ansatz überhaupt eine Leistung erbringt. Darauf folgt die Standard-Baseline, also die Methode, die häufig für die genannte Aufgabe angewendet wird. Sie dient dazu, die Schwierigkeit der Aufgabe einzuschätzen. Ein Teilaspekt eines neuen Ansatzes kann ebenfalls als Baseline („Sub-Approaches“) dienen, um den Einfluss verschiedener Ausprägungen des Ansatzes zu analysieren. Schließlich gibt es den „State of the Art“, die derzeit beste veröffentlichte Methode für die Bewältigung der Aufgabe. Sie dient dazu, zu überprüfen, ob der neue Ansatz tatsächlich der effektivste ist.

Kapitel 3

Konzepte

In diesem Kapitel werden die zentralen Konzepte vorgestellt, die als Grundlage für die anschließende Implementierung dienen. Der Einstieg erfolgt mit dem Abschnitt 3.1 zu den Software-Anforderungen. In Abschnitt 3.2 wird die Definition der potenziellen Nutzer erläutert, welche die entwickelte Anwendung nutzen werden. Abschnitt 3.3 bietet einen allgemeinen Überblick über den Ablauf der Anwendung, während Abschnitt 3.4 einen Einblick in die Auswahl der verwendeten Klassifizierer gibt. Abschließend wird in Abschnitt 3.5 die Entscheidung für die Plattformauswahl begründet.

3.1 Anforderungen

Aus dem Grundlagenkapitel im Abschnitt 2.1 geht hervor, dass die Anforderungserhebung für die Softwareentwicklung essenziell ist. Die initialen

Initiale Anforderungen für die Bachelorarbeit	
Funktional	✓RF1 Erfassung von App-Reviews aus: ✓RF1.A Google Play Store ✓RF1.B Apple App Store ✓RF1.C CSV-Datei ✓RF2 Filteroptionen (z.B. nach Sternebewertung oder Länge des Reviews) ✓RF3 Analysemöglichkeiten (z.B. Sentimentanalyse) ✓RF4 Visualisierung der Analyseergebnisse
Nicht Funktional	✓RN1 Modularität ✓RN2 Usability ✓RN3 Performance der Klassifizierer

Abbildung 3.1: Priorisierte initiale Anforderungen für die Bachelorarbeit

Anforderungen, die in Abbildung 3.1 priorisiert aufgelistet werden, resultierten aus der Ausschreibung der Bachelorarbeit des Instituts für Praktische Informatik, genauer aus dem Fachgebiet Software Engineering. Neben diesen

initialen Anforderungen, die lediglich die Grundlage der Software darstellen sollten, wurden erweiterte Anforderungen (siehe Abbildung 3.2) berücksichtigt, die auf den ursprünglichen Anforderungen aufbauen. Um die Wünsche der Nutzer besser zu verstehen, wurde die Perspektive eines App-Analysten eingenommen. Infolgedessen wurden weitere Funktionen, insbesondere das Session-Management (RF5), integriert, um verschiedene Konten innerhalb der Software zu ermöglichen. Darüber hinaus ergaben sich spezifischere Anforderungen, etwa ein FAQ (RF8) zum technischen Hintergrund der Klassifizierer oder einen App-Store-Vergleich (RF6) der App-Reviews. Des Weiteren fanden Gespräche mit Informatikstudierenden statt, die potenzielle zukünftige Fachkräfte im Bereich Softwareentwicklung sind, um Anregungen zu neuen Funktionen zu sammeln. Dabei ergaben sich Anforderungen bezüglich der Verbesserung der Performance der Klassifizierer (RN6). Ebenfalls wurden im Unternehmen *devoteam* [13] erfahrene Entwickler konsultiert, um tieferes Wissen in der Anforderungserhebung zu gewinnen. Hierbei wurden Empfehlungen ausgesprochen, die Software testbar zu gestalten und Use-Cases (RN5) zu pflegen, um einen nachhaltigen Produktlebenszyklus der Software anzustreben. Zusammenfassend lässt sich feststellen, dass

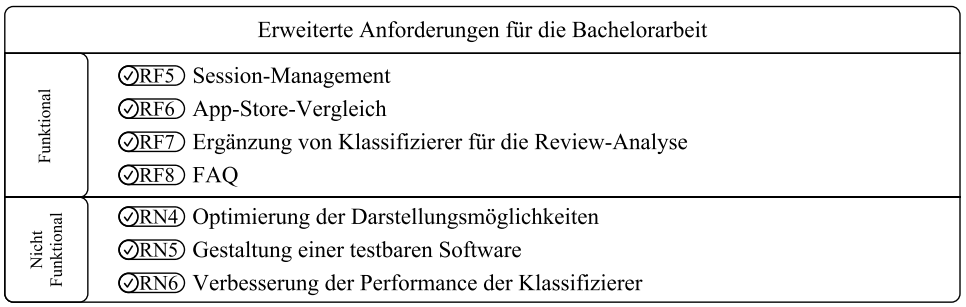


Abbildung 3.2: Priorisierte erweiterte Anforderungen für die Bachelorarbeit

die initialen Anforderungen stetig erweitert wurden. Beispiele hierfür sind die Ergänzung von Klassifizierer (RF7) für die Review-Analyse oder die Optimierung der Darstellungsmöglichkeiten (RN4).

3.2 Nutzer

Die neu entwickelte Software *feelio* soll dem Benutzer eine umfassende Analyse seiner importierten oder gecrawlten App-Reviews bieten. Der typische Benutzer definiert sich dabei als App-Analyst, Entwickler, Marktforschungsinstitut oder ähnlich engagierter Reviews-Analyst, der beabsichtigt, nützliche Informationen aus den Inhalten der Reviews zu extrahieren. Es wurde der kurze Name *feelio* für die Applikation gewählt, um sicherzustellen, dass im Arbeitsumfeld sofort klar ist, um welche Anwendung es sich handelt. Dies wird am besten durch einen einprägsamen und kurzen Namen erreicht [42].

Die Nutzer der Applikation könnten unterschiedliche Intentionen hinsichtlich des Verwendungszwecks der Anwendung haben. So kann der App-Analyst oder Entwickler beispielsweise die täglich anfallende Menge an Reviews besser interpretieren. Besonders für das Requirements Engineering bietet die Software nützliche Methoden, um einen täglichen, monatlichen oder jährlichen Statusbericht ihrer App zu erstellen. Auf diese Weise können frühzeitig aufkommende Trends erkannt und darauf entsprechend reagiert werden. Ein Marktforschungsinstitut oder eine Einzelperson mit Interesse an einem Marktüberblick könnte ebenfalls einen Nutzen aus der Anwendung ziehen. Dank der Möglichkeit, App-Reviews aus den beliebtesten App-Stores oder einer CSV-Datei zu analysieren, stellt die Software ein wertvolles Werkzeug für diese Zielgruppe dar. Durch den Einsatz der Software und die Verwendung verschiedener Klassifizierer lassen sich zahlreiche Korrelationen zwischen unterschiedlichen Metriken identifizieren. Diese Erkenntnisse können für die Nutzergruppe von entscheidender Bedeutung sein und wertvolle Einblicke liefern.

3.3 Funktionsablauf der Anwendung

Nach der Identifizierung der potenziellen Nutzer der Applikation werden die verschiedenen Interaktionsmöglichkeiten vorgestellt, mit denen die Nutzer in der fertiggestellten Software interagieren werden. Bei der Gestaltung dieser Prozesse wurde stets auf die „10 Usability Heuristics for User Interface Design“ von Jakob Nielsen [55] geachtet (siehe Kapitel 2.2). Das Ziel war es, dem Nutzer eine kohärente Struktur innerhalb der Software zu bieten und eine einzigartige Benutzererfahrung zu gewährleisten.

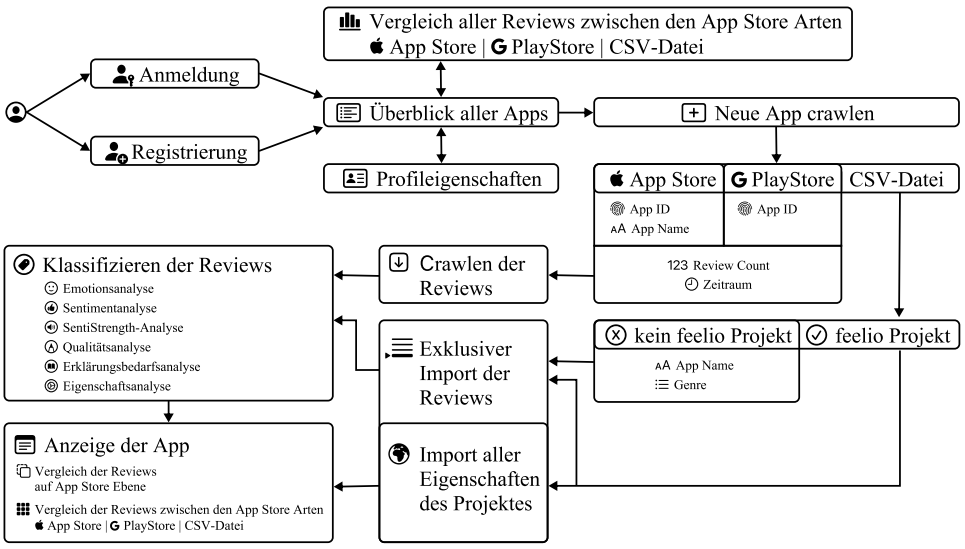


Abbildung 3.3: Funktionsablauf der Anwendung

Den Ablauf der Interaktionen mit der Applikation vermittelt die Abbildung 3.3. Eine Person, welche die Anwendung nutzen möchte, erhält ohne Authentifizierung keinen Zugang zum Funktionsumfang der Anwendung. Hierfür ist eine Registrierung erforderlich, oder, falls bereits ein Konto besteht, eine Anmeldung. Dieser Prozess bedient beispielhaft die Usability-Heuristiken eins, drei und vier. Heuristik eins kommt zur Anwendung, weil der Nutzer ein Konto erstellen kann und daher weiß, dass seine Daten aus vorherigen Sitzungen gespeichert sind, sobald er angemeldet ist. Heuristik drei spiegelt sich wider, indem der Nutzer die Freiheit hat, verschiedene Konten zu erstellen. So kann er verschiedene Analysen durchführen, ohne dass andere Konten davon Kenntnis erhalten. Heuristik vier zeigt sich darin, dass es für eine Anwendung üblich ist, Fortschritte über ein allgemeines Profil speichern zu können. Dies gewährleistet eine konsistente Benutzererfahrung. Nach erfolgreichem Anmelde- oder Registrierungsprozess gelangt der Nutzer zur Hauptseite, auf der alle bereits analysierten Apps aufgelistet sind. Über die Navigationsleiste hat der Nutzer die Möglichkeit, drei Hauptseiten von jeder Unterseite aus aufzurufen: den Vergleich aller Reviews zwischen den App-Store-Arten, einen Überblick aller Apps und die Profilinformationen. Die Funktion der Navigationsleiste erfüllt die Usability-Heuristik vier, da eine standardisierte Navigation normalerweise über eine Navigationsleiste im oberen Bereich der Seite erfolgt. Dies ermöglicht dem Nutzer sofortiges Verständnis des gesamten Aufbaus der Webseite und verhindert, dass er lange nach Navigationsoptionen suchen muss.

Um eine neue App-Analyse zu beginnen, muss dies über die Seite „Überblick aller Apps“ eingeleitet werden. Dabei stehen dem Nutzer drei verschiedene Review-Datenquellen zur Auswahl: der Apple App Store, der Google Play Store oder eine CSV-Datei. Für die Kategorien Apple App Store und Google Play Store sind die Felder „App-ID“, „Review Count“ und der Zeitraum, in dem die Reviews gecrawlt werden sollen, obligatorisch. Für den Apple App Store ist zusätzlich der „App Name“ erforderlich. Hilfestellungen zu den jeweiligen Eingabefeldern sind durch ein Info-Icon direkt in der Eingabemaske integriert, sodass jeder Anwender sofort versteht, welche Informationen genau benötigt werden. Dies bedient die Usability-Heuristik zehn und unterstützt den Nutzer optimal beim Erstellen einer App-Analyse. Beim Import einer CSV-Datei wird im Hintergrund geprüft, ob diese ein bestehendes *feelio*-Projekt beinhaltet. Wenn dies der Fall ist, kann der Nutzer entscheiden, ob er das gesamte Projekt mit all seinen Analysen oder lediglich die vorhandenen App-Reviews importieren und somit die Analyse der Reviews erneut durchführen möchte. Mehrere Hinweisfenster unterstützen dabei, klar zu kommunizieren, welches Importverfahren der Nutzer bevorzugt (Usability-Heuristiken: zwei, fünf und sieben). Wenn kein *feelio*-Projekt in der CSV-Datei gefunden wird, sind die Eingabefelder „App Name“ verpflichtend, während das Hinzufügen des Genres optional ist. Bei einer fehlerhaften Eingabe, wie beispielsweise einer nicht korrekt formatierten „App-ID“, zeigt

das User Interface in Echtzeit einen Fehler mit entsprechender Hilfestellung an. Dies verhindert Frustration beim Nutzer und erfüllt somit die Usability-Heuristik neun. Es ist möglich, die verschiedenen Review-Datenquellen kombiniert zu nutzen. So kann beispielsweise eine App-Analyse erstellt werden, die sämtliche Datenquellen, eine CSV-Datei, den Apple App Store und den Google Play Store, umfasst und somit Reviews aus allen genannten Quellen integriert. Sobald der Bestätigungsbutton zum endgültigen Crawlen und Analysieren der App-Review-Daten aktiviert wird, erscheint während der Ladezeit ein Fortschrittsindikator. Dieser visualisiert, je nach aktuellem Hintergrundprozess, den Status und zeigt einen prozentualen Ladebalken an. Damit folgt die Anwendung der Usability-Heuristik eins und sorgt dafür, dass der Nutzer jederzeit den aktuellen Zustand der Applikation erkennt und in der Interaktion nicht irritiert wird. Jedes Review wird sechs verschiedenen Analysemethoden unterzogen:

- **Die Emotionsanalyse** untersucht den Inhalt des Reviews auf Basis der Grundemotionen nach Ekman [17]: Zorn („anger“), Abscheu („disgust“), Furcht („fear“), Freude („joy“), Traurigkeit („sadness“), Überraschung („surprise“) und eine zusätzliche neutrale Klasse. Diese Methode bietet eine Vielzahl an Vorteilen und liefert einen Ersteindruck davon, auf welcher emotionalen Ebene sich das Review bewegt. Ein App-Analyst könnte beispielsweise Reviews, die das Label „joy“ tragen, genauer analysieren, um herauszufinden, welche Funktionen den Nutzern besonders viel Freude bereiten. Auf diese Weise kann mehr Wert auf die Entwicklung solcher Features gelegt und sichergestellt werden, dass diese nützlichen Funktionen nicht versehentlich bei zukünftigen Aktualisierungen entfernt werden. Umgekehrt kann durch die Emotionsanalyse nach dem Label „sadness“ gesucht werden, um die Unzufriedenheit der Kunden besser verstehen zu können. Dieser Ausdruck kommt insbesondere dann zum Vorschein, wenn eine geliebte Funktion entfernt wurde. Der Tag „anger“ kann hingegen auf generelle, extreme Unzufriedenheit hinweisen und vielleicht sogar das potenzielle Entfernen der App durch den Nutzer signalisieren.

- **Die Sentimentanalyse** besteht aus einem eigenständig trainierten Transformer-Modell, welches jedem Review eine der Sentimentpolaritäten (positiv, neutral oder negativ) zuordnet. Durch die Anwendung dieses Klassifizierers erhält man einen guten Überblick über das grundlegende Sentiment des Reviews. Die Verwendung des eigenständig trainierten Transformer-Modells gewährleistet eine möglichst präzise Vorhersage der Sentimentpolarität für ein Review. Zusätzlich kommt das NLP-Tool SentiStrength [83] zum Einsatz, welches positive und negative Begriffe innerhalb eines Reviews kennzeichnet. Die Funktion erweist sich als besonders vorteilhaft, da sie App-Analysten dabei unterstützt, gezielt jene Abschnitte eines Reviews zu identifizieren,

die positive oder negative Stimmungen ausdrücken. Häufig beinhalten Reviews sowohl positive als auch negative Kommentare, was zu einer Gewichtung für die Gesamtsentimentpolarität führt [25]. Diese detaillierte Analyse ermöglicht eine präzisere Wahrnehmung von Teilsentimenten innerhalb des Reviews. Des Weiteren wird mithilfe von SentiStrength für jedes Review ebenso eine Sentimentpolarität ermittelt, um die Performance im Vergleich zum eigenständig trainierten Transformer-Modell zu evaluieren. Eine derartige doppelte Darstellung der Sentimentpolarität könnte den Nutzer verwirren. Infolgedessen würde lediglich die effektivere Analysemethode in das finale Produkt integriert werden.

- **Die Erklärungsbedarfsanalyse** („Explanation Needed“) ermittelt, ob ein Review Erklärungsbedarf aufweist oder nicht. Eine mögliche Anwendung dieses Klassifizierers ergibt sich, wenn App-Analysten auf Reviews reagieren möchten, um auf Fragen und Anliegen der Review-Autoren einzugehen. Damit können sie nicht nur das Unternehmensimage pflegen, sondern auch wertvolle Hinweise zur Verbesserung der App sammeln [49]. Mithilfe dieser Funktion können sie alle Reviews filtern, die eine Antwort erfordern, und somit effektiv und zielgerichtet im jeweiligen App Store auf die Reviews reagieren.

- **Die Qualitätsanalyse** („Gibberish“) beurteilt, wie gut ein jeweiliges Review vom Verfasser geschrieben wurde. In vielen Bereichen des Internets hat sich eine Kultur der informellen Sprache entwickelt, so auch innerhalb der für Apps verfassten Reviews [6]. Diese Reviews unterscheiden sich von formalen Texten dadurch, dass sie meist kürzer sind, Schreibfehler enthalten oder erfundene Wörter wie zum Beispiel „spamspamspamspam“ aufweisen können. Solche sprachlichen Herausforderungen können das Verständnis und die Performance mancher Klassifizierer beeinflussen. Daher wurde ein Klassifizierer verwendet, der ein Review auf Grundlage seiner sprachlichen Qualität in die folgenden Kategorien einordnet: sehr schlecht („noise“), schlecht („word salad“), halbwegs gut („mild Gibberish“), sehr gut („clean“). Je nach gecrawlter App kann ein signifikanter Anteil der Reviews schlecht geschrieben sein. Unter der Annahme, dass das bloße Aneinanderreihen von zufälligen Wörtern keine effektive Review-Aussagekraft hat, können solche Reviews, die zum Beispiel als „noise“ gekennzeichnet sind, herausgefiltert werden. So ermöglicht die Qualitätsanalyse dem App-Analysten, seinen Fokus auf potenziell inhaltlich relevantere Reviews zu legen.

- **Die Eigenschaftsanalyse** („Characteristic“) klassifiziert jedes Review anhand spezifischer Merkmale: Nutzererfahrung („user experience“), Funktionsanforderung („feature request“), Textbewertung („text ra-

ting“) oder Fehlerbericht („bug report“). Diese Einteilung der Reviews erweist sich insbesondere für das Requirements Engineering als überaus nützlich, indem sie eine effiziente Vorsortierung der Reviews ermöglicht [22, 26, 64, 66]. Dies erleichtert die schnelle Ableitung gezielter Maßnahmen zur Verbesserung der App, optimiert das Nutzererlebnis und ermöglicht eine schnelle Reaktion auf Marktentwicklungen.

Die gesamte Applikation wurde unter Verwendung eines modularen Ansatzes entwickelt, der das Austauschen und Hinzufügen von Klassifizierern ermöglicht. Es ist zudem hervorzuheben, dass zwischen einigen der eingesetzten Klassifizierer Überschneidungen bestehen. Dies ermöglicht einen Vergleich der eingesetzten Klassifizierer, um ihre jeweiligen Stärken in verschiedenen Bereichen zu erkennen und zu nutzen. Nach Abschluss der Review-Verarbeitung stellt die Anwendung dem Nutzer detaillierte und aufbereitete Statistiken für die individuelle App-Analyse zur Verfügung. Je nachdem welche Review-Datenquellen vom Nutzer ausgewählt wurden, besteht die Möglichkeit, die Reviews separat nach Datenquelle (bzw. App Store Typ) zu analysieren, beispielsweise nur die Reviews aus dem Apple App Store, dem Google Play Store oder aus einer CSV-Datei. Zusätzlich zu den Einzeldarstellungen der Review-Datenquellen bietet die Anwendung auch die Option, die verschiedenen Datenquellen miteinander zu vergleichen. So können wesentliche Unterschiede zwischen den Metriken der verschiedenen Review-Datenquellen herausgearbeitet werden. Insbesondere bei Cross-Plattform-Apps (Apps, die gleich auf vielen unterschiedlichen Plattformen funktionieren) bietet sich so die Chance, die Reaktionen auf einzelne Features oder bestimmte Fehler auf den unterschiedlichen Plattformen zu beobachten. Hierdurch kann erkannt werden, ob etwa spezifische Probleme aufgrund der verschiedenen Gerätetypen entstehen. Auf jeder Analyse-Seite befinden sich die Hauptkategorien: „Statistiken“ („Statistics“), „Bildschirmaufnahmen“ („Screenshots“), „Reviews“ und „Grafiken“ („Charts“). Jede dieser Hauptkategorien kann nach Belieben ein- und ausgeblendet werden, um die Übersichtlichkeit zu gewährleisten. Ein App-Analyst hat somit die Möglichkeit, gezielt jene Analysebestandteile einzublenden, die für ihn von Relevanz sind. Dies optimiert den Analyseprozess und minimiert das Potenzial für Fehler bei Metrikvergleichen, ganz im Einklang mit den Usability-Heuristiken fünf, sieben und acht.

In der Hauptkategorie „Statistiken“ erhält der Nutzer eine Übersicht über die durchschnittliche Sternebewertung der erfassten Reviews, deren durchschnittliche Länge, das durchschnittliche Sentiment sowie den Zeitraum, aus dem die Reviews stammen. Diese grundlegenden Statistiken verschaffen einen guten Überblick über die Merkmale einer App und können vom App-Analysten effektiv genutzt werden, um Korrelationen zu den angewandten Review-Klassifizierungen zu erkennen. Das durchschnittliche Sentiment wird mittels eines Farbverlaufs, der sich zwischen Grün und

Rot erstreckt, veranschaulicht. Dies ermöglicht dem Analysten, auf den ersten Blick das Sentiment-Niveau der App-Reviews zu erfassen und bei Bedarf umgehend geeignete Gegenmaßnahmen einzuleiten. Innerhalb der zweiten Hauptkategorie „Screenshots“ werden die aktuellen Werbebilder der App aus dem jeweiligen App Store dargestellt. Dadurch erhält der Analyst einen Einblick in die visuelle Präsentation der App, wie sie den Nutzern beim Download angezeigt wird. Diese Sicht ermöglicht es dem Analysten, die Werbebilder als zusätzliches Analysekriterium zu verwenden und zu überprüfen, ob die Features, die in den Reviews am häufigsten hervorgehoben wurden, auch in der App-Werbung richtig betont werden.

Die Hauptkategorie „Reviews“ beinhaltet alle zusammengetragenen Reviews, inklusive ihrer zugehörigen Klassifizierungs-Tags. Zur besseren visuellen Unterscheidbarkeit der verschiedenen Klassifizierungen werden diese durch passende Emojis, Farben und Formen hervorgehoben. Dadurch kann der Analyst die unterschiedlichen Klassifizierer auf den ersten Blick unterscheiden und das Reviews schneller einschätzen. Die Sortierung der Reviews kann nach unterschiedlichen Vergleichskriterien angepasst werden, darunter das Erstellungsdatum, die Sternebewertung und die Länge der Reviews. Hierdurch kann der Nutzer zuerst die Reviews anschauen, die ihn am meisten interessieren (Heuristiken: zwei, vier und sechs).

Die abschließende Hauptkategorie „Charts“ fasst alle Klassifizierer in mehreren Diagrammen zusammen. Für den Klassifizierer, der zur Bestimmung der Sentimentpolarität der Reviews eingesetzt wird, stehen zwei Diagramme zur Verfügung: Ein Kreisdiagramm, das die Gesamtverteilung der Sentimentpolaritäten darstellt, sowie ein Balkendiagramm, das den monatsabhängigen Verlauf der Verteilung der Sentimentpolaritäten visualisiert. Das Balkendiagramm eignet sich ausgezeichnet, um Trends in der Verteilung der Sentimentpolarität zu erkennen und darauf basierend zukünftige Reaktionen zu planen. Für den Klassifizierer, der die Qualität des geschriebenen Formats („Gibberish“) ermittelt, wird ein Kreisdiagramm verwendet. Dieses illustriert die Gesamtverteilung der Reviews in den jeweiligen Qualitätskategorien. Ähnlich verfahren die Klassifizierer, welche die Emotionen identifizieren und einen Erklärungsbedarf erkennen: Beide setzen Kreisdiagramme zur Visualisierung der Gesamtverteilungen ihrer jeweiligen Kategorien ein. Mithilfe der Kreisdiagramme kann auf einen Blick die Verteilung eines Maßes erfasst werden. Die kreisförmige Darstellung ermöglicht es, auf einfache Weise zu erkennen, welcher Anteil am größten ist und wie die einzelnen Anteile zueinander in Relation stehen. Für den Klassifizierer, der die Zuweisung der Charakteristiken vornimmt, wird ein Balkendiagramm verwendet. Dieses visualisiert die Wahrscheinlichkeit, mit der ein Review beispielsweise als Fehlerbericht („bug report“) gekennzeichnet ist. Die Wahl des Balkendiagramms bietet sich hier besonders an, da es eine klare, übersichtliche Darstellung der unterschiedlichen Wahrscheinlichkeiten ermöglicht und es den direkten Vergleich der einzelnen Kategorien unter-

stützt. Es erlaubt dem Analysten, schnell und effizient Einblicke in die Daten zu gewinnen und eventuelle Trends oder Auffälligkeiten zu identifizieren.

In der oberen Ecke jeder Statistik-Seite befinden sich vielfältige Filteroptionen: Sternebewertung („Rating“), Länge („Length“), Sentiment, Qualität des Reviews („Gibberish“), Emotion, Charakteristik („Charakteristic“) und Erklärungsbedarf („Explanation Needed“). Je nachdem, welcher Filter aktiviert wird, erscheinen individuelle Einstellungsoptionen für eine präzise Anpassung des Filters. Es ist möglich, alle Filter miteinander zu kombinieren und in Echtzeit (siehe Usability-Heuristik eins und drei) zu beobachten, wie sich die Werte der Statistik-, Review- und Graphensektion entsprechend der vorgenommenen Einstellungen verändern. Diese Funktion erlaubt es, Korrelationen zwischen verschiedenen Attributen eines Reviews zu erkennen. So könnte zum Beispiel aufgezeigt werden, dass die Sternebewertung oder die Einordnung als Fehlerbericht („bug report“) in einem Review mit ihrer Länge korreliert [64].

Eine weitere Funktion der Anwendung befindet sich ebenso im oberen Bereich jeder Statistik-Seite: das Suchfeld. Mit diesem lassen sich mehrere Suchbegriffe eingeben, um gezielt in den Reviews nach bestimmten Schlüsselwörtern zu suchen. In der Folge werden ausschließlich die Reviews angezeigt, welche die eingegebenen Schlüsselbegriffe enthalten. Zudem passen sich die Statistiken und Diagramme entsprechend den angezeigten Reviews an. Diese Funktion erweitert die Möglichkeiten zur gezielten Analyse und kann dazu beitragen, bestimmte Themen oder Aspekte in den Reviews effektiver zu untersuchen. Im unteren Bereich der Statistik-Seite befindet sich eine Navigationsleiste, die einen schnellen Zugriff auf die jeweiligen Review-Datenquellen oder auch einen Vergleich dieser Datenquellen ermöglicht. Diese üblicherweise als „Tab-Bar“ bezeichnete Funktion ist in den meisten modernen Software-Anwendungen vorzufinden. Sie erhöht die Nutzerfreundlichkeit, indem sie eine einfache und schnelle Navigation innerhalb der Anwendung sicherstellt (Usability-Heuristik vier).

Jede gecrawlte App bietet die Option, sie in Gänze zu entfernen (Usability-Heuristik zwei). Auf diese Weise kann der App-Analyst überflüssige Analysen effektiv bereinigen. Darüber hinaus stellt diese Funktion einen Mechanismus zur Korrektur von Eingabefehlern dar, die während der Erstellung der App-Analyse aufgetreten sind. Zusätzlich besteht die Option, die App-Analyse in eine CSV-Datei zu exportieren (Usability-Heuristik vier und sieben). Ein Vorteil hierbei ist, dass diese CSV-Datei an Mitarbeiter versendet werden kann, die sie anschließend in ihr *feelio*-Konto importieren können. Zudem kann eine CSV-Datei als vielseitiges Format auch für andere Anwendungen genutzt werden, um so eine eventuelle Weiterverarbeitung der Daten zu ermöglichen. Unabhängig von der tatsächlichen Aufbereitung der App-Analyse existiert zudem die Hauptseite „Profileigenschaften“. Auf dieser Seite kann sich der Nutzer von seinem Konto abmelden oder sogar sein gesamtes Konto inklusive aller Analysedaten löschen (Usability-Heuristik vier).

3.4 Klassifizierer

Die ausgewählten Klassifizierer sind so konzipiert, dass sie den Bedürfnissen eines App-Analysten gerecht werden. Verschiedene NLP-Techniken bilden die Basis für diese Auswahl. Dazu gehören unter anderem das Feinabstimmen eines Modells mittels des Transfer-Learning-Ansatzes, die Verwendung vordefinierter Modelle von Huggingface, Technologien wie SentiStrength und weitere lexikonbasierte Ansätze, die in der Anwendung zur Klassifizierung der Reviews genutzt werden. Zusätzlich zu den bereits vorhandenen Klassifizierern können weitere hinzugefügt werden, um den spezifischen Bedürfnissen des App-Analysten gerecht zu werden.

3.5 Plattformwahl

Die Wahl der Plattform spielt eine entscheidende Rolle dafür, wie die Anwendung im späteren Verlauf des Entwicklungszyklus genutzt werden kann. Die Anwendung *feelio* ist darauf ausgelegt, ein hohes Maß an Flexibilität zu bieten. Durch die Verwendung eines responsiven Layout-Ansatzes einer Web-App passt sich das Format der Anwendung individuell an den jeweiligen Gerätetyp an, auf dem sie ausgeführt wird. Mit dem responsiven Layout-Ansatz werden die Usability-Heuristiken vier, sieben und acht verfolgt. Die Anwendung ist sowohl für kurze Nutzungsszenarien, beispielsweise auf dem Smartphone, als auch für längere, intensive Nutzungsphasen am Desktop-PC perfekt ausgerichtet. Um einen konsistenten Datenzugriff auf mobilen und stationären Geräten zu ermöglichen, bietet die Funktion der Kontenerstellung eine Lösung. Auf diese Weise ist ein Zugriff auf die gleiche Datengrundlage stets möglich, ob zu Hause, unterwegs oder auf der Arbeit.

3.5.1 Modularität

Um die Langlebigkeit einer Anwendung in einem Arbeitsumfeld zu gewährleisten, wird ein besonderer Wert darauf gelegt, dass neue Funktionen leicht zu integrieren sind. In der heutigen Zeit ist es für eine Software unerlässlich, schnell auf Marktveränderungen reagieren zu können [11]. Dies beinhaltet auch das Implementieren der aktuell besten Methoden für NLP-Analysen, um App-Reviews auswerten zu können. Folglich ist eine Anwendung wie *feelio* von Beginn an modular konzipiert, um veraltete oder nicht mehr benötigte Klassifizierer zu entfernen und neue, „State of the Art“-Klassifizierer zu integrieren. Neben der Modularität der Klassifizierer sind auch die anderen Komponenten der Software austauschbar und erweiterbar.

Kapitel 4

Implementierung

Nach ausführlicher Planung in der Konzeptphase, folgt nun die Erläuterung der Umsetzung. Hierbei werden die verschiedenen Implementierungsvarianten dargestellt, die realisiert wurden. Im weiteren Verlauf des Prozesses werden zudem die aufgetretenen Probleme sowie deren jeweilige Lösungsansätze besprochen.

4.1 Architektur

Die Anwendung folgt in ihrer architektonischen Gestaltung, einem klassischen Server-Client-Modell. Im Inneren des Clients und des Servers befindet sich eine modulare Struktur, welche die spätere Erweiterung der Applikation erheblich erleichtert. Im Laufe des Entwicklungsprozesses wurden verschiedene Prinzipien berücksichtigt, um die Modularität zu gewährleisten. Schlüsselkonzepte, die von besonderer Relevanz sind, umfassen Kapselung [81], Abstraktion, geringe Kopplung und hohe Kohäsion [88]. Durch die Einhaltung dieser Kriterien konnte eine monolithische Herangehensweise vermieden werden. Der Client und der Server sind mittels einer Express-

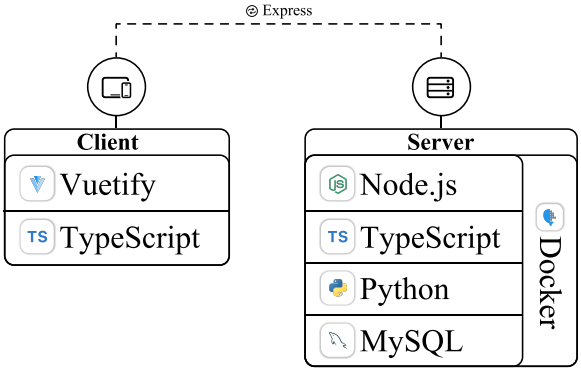


Abbildung 4.1: Die Verbindung zwischen Client und Server

Schnittstelle verbunden (siehe Abbildung 4.1). Die Entscheidung fiel auf Express [59], da es sich um einen anerkannten und weitverbreiteten Standard handelt. Außerdem lässt sich das Session-Management sowie weitere Kommunikationsparameter effizient mit dieser Schnittstelle konfigurieren. Auf der Client-Seite befindet sich das Frontend, das mithilfe von Vuetify [85] ein ansprechendes, benutzerfreundliches Design bietet, welches auf den Prinzipien des Material Designs basiert. Dabei wurde nicht die standardmäßige JavaScript-Variante von Vuetify genutzt, sondern eine Integration mit TypeScript [52] durchgeführt. TypeScript wurde ausgewählt, um eine Typisierung von Parametern zu ermöglichen. Das Anlegen eines Modells vereinfacht das Lesen und Verstehen des Codes erheblich und verdeutlicht, welcher Datentyp in welchem Prozess erwartet wird. Die Modularität der Software kann somit auch durch die Typisierung optimiert werden, da das Datenmodell auf organisierte Weise strukturiert werden kann.

Vuetify bietet die Möglichkeit, eine Applikation hierarchisch zu strukturieren. Die Gestaltung der Applikation beginnt daher mit dem Layout, welches den grundsätzlichen Aufbau einer Webseite bestimmt, beispielsweise dass die Navigationsleiste stets oben positioniert ist. Dieses Layout wird durch die Views ausgefüllt, die verschiedene Bereiche der Webseite darstellen. Zudem beinhaltet es Komponenten, die spezifische, selbstdefinierte UI-Elemente umfassen. Durch diesen Ansatz kann viel redundanter Code vermieden und der gesamte Umfang einer Applikation überschaubar gehalten werden. Der Vorteil liegt hierbei in der Ersparnis von Entwicklungsaufwand und der Gewährleistung einer klaren, gut strukturierten Benutzeroberfläche. Ein Prototyp für die Benutzeroberfläche, erstellt mit Figma [20], kann im Abschnitt A.5.2 des Anhangs eingesehen werden.

Auch serverseitig wird eine TypeScript-Variante von Node.js [60] verwendet. Dies ermöglicht sowohl auf Frontend- als auch auf Backend-Seite die Nutzung der Vorteile einer Typisierung. Folglich kann dasselbe Datenmodell in beiden Bereichen angewendet werden. Zusätzlich zur Node.js-Umgebung existieren eine Python- und eine MySQL-Umgebung [72, 61]. Diese drei Umgebungen werden mithilfe von Docker [16] verwaltet, wodurch drei unabhängige Docker-Container erstellt werden. Docker [16] wurde zur Verteilung des Servers eingesetzt, um die Portabilität der Anwendung zu erhöhen. Im Grunde genommen ist es nur notwendig, Docker zu installieren, um den eingerichteten Server auf jeder Maschine unabhängig von den spezifischen Systemeigenschaften auszuführen. Für die Persistenz der Daten kommt das weit verbreitete, robuste und bewährte relationale Datenbankmanagementsystem MySQL zum Einsatz. Die Entscheidung fiel auf eine relationale Datenbank, da eine tabellenbasierte Datenstruktur sich als vorteilhafter erwies. Die grundlegende Struktur der Daten wird sich in der nahen Zukunft nicht ändern, da eine gecrawlte App stets bestimmten Eigenschaften folgt. Im Falle von Datenbankanpassungen müssen in den Tabellen lediglich Attribute hinzugefügt oder gegebenenfalls entfernt werden. Solche Modifikationen be-

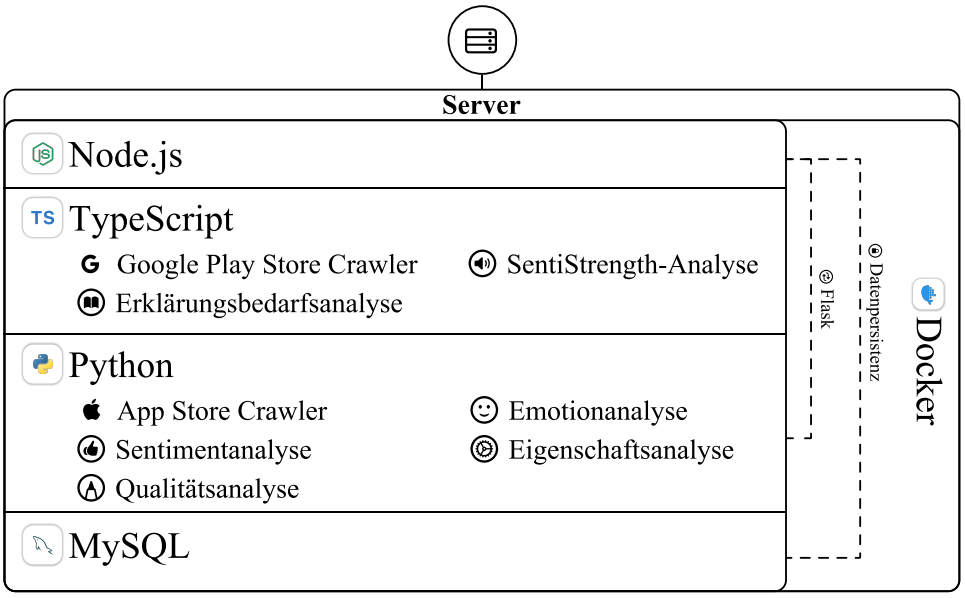


Abbildung 4.2: Übersicht der Serverkomponenten und interner Verbindungen

einflussen jedoch nicht die Gesamtstruktur der Datenbasis, weshalb sich ein relationales Datenbankmanagementsystem als passende Wahl erweist. Die Docker-Container werden mithilfe einer Docker-Compose-Datei miteinander verknüpft. Dabei übernimmt Node.js die Express-Kommunikation mit dem Client und etabliert über eine Flask-Schnittstelle [65] eine Verbindung zu einem Python-Container (siehe Abbildung 4.2).

Zusätzlich besteht von Node.js aus noch eine Verbindung zur MySQL-Datenbank, wobei zentralisierte Datenanpassungen auf dieser durchgeführt werden. Der Grund für das Vorhandensein sowohl eines vollständigen Node.js-Dienstes als auch eines Python-Dienstes liegt darin, dass verschiedene Bibliotheken unterschiedliche Sprachen erfordern. Durch diese beiden Dienste ist die Software gut auf zukünftige Anforderungen vorbereitet.

Innerhalb des durch TypeScript typisierten Node.js wurden der Google Play Store Crawler, die SentiStrength-Analyse und die Erklärungsbedarfsanalyse implementiert. Die zusätzlichen Bibliotheken, die innerhalb der Python-Umgebung zum Einsatz kommen, umfassen den Apple App Store Crawler, die Emotionsanalyse, die Sentimentanalyse, die Qualitätsanalyse und die Eigenschaftsanalyse.

4.2 Klassifizierer

4.2.1 Emotionsanalyse

Für die Klassifizierung eines Reviews in eine vordefinierte Emotion, wird das Large-Language-Modell (LLM) „Emotion-English-Distilroberta-Base“ [31] von j-hartmann verwendet. Dieses Modell ermittelt die Emotion eines Textes basierend auf Ekman's sechs Basisemotionen [17] und einer zusätzlichen neutralen Klasse: Wut („anger“), Ekel („disgust“), Angst („fear“), Freude („joy“), Neutralität („neutral“), Traurigkeit („sadness“) und Überraschung („surprise“). Das LLM basiert auf einer feinabgestimmten Variante von „DistilRoBERTa“ [30]. „DistilRoBERTa“ ist wiederum ein aus „RoBERTa“ [33] abgeleitetes sogenanntes „destilliertes“ Modell, dass das Verhalten des größeren, komplexeren Modells „RoBERTa“ zu imitieren versucht. Das Endergebnis ist ein Modell, das eine ähnliche Performance wie das ursprüngliche Modell liefert, jedoch mit dem Vorteil eines geringeren Speicherbedarfs und einer insgesamt höheren Effizienz [30]. Das LLM von j-hartmann [31] zählt zu den erfolgreichsten Modellen zur Emotionsklassifizierung von Texten auf Hugging Face (Stand 20.06.2023) und findet zahlreiche wissenschaftliche Anwendung. Butt et al. [4] nutzten das LLM unter anderem für die Erkennung von Fehlinformationen auf der Plattform Twitter, Kuang et al. [43] setzten es ein, um ein Werkzeug zur präzisen Beschreibung von Musik in Textform zu entwickeln und Rozado et al. [75] verwendeten es zur Analyse von Nachrichtenschlagzeilen, um die Veränderung der in Schlagzeilen dargestellten Emotionen über die letzten Jahrzehnte zu untersuchen.

4.2.2 Sentimentanalyse

Die Sentimentanalyse basiert auf einem selbst feinabgestimmten Modell des LLM namens „Twitter-Roberta-Base-Sentiment-Latest“ [35] von cardiffnlp.

Loureira et al. [48] von cardiffnlp beschreiben in ihrer Arbeit „TimeLMs: Diachronic Language Models from Twitter“ die Umsetzung ihres LLM. Die Sentimentanalyse liefert die Sentimentpolaritäten: positiv, neutral und negativ. Innerhalb eines Reviews wird die jeweilige Sentimentpolarität zusätzlich durch ein kleines Kreissymbol in den entsprechenden Farben grün, grau und rot dargestellt. Dies ermöglicht es dem Nutzer, auf den ersten Blick die vorherrschende Sentimentpolarität zu erfassen.

Um innerhalb eines Reviews positive und negative Ausdrücke zu identifizieren und somit die Teil-Sentiments des Reviews zu erfassen, wird SentiStrength [83] eingesetzt. Es bietet eine kontinuierliche Bewertung emotionaler Sentiments auf einer Skala von -1 (negativ) bis -5 (extrem negativ) und +1 (positiv) bis +5 (extrem positiv). Dieser kontinuierliche Sentiment-Wert wird genutzt, um die Stimmung einzelner Wörter visuell hervorzuheben. Wörter, die einen Wert größer oder gleich +1 aufweisen, erhalten eine grüne Umrandung, während Wörter, die einen Wert kleiner

oder gleich -1 aufweisen, eine rote Umrandung bekommen. Zusätzlich wird der Sentiment-Wert aller Wörter im Review summiert und eine entsprechende Polarität unterhalb des Reviews angezeigt. Hierbei bedeutet eine Sentiment-Summe kleiner null, dass das Review insgesamt eine negative Stimmung aufweist. Ein Wert größer null hingegen deutet auf eine positive Stimmung hin und ein Wert von genau null steht für eine neutrale Stimmung. Analog zur Sentimentanalyse wird ein Kreis in der jeweiligen Farbe rot für negativ, grün für positiv und grau für neutral zur visuellen Darstellung des Gesamtsentiments angezeigt.

4.2.3 Qualitätsanalyse

Für die Einteilung eines Reviews in die Qualitätskategorien „sehr schlecht“ („noise“), „schlecht“ („word salad“), „halbwegs gut“ („mild Gibberish“) und „sehr gut“ („clean“) wurde das LLM „Autonlp-Gibberish-Detector-492513457“ [32] von Madhurjindal verwendet. Dieses Modell gehört zu den am häufigsten heruntergeladenen Modellen und zählt zu den besten Klassifizierern zur Qualitätsanalyse von Texten auf Hugging Face (Stand 20.06.2023). Mit einem veröffentlichten Micro-F1-Score von 0.9736 erweist sich das Modell als besonders leistungsstark. Das LLM basiert auf einer feinabgestimmten Variante von „DistilBERT“ [76], einem „destillierten“ Modell, das von „BERT“ [12] abgeleitet wurde. Die Kennzeichnung eines Reviews erfolgt sowohl durch die Qualitätskategorie in Textform als auch durch ein quadratisches Icon, welches je nach Qualitätskategorie einen Farbton des Farbverlaufs von Grün bis Rot anzeigt. Das quadratische Icon dient zur Abgrenzung von der Sentimentanalyse, die ein kreisförmiges Icon als Kennzeichnung verwendet. Hierdurch wird dem Nutzer eine eindeutige Unterscheidung ermöglicht.

4.2.4 Erklärungsbedarfsanalyse

Die mithilfe des „string matching“ durchgeführte Analyse des Erklärungsbedarfs ermöglicht eine Vorhersage darüber, ob ein Review weiterführende Erläuterungen benötigt. Hierfür wurden spezifische Schlüsselwörter ermittelt, die mit hoher Wahrscheinlichkeit in einem Review vorkommen, das weiterer Erläuterung bedarf (siehe Abbildung 4.3). Die Grundidee des „string matching“ liegt darin, die Anwesenheit von bestimmten Wörtern aus einer definierten Liste von Schlüsselbegriffen als Indikator für einen Erklärungsbedarf zu sehen. Gefundene Schlüsselbegriffe innerhalb des Reviews werden mit einer blauen Umrandung markiert, um potenziellen Erklärungsbedarf auf den ersten Blick zu erkennen. Zudem wird jedes Review, in der ein solcher Schlüsselbegriff gefunden wird, markiert, um zu signalisieren, dass es Erklärungsbedarf enthält. Diese Markierung erfolgt mittels eines kleinen blauen Kreises und der Beschriftung „Explanation Needed“. Die Farbe Blau

sticht unter den anderen verwendeten Farben hervor und wird ausschließlich für diesen Klassifizierer verwendet, um die Orientierung der Nutzer zu erleichtern.

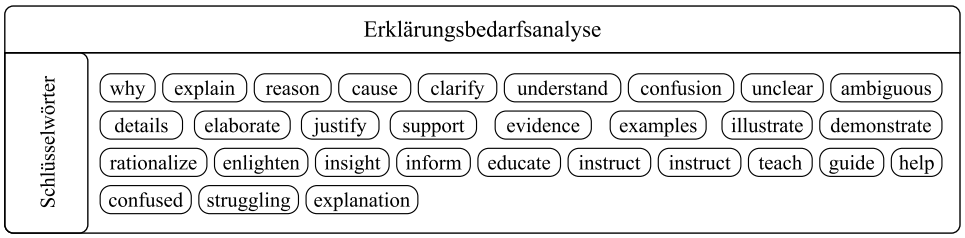


Abbildung 4.3: Liste der Schlüsselwörter, die für die Methode des „string matching“ im Rahmen der Erklärungsbedarfsanalyse verwendet wurden

4.2.5 Eigenschaftsanalyse

Die Eigenschaftsanalyse dient zur Bestimmung der Charakteristiken eines Reviews. Hierbei wird das LLM „Bart-Large-Mnli“ [29] von Hugging Face eingesetzt, welches ursprünglich von Meta, ehemals Facebook, entwickelt wurde. Es ist das am häufigsten heruntergeladene Modell in der Kategorie „Zero-Shot Classification“ auf der Plattform Hugging Face (Stand 20.06.2023). Das LLM ermöglicht in dem Anwendungsfall dieser Arbeit eine gezielte Kategorisierung von Reviews basierend auf spezifischen Charakteristika. Im Rahmen der Review-Analysen standen folgende Merkmale im Fokus: Nutzererfahrung („user experience“), Funktionsanforderung („feature request“), Textbewertung („text rating“) und Fehlerbericht („bug report“). Als Inspirationsquelle für die Auswahl dieser Merkmale diente die wissenschaftliche Arbeit von Maalej et al. [49], welche eine umfassende Übersicht über potenzielle Charakteristika eines Reviews bietet. Es ist wichtig zu berücksichtigen, dass ein Review mehrere Merkmale gleichzeitig aufweisen kann. So kann es beispielsweise sowohl eine Funktionsanforderung als auch einen Fehlerbericht beinhalten. In diesem Zusammenhang werden jedoch nur jene Merkmale in die Eigenschaftklassifizierung einbezogen, die eine Übereinstimmungswahrscheinlichkeit von über 50% aufweisen.

4.3 Crawler

Die Arbeit fokussiert sich auf die Analyse von englischsprachigen App-Reviews. Stand 2023 dominieren bezüglich des App-Volumens der Google Play Store und der Apple App Store deutlich den Markt. Der Amazon App Store stellt im Vergleich dazu nur eine Randgruppe dar, was die Distribution von Apps betrifft [1]. Aufgrund dieser Gegebenheiten wurde entschieden, den Google Play Store und den Apple App Store als Datenquellen für die

Review-Analyse zu nutzen. Um Zugriff auf die Review-Daten der App Stores zu erhalten, wurden der „app-store-scraper“ [10] für den Apple App Store und der „google-play-scraper“ [19] für den Google Play Store eingesetzt. Für das Crawlen der Apple App Store Reviews wurde die Python-Variante gewählt, da die Node.js-Version [18] die Einschränkung aufweist, nur Reviews laden zu können, die auf maximal zehn Review-Seiten erscheinen. Diese Limitierung besteht in der Python-Version nicht, wodurch ein grenzenloses Crawlen von Reviews ermöglicht wird. Beim Google Play Store wurde die Node.js-Variante bevorzugt, da sie einwandfrei und effizient funktioniert. Sowohl der Crawler für den Apple App Store als auch für den Google Play Store haben die Fähigkeit, neben den App-Reviews zusätzliche Informationen über die gecrawlten Apps bereitzustellen. Insbesondere wurden der originale App-Name, das Genre, das Entwicklerstudio samt der Entwickler-URL, die Sternebewertung, der Preis und die Bildschirmaufnahmen der App von den Crawlern verwendet und in der Benutzeroberfläche dargestellt. Diese zusätzlichen Informationen bieten dem App-Analysten eine gute Übersicht und helfen ihm dabei, die App einzuordnen. Beispielsweise ermöglicht ein Klick auf das Entwicklerstudio eine direkte Weiterleitung zur Unternehmensseite des Entwicklerstudios, um dort weitere Informationen über den Entwickler einzuholen.

4.4 Feinabstimmung eines Datenmodells

Für die Sentimentanalyse wurde das Verfahren des Transfer-Learning angewendet, bei dem ein bereits bestehendes Modell zusätzlich auf einer spezifischen Datenbasis, in diesem Fall der Datenbasis für App-Reviews, weiter optimiert wird, um bessere Vorhersagen als das ursprüngliche Modell zu erzielen. Als Ausgangspunkt diente das LLM „Twitter-Roberta-Base-Sentiment-Latest“ [35] von cardiffnlp. Zahlreiche LLMs, wie beispielsweise das RoBERTa-Modell, stützen sich auf Daten, die mehrere Jahre zurückliegen [48]. Gesellschaftlich prägende Ereignisse, wie beispielsweise die COVID-19-Pandemie, sind in älteren LLMs nicht berücksichtigt, so auch beim RoBERTa-Modell. Die Entscheidung fiel auf das Modell „Twitter-Roberta-Base-Sentiment-Latest“, da es sich um ein äußerst modernes Modell handelt, das kontinuierlich mit aktuellen Tweets trainiert wird, um stets auf einer aktuellen Datenbasis zu basieren und so adäquate, dem Zeitgeist entsprechende Vorhersagen liefern zu können. Zudem gehört das Modell zu den beliebtesten Modellen für die Klassifizierung der Sentimentpolaritäten: positiv, neutral und negativ (Stand 20.06.2023).

4.4.1 Erstellen des Datensatzes

Bevor das ursprüngliche Modell spezifisch auf das Gebiet der App-Reviews zugeschnitten werden konnte, war es notwendig, einen entsprechenden

Datensatz zu erheben. Dieser Datensatz, der für das Training des Modells genutzt wurde, musste Reviews enthalten, die bereits mit den Sentimentpolaritäten positiv, neutral und negativ annotiert waren. Im Internet konnte kein geeigneter öffentlicher Datensatz gefunden werden, bis auf einen von Lin et al. [46]. Dieser erwies sich jedoch wegen seiner geringen Größe von 341 annotierten App-Reviews und der unausgewogenen Verteilung der Sentimentpolaritäten (nur 7% neutrale Annotationen) als ungeeignet für das Training eines Modells. Daher war es notwendig, einen eigenen Datensatz zu erstellen.

Datengrundlage des Datensatzes

Für die Erstellung des zu annotierenden Datensatzes wurde eine vielfältige Auswahl an Review-Daten angestrebt. Dabei waren mehrere selbstgesetzte Bedingungen zu berücksichtigen. Die Reviews sollten gleichmäßig vom Google Play Store und vom Apple App Store stammen, um die unterschiedlichen Nutzergruppen beider Plattformen im Training zu berücksichtigen [49]. Darüber hinaus war es wichtig, dass sie aus verschiedenen App-Genres entnommen wurden und in gleichen Anteilen von sowohl bezahlten als auch kostenlosen Apps kamen. Durch diese Vorgehensweise besteht die Aussicht, fundierte Vorhersagen über zahlreiche App-Genres und Preiskategorien hinweg zu erhalten. Von Vorteil ist außerdem, dass die Reviews bereits einer vorläufigen Klassifizierung hinsichtlich ihrer Sentiment-Polarität unterzogen wurden. Dies soll eine ausgewogenere Verteilung von positiven, neutralen und negativen Einträgen als Datenbasis gewährleisten, um einen ausgeglichenen Datensatz für das Training zu erhalten. Angesichts der genannten Bedingungen wurden in beiden App Stores die sechs populärsten Genres ermittelt (siehe Abbildung 4.4). Innerhalb jedes Genres wurden sowohl

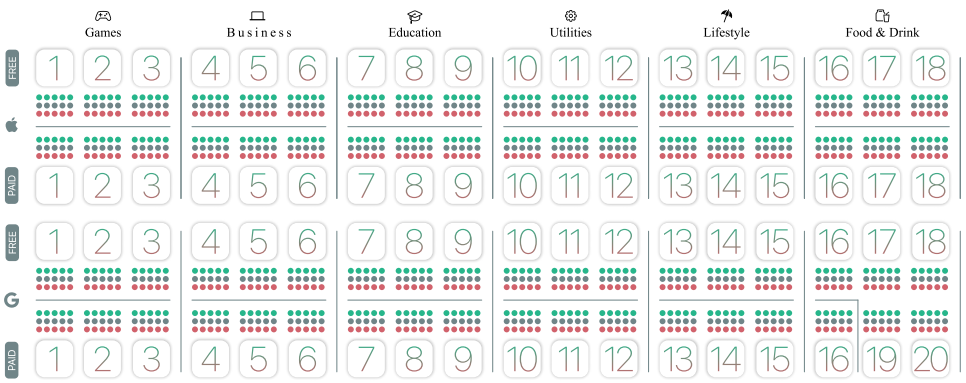


Abbildung 4.4: Visualisierung der Review-Datensatzerhebung; grüne, graue und rote Kreise repräsentieren die vorläufige Klassifizierung der Sentiments für alle Reviews der betreffenden Apps

die drei meistgeladenen kostenlosen Apps als auch die drei populärsten kostenpflichtigen Apps ausgewählt. Für jede App wurden 15 Reviews gesammelt, jeweils fünf, die als positiv, neutral und negativ von dem Ausgangsmodell [35] eingestuft wurden. Die zur Klassifizierung verwendeten Reviews waren stets die neuesten und stammten aus keiner Zeit vor dem 01.01.2020. Im Google Play Store konnte jedoch für das Genre „Food & Drink“ leider nur eine App gefunden werden, die den Anforderungen von fünf Reviews jeder Sentimentpolarität entsprach. Dafür wurden zwei zusätzliche kostenlose Apps des Genres „Food & Drink“ in die Sammlung aufgenommen. Die geringe Anzahl an Reviews für kostenpflichtige Apps im Bereich „Food & Drink“ ist auf die Dominanz der kostenlosen Alternativen zurückzuführen. Große Fast-Food-Ketten wie McDonald's oder DoorDash verfügen über eine breite Kundenbasis und dominieren die Downloadzahlen, wodurch das Genre „Food & Drink“ zu einem der beliebtesten im Google Play Store wird, obwohl die kostenpflichtigen Optionen des Genres wenig Downloads und somit wenige Reviews aufweisen. Dennoch wurde eine vielfältige Datenbasis erstellt. So resultierte eine möglichst gleichverteilte Anzahl von 1080 zu annotierenden Reviews aus beiden App Stores als Datenbasis zum Stichtag 25.05.2023.

Annotation des Datensatzes

Nachdem die Datenbasis erstellt wurde, begann die Phase der Annotation dieser Reviews. Elf Fragebögen mit jeweils etwa 100 zu annotierenden Reviews wurden mithilfe der Plattform Google Forms erstellt. Die Probanden wurden dazu aufgefordert, jedes Review, bzw. jede Frage, hinsichtlich ihrer Sentimentpolarität zu klassifizieren. Die Bearbeitung eines Fragebogens nahm etwa eine Stunde in Anspruch und wurde von elf Informatikstudierenden der Leibniz Universität Hannover durchgeführt. Die Wahl fiel auf Informatikstudierende als Probanden, da sie ein technisches Verständnis im Kontext der Softwareentwicklung mitbringen. Dies ermöglicht ihnen, ein Review besser im Sinne eines Softwareentwicklers oder eines App-Analysten zu bewerten, die letztendlich die Anwendung verwenden werden.

4.4.2 Qualitätskontrolle der Annotation

Um sicherzustellen, dass die Probanden bewusste Entscheidungen bezüglich der Annotation getroffen haben und nicht aus Unsicherheit willkürlich annotiert haben, war es notwendig, eine zusätzliche Nachbefragung für jeden teilnehmenden Probanden durchzuführen. Des Weiteren konnte durch die Umfrage bestätigt werden, dass die Probanden das Sentiment sowohl anhand inhaltlicher Kriterien als auch nach der Ausdrucksweise beurteilt haben. Wie in Abbildung 4.5 dargestellt, hat der Großteil der Probanden beide Kriterien der Sentimentanalyse, die in den Grundlagen (siehe Abschnitt 2.5.1) beschrieben wurden, berücksichtigt. Abbildung 4.6 bestätigt, dass der Großteil der Probanden sich bei der Annotation sicher fühlte. Zwei Probanden

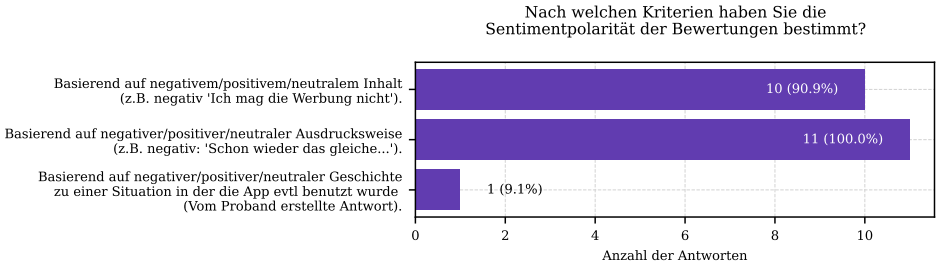


Abbildung 4.5: Nachfrage zur Grundlage der gegebenen Antworten

gaben eine neutrale Antwort an und ein Proband äußerte Unsicherheit. Daraus kann gefolgert werden, dass die Annotation bei jeden der Probanden gewisse Unsicherheiten hervorrief, die meisten jedoch zuversichtlich bezüglich ihrer Annotation waren. Zudem wurde überprüft, ob die Probanden die

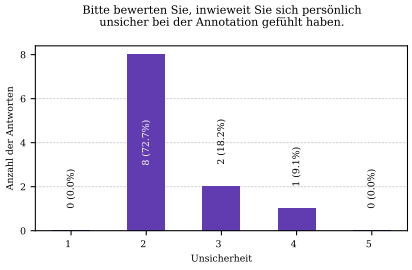


Abbildung 4.6: 1: überhaupt keine Unsicherheiten, 5: sehr große Unsicherheiten

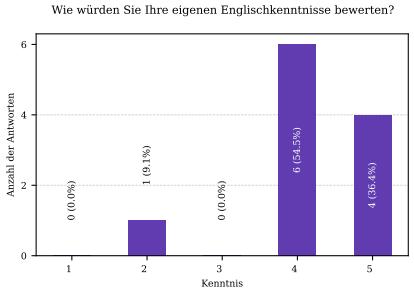


Abbildung 4.7: 1: sehr geringe Kenntnisse, 5: sehr gute Kenntnisse

Reviews auf Englisch problemlos verstehen konnten. Abbildung 4.7 zeigt, dass lediglich ein Proband über eingeschränkte Englischkenntnisse verfügte, während die übrigen ihre Englischkenntnisse als gut bis sehr gut einschätzten.

Abschließend wurde untersucht, wie eventuelle Unsicherheiten entstanden sind (siehe Abbildung 4.8). Die Probanden äußerten, dass die Analyse

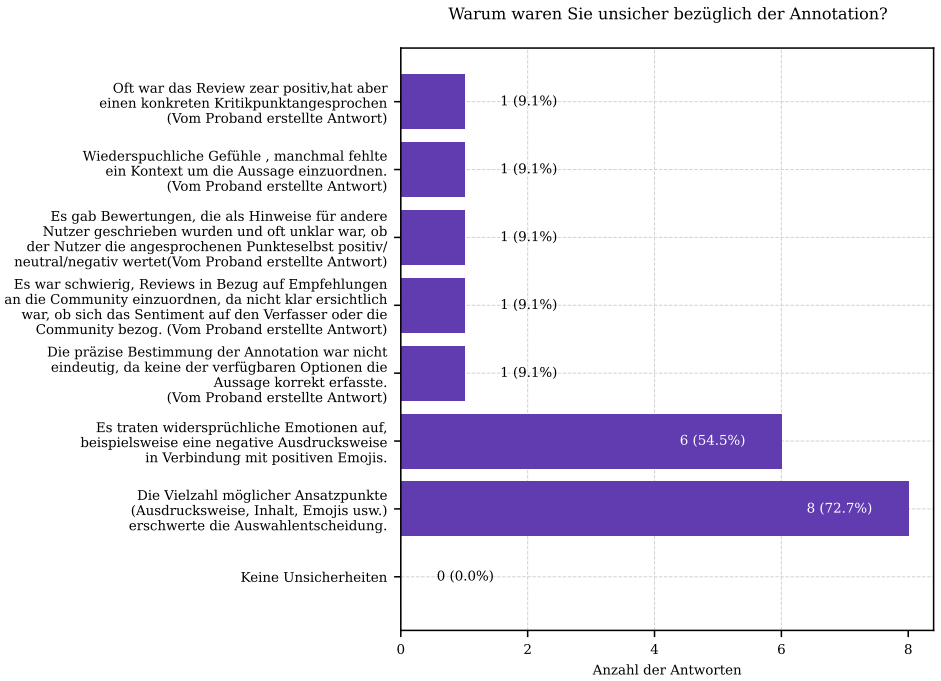


Abbildung 4.8: Nachfrage zur Unsicherheit der gegebenen Antworten

eines Reviews komplex ist, da es viele Aspekte gibt, die bei einem Review berücksichtigt werden müssen. Zudem berichteten zwei Probanden, dass es herausfordernd war, das Sentiment entweder dem Autor oder der Community zuzuordnen, insbesondere wenn in einigen Reviews Handlungsempfehlungen an die Community gerichtet waren.

Auswertung der Annotation

Die elf ausgefüllten Fragebögen wurden jeweils in eine CSV-Datei exportiert. Anschließend wurden diese Daten in ein Hugging Face „Dataset“[1] konvertiert. Dieses umfasst die Spalten „text“ für den Inhalt der Rezension, „label“ für die annotierte Sentimentpolarität und „group“ zur Kennzeichnung der Fragebogenzugehörigkeit.

[1] `https://timokurtz.de/bachelor/appReviews.html`

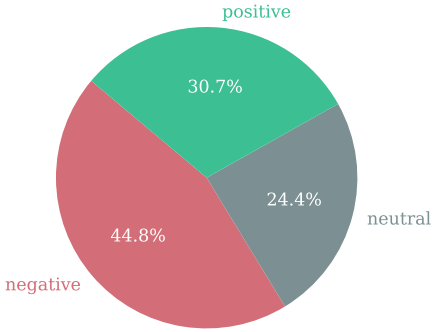


Abbildung 4.9: Verteilung der Sentimentpolaritäten in den aus der Umfrage resultierenden annotierten Daten

Es stellte sich heraus, dass die ausgewertete Annotation ein Ungleichgewicht bei der manuellen Zuordnung der Sentimentpolaritäten aufweist (siehe Abbildung 4.9). Die negative Sentimentpolarität dominierte in den annotierten Daten. Da eine gleichmäßige Verteilung der Trainingsdaten wichtig ist, wurde zusätzlich die Methode des Oversamplings angewendet. Das führte zur Erstellung eines weiteren Hugging Face „Dataset“[2] , in dem zufällig ausgewählte positive und neutrale Daten dupliziert wurden, um die gleiche Anzahl an Datenbeständen zu erreichen, wie es bei der am häufigsten annotierten Polarität, nämlich der negativen, der Fall war.

4.4.3 Training des Modells

Für das Training des Ausgangsmodells auf ein spezifisches Teilgebiet (auf Englisch als “downstream task“ bezeichnet), wird die bekannte und weit verbreitete Open-Source-Bibliothek Transformers [34] von Hugging Face verwendet.

Drei unterschiedliche Modelle[3] wurden trainiert, um eine Evaluation durchführen zu können und zu ermitteln, welches Modell in den Metriken besser abschneidet. Das erste Modell wurde auf Basis eines „oversampled“ Datensatzes trainiert. Sowohl das zweite als auch das dritte Modell wurden unter Verwendung des ursprünglichen, unveränderten Datensatzes trainiert. Dabei wurde beim zweiten Modell die „Loss Function“ modifiziert, um das Ungleichgewicht der annotierten Daten auszugleichen. Das Cross-Validation-Verfahren „StratifiedGroupKFold“ [78] mit k = 10 wurde für jedes Modell angewendet, um eine optimale Verteilung des Trainings- und Testdatensatzes zu erzielen.

[2] `https://timokurtz.de/bachelor/appReviewsOversampled.html`
[3] `https://timokurtz.de/bachelor/finetunedModels.html`

Die jeweiligen optimal verteilten Trainings- und Testdatensätze wurden ebenfalls als Hugging Face „Dataset“[4] gespeichert. StratifiedGroupKFold kombiniert die Methoden „StratifiedKFold“ [79] und „GroupKFold“ [77]. Das k in „StratifiedKFold“ oder „GroupKFold“ bedeutet, dass der annotierte Datensatz k-mal („k-Folds“) hintereinander in variierenden Kombinationen aufgeteilt wird. „StratifiedKFold“ gewährleistet, dass bei der Unterteilung des Datensatzes in jeweilige Trainings- und Testdatensätze pro Fold eine gleichmäßige Verteilung der annotierten Daten vorliegt. „GroupKFold“ wiederum stellt sicher, dass das Modell pro „Fold“ nicht von derselben Gruppe evaluiert wird, die sich bereits im Trainingsdatensatz befindet.

Eine Übersicht der einzelnen Schritte zur Feinabstimmung des Modells für die Sentimentanalyse ist in Abbildung 4.10 dargestellt.

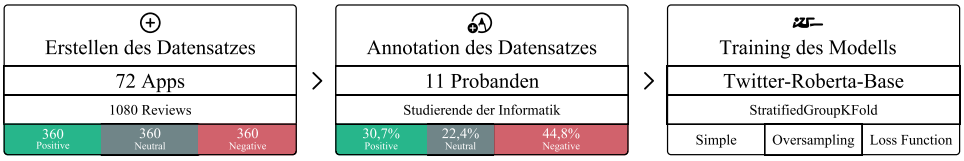


Abbildung 4.10: Darstellung des Feinabstimmungsprozesses für die Sentimentanalyse

4.5 Methoden der Softwareprüfung

Um einen kontinuierlichen Qualitätsstandard der Software sicherzustellen, sind regelmäßige Software-Tests unerlässlich [8]. Der erste Schritt bestand in der Umsetzung von Use-Case-Tests, welche die Kernfunktionen der Software abdecken und zur Kontrolle dienen, ob die Anforderungen des Kunden erfolgreich realisiert wurden. Im Laufe der Softwareentwicklung wurden die bestehenden Use-Case-Tests kontinuierlich mit dem aktuellen Softwarestand abgeglichen und bei Bedarf Anpassungen an der Software vorgenommen. Die Überprüfung der Use-Case-Tests erfolgte durch die Interaktion mit der Benutzeroberfläche und die Umsetzung der spezifischen Anforderungen aus den Use-Case-Tests. Allerdings ermöglicht diese Methode des manuellen Testens keinen automatisierten Testablauf.

4.5.1 Postman

Für die automatisierte Überprüfung der erstellten API der Applikation *feelio* wurde die Testsoftware Postman [69] eingesetzt. Die Entscheidung zugunsten von Postman wurde aufgrund seiner umfangreichen Auswahl an Testmöglichkeiten und Beliebtheit bei renommierten IT-Unternehmen wie LinkedIn, Meta und Slack getroffen. Innerhalb von Postman wurden

[4] `https://timokurtz.de/bachelor/optimalTrainingsTestSets.html`

sämtliche HTTP-Anfragen, die der Server verarbeitet, nachgebildet. Darüber hinaus wurden die Use-Case-Tests in Postman integriert, indem Funktionsabläufe, sogenannte „Testsuites“, von HTTP-Anfragen erstellt wurden.

Managementtauglichkeit

Ein weiterer Vorteil des Einsatzes der Testsoftware Postman besteht darin, dass die Verständlichkeit von Testfällen auch für Personen mit grundlegenden Programmierkenntnissen gegeben ist. Oftmals bevorzugen Manager lesbaren Code, der auch ohne Kenntnisse einer Programmiersprache verständlich ist. Postman setzt standardmäßig die JavaScript-Bibliothek „chai.js“ [7] ein, die sogenannte Kettenmethoden bereitstellt (siehe Listing 4.1).

```
var expect = require('chai').expect
  , foo = 'bar'
  , beverages = { tea: [ 'chai', 'matcha', 'oolong' ] };

expect(foo).to.be.a('string');
expect(foo).to.equal('bar');
expect(foo).to.have.lengthOf(3);
expect(beverages).to.have.property('tea').with.lengthOf(3);
```

Listing 4.1: Beispiele von Kettenmethoden in chai.js

Solche Kettenmethoden ermöglichen Entwicklern das Schreiben von Code, der natürlicher Sprache sehr ähnelt und somit sowohl lesbarer als auch verständlicher wird.

Kapitel 5

Evaluation

Das nachfolgende Kapitel dient der Auswertung der implementierten Software. Zunächst wird in Abschnitt 5.1 die Performance der Klassifizierer bewertet. Einschränkungen und Risiken hinsichtlich der Validität der Software werden in Abschnitt 5.2 erörtert. Abschließend findet eine zusammenfassende Betrachtung in Abschnitt 5.3 statt.

5.1 Performance

Für die Bewertung der Leistung der Klassifizierer werden die Metriken Accuracy, F1-Score, Precision, Recall und deren Durchschnittstypen Micro und Macro für Multiklassenprobleme verwendet, wie sie im Abschnitt 2.6 beschrieben sind. Als Ausgangspunkt dient jeweils die ermittelte Konfusionsmatrix des entsprechenden Klassifizierers. Die Performance jedes Klassifizierers wird stets mit seiner eigenen Baseline verglichen.

5.1.1 Feinabgestimmtes Sentimentanalysemodell

Zunächst wird das eigens feinabgestimmte Modell betrachtet, das speziell für die Sentimentanalyse entwickelt wurde. Tabelle 5.1 gibt einen Überblick über die Metriken der einzelnen Modellvarianten. Dabei dient das Ursprungsmodell „Twitter-Roberta-Base“ als Baseline, die von den individuell angepassten Modellvarianten: „Standard“, „Loss Function“ und „Oversampling“, in allen Metriken übertroffen wird. Die besten Werte innerhalb einer Metrik sind jeweils grau markiert. Es zeigt sich, dass das feinabgestimmte Modell, das ohne zusätzliche Modifikationen auskommt, im Vergleich zu den anderen feinabgestimmten Modellvarianten („Loss Function“ und „Oversampling“) in allen Metriken das schlechteste Ergebnis erzielt. Das feinabgestimmte Modell, bei welchem eine Anpassung der „Loss Function“ vorgenommen wurde, erzielt für den Durchschnittstyp Micro die besten Ergebnisse.

Metrik		Standard	Loss Function	Oversampling	Twitter-Roberta-Base
Micro	F1-Score	0.8416	0.8515	0.8235	0.7228
	Precision	0.8416	0.8515	0.8235	0.7228
	Recall	0.8416	0.8515	0.8235	0.7228
Macro	F1-Score	0.8138	0.8242	0.8267	0.6922
	Precision	0.8105	0.8218	0.8245	0.6909
	Recall	0.8211	0.8284	0.8297	0.7057
Accuracy		0.8416	0.8515	0.8235	0.7228

Tabelle 5.1: Feinabgestimmtes Sentimentanalysemodell und dessen Ausgangsmodell

Im Gegenzug erreicht die „Oversampling“-Variante des feinabgestimmten Modells die höchsten Werte im Durchschnittstyp Macro. Durch die Anwendung des „Oversampling“-Verfahrens wird sichergestellt, dass Reviews jeder Sentimentpolarität in gleicher Anzahl vorliegen. Folglich ist der Macro-Score für die „Oversampling“-Variante höher als beim „Standard“-Modell und der „Loss Function“-Anpassung. Dies lässt sich dadurch erklären, dass bei der generellen Bestimmung des Macro-Scores jede annotierte Kategorie die gleiche Gewichtung erhält. Weitere Einblicke, warum die Varianten „Loss Function“ und „Oversampling“ bei den jeweiligen Durchschnittstypen konträre Ergebnisse erzielen, liefert Tabelle 5.2. Aus der Abbildung 4.9 im vorherigen Kapitel geht hervor, dass negativ annotierte Reviews mit 44,8% einen signifikanten Anteil einnehmen. Dagegen sind positiv annotierte Reviews mit einem Anteil von 30,4% etwas seltener vertreten. Besonders auffällig ist die starke Unterrepräsentation neutraler Reviews, die lediglich 24,4% ausmachen. Dies stellt eine deutliche Diskrepanz im Vergleich zu den negativ bewerteten Reviews dar. Die in Tabelle 5.2 aufgeführten Ergebnisse illustrieren eindrücklich die ungleiche Verteilung der Sentiment-Polaritäten. Es zeigt sich, dass das Modell, in der die „Loss Function“ angepasst wurde, in allen Metriken die Vorhersage der positiven Sentimentpolarität deutlich besser treffen kann als die alternativen Modelle. Insbesondere die Bestimmung der positiven Sentimentpolarität hat sich innerhalb der Metrik Precision erheblich verbessert (um mehr als zwei Werte im Vergleich zum Standardmodell). Die Verbesserungen bei der Vorhersage der neutralen Sentimentpolarität fallen beim „Loss Function“-Modell im Vergleich zum „Standard“-Modell nur geringfügig aus. Dieser Sachverhalt resultiert aus lediglich geringfügigen Anpassungen der Hyperparameter.

Metrik		Standard	Loss Function	Oversampling	Twitter-Roberta-Base
F1-Score	Positive	0.9167	0.9296	0.9143	0.8421
	Neutral	0.6512	0.6667	0.7436	0.5116
	Negative	0.8736	0.8764	0.8222	0.7229
Precision	Positive	0.8684	0.8919	0.8889	0.7619
	Neutral	0.6364	0.6667	0.7436	0.5000
	Negative	0.9268	0.9070	0.8409	0.8108
Recall	Positive	0.9706	0.9706	0.9412	0.9412
	Neutral	0.6667	0.6667	0.7436	0.5238
	Negative	0.8261	0.8478	0.8043	0.6522

Tabelle 5.2: Attribute des feinabgestimmten Sentimentanalysemodells und dessen Ausgangsmodell

Die Bestimmung der negativen Sentimentpolarität konnte bei der „Loss Function“-Anpassung, besonders in Bezug auf die Metrik des Recall, im Vergleich zum „Standard“-Modell verbessert werden. Bei der Anwendung der „Oversampling“-Methode wird deutlich, dass sie in allen Metriken bei der Bestimmung der neutralen Sentimentpolarität den anderen Modellen überlegen ist. Jedoch hat die „Oversampling“-Methode eine nachteilige Auswirkung auf die Fähigkeit zur Vorhersage sowohl der positiven als auch der negativen Sentimentpolarität. Wie es sich bei den feinabgestimmten Modellen schon gezeigt hat, weist das Ausgangsmodell ebenfalls die niedrigste Leistung für die Bestimmung der neutralen Sentimentpolarität auf. Somit hat bereits das Ausgangsmodell Schwierigkeiten bei der Bestimmung der neutralen Sentimentpolarität. Gemäß den Erkenntnissen aus den annotierten Reviews überwiegen die Anteile an positiven und negativen Reviews gegenüber den neutralen (siehe Abbildung 4.9). Daher erweist sich eine Gewichtung einer Polarität als vorteilhaft, und es bietet sich an, die Metriken unter Berücksichtigung des Durchschnittstyps Micro zu betrachten. Mit diesem Durchschnittstyp erzielt das feinabgestimmte Modell, bei dem eine Anpassung der „Loss Function“ vorgenommen wurde, die besten Ergebnisse und erreicht ebenfalls die höchste Accuracy. Aus diesem Grund wird das Modell mit der „Loss Function“-Anpassung in der Anwendung *feelio* eingesetzt.

Zusammenfassend lässt sich festhalten, dass trotz der 1080 annotierten Reviews weitere Daten erforderlich sind, um das Modell besser an die Domäne der App-Reviews anpassen zu können. Die Aufbereitung der Umfragedaten zur Gewährleistung eines möglichst ausgeglichenen Datensatzes (gleiche Anzahl von Daten für jede Sentimentpolarität) für das Training des Modells war leider nicht ausreichend. Obwohl das Ausgangsmodell die ungleiche Verteilung deutlich reduzieren konnte, war es nicht in der Lage, eine

annähernde Gleichverteilung im Vergleich zum „Ground Truth“ zu erreichen. Es hätte hier mehr annotierte Daten bedurft, einerseits, um eine ausgewogene Datenbasis mit einer hinreichenden Gesamtzahl an annotierten Daten zu schaffen und andererseits, um zusätzliche Sonderfälle für die Erkennung der Sentimentpolaritäten berücksichtigen zu können. Ein weiteres mögliches Problem könnte jedoch in der Art und Weise der Datenannotation liegen, wie in Abschnitt 5.2 detaillierter besprochen wird.

5.1.2 Stichprobenanalyse

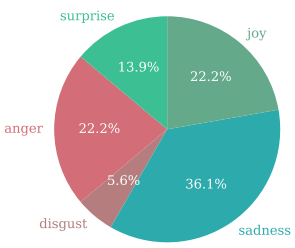
Für die Performancebewertung aller eingesetzten Klassifizierer wurde eine Stichprobe aus eigens annotierten Reviews erstellt. Diese Stichprobe umfasst 36 Reviews der beliebtesten Apps aus den drei populärsten Genres beider App Store Varianten. Um trotz der vergleichsweise kleinen Stichprobe eine möglichst vielfältige Zusammenstellung von Reviews zu gewährleisten, wurde für jede kostenpflichtige und kostenlose App-Varinate jeweils ein Review mit weniger als drei Sternen, eine mit genau drei Sternen und eine mit mehr als drei Sternen ausgewählt. Um zusätzlich aussagekräftigere Reviews zu erhalten, wurde eine Mindestlänge von 190 Zeichen festgelegt. So wurden Reviews, die beispielsweise nur einfache Phrasen wie „good“ enthalten, aussortiert. Die ausgewählten Reviews waren stets die neuesten und stammten nicht aus einer Zeit vor dem 01.01.2020. Die Stichprobe umfasste folglich zwölf Apps mit jeweils drei Reviews, wodurch sich insgesamt 36 Reviews ergaben, die von dem Autor dieser Arbeit, hinsichtlich aller Klassifizierungsmerkmale, annotiert wurden. Die Tabelle 5.3 zeigt die evaluierten Metriken der Modellergebnisse im Vergleich zur Baseline für die verschiedenen Klassifizierer. Der jeweils beste Metrik-Wert aus den Modellergebnissen und der Baseline ist in grauer Farbe hervorgehoben.

Metrik			Emotion	Sentiment	SentiStrength	Gibberish	Explanation Needed	Characteristic
Modell	Micro	F1-Score	0.7778	0.8889	0.3889	0.8333	0.8056	0.8774
		Precision	0.7778	0.8889	0.3889	0.8333	0.8056	0.8293
		Recall	0.7778	0.8889	0.3889	0.8333	0.8056	0.9315
	Macro	F1-Score	0.7663	0.8834	0.3697	0.7778	0.7979	0.8160
		Precision	0.6652	0.8974	0.4838	0.7500	0.7938	0.7665
		Recall	0.8288	0.8968	0.5026	0.9000	0.8144	0.8880
	Accuracy		0.7778	0.8889	0.3889	0.8333	0.8056	0.5556
Baseline	Micro	F1-Score	0.1111	0.5833	0.5833	0.8333	0.6389	0.4559
		Precision	0.1111	0.5833	0.5833	0.8333	0.6389	0.4921
		Recall	0.1111	0.5833	0.5833	0.8333	0.6389	0.4247
	Macro	F1-Score	0.1670	0.7368	0.7368	0.9091	0.7797	0.4382
		Precision	0.1534	0.5833	0.5833	0.8333	0.6389	0.5006
		Recall	0.0808	0.3333	0.3333	0.5000	0.5000	0.4521
	Accuracy		0.1111	0.5833	0.5833	0.8333	0.6389	0.1111

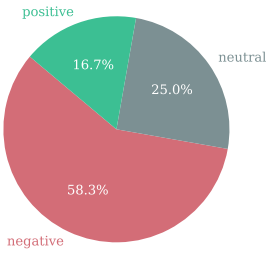
Tabelle 5.3: Auswertung der Stichprobe im Hinblick auf den Vergleich der Modellergebnisse mit der Baseline

Emotionsanalyse

Die Baseline der Emotionsanalyse beruht auf der zufälligen Zuordnung einer Emotion. Die Ergebnisse des Klassifizierers für die Emotionsanalyse übertreffen deutlich alle Metrik-Werte der trivialen Baseline. Alle Metrikwerte der Modellergebnisse übersteigen den Wert von 0,7663, lediglich die Macro-Precision bildet mit 0,6652 eine Ausnahme und stellt den schlechtesten Metrik-Wert dar. Das schlechtere Ergebnis im Vergleich zum Micro-Score könnte auf die ungleichmäßige Gewichtung der Kategorien in der Annotation der Daten zurückzuführen sein (siehe Abbildung 5.1a). Die Tabelle 5.4 mit der ausführlichen Aufschlüsselung der Metriken für jedes Emotionsattribut zeigt, dass die drei besten Precision-Scores der insgesamt sechs Attribute der Emotionsanalyse („anger“, „joy“ und „surprise“) allesamt einen Wert von über 0,8333 aufweisen und einen Gesamtanteil von 58,3% der Reviews aus der Stichprobe bilden. Attribute, die häufig in der Stichprobe vertreten sind, weisen demzufolge bessere Precision-Scores auf, was zu einer besseren Micro-Precision im Vergleich zur Macro-Precision führt. Die höchsten F1-Scores werden durch die Attribute „joy“ und „surprise“ repräsentiert, die über 0,9 liegen. Im Gegensatz dazu zeigen „anger“, „disgust“ und „sadness“ niedrigere F1-Scores. Dies könnte darauf zurückzuführen sein, dass die Unterscheidung zwischen den Gruppen „anger“, „disgust“ und „sadness“ sehr komplex und möglicherweise auch subjektiv sein kann. Der Bindestrich bei „fear“ signalisiert, dass dieses Attribut im „Ground Truth“ der Stichprobe nicht aufgetreten ist und demzufolge nicht in der Auswertung der Metriken berücksichtigt wird.



(a) Verteilung der Emotionsattribute in den aus der Stichprobe resultierenden annotierten Daten



(b) Verteilung der Sentimentpolaritäten in den aus der Stichprobe resultierenden annotierten Daten

Abbildung 5.1: Attributverteilung innerhalb des „Ground Truth“ der Stichprobe, insbesondere im Kontext der Emotionsanalyse und der Sentimentanalyse

Metrik		anger	disgust	fear	sadness	joy	surprise
Modell	F1-Score	0.5455	0.6667	-	0.7692	0.9412	0.9091
	Precision	1.0000	0.5000	0.0000	0.7692	0.8889	0.8333
	Recall	0.3750	1.0000	-	0.7692	1.0000	1.0000
Baseline	F1-Score	0.1176	-	-	0.2500	0.1333	-
	Precision	0.1111	0.0000	0.0000	0.6667	0.1429	0.0000
	Recall	0.1250	0.0000	-	0.1538	0.1250	0.0000

Tabelle 5.4: Attribute des Klassifizierers der Emotionsanalyse

Feinabgestimmte Sentimentanalyse

Unter der Annahme, dass ein App-Review eher eine negative Sentimentpolarität aufweist, wurde die Baseline so gestaltet, dass sie jedes Review als negativ kennzeichnet. Die Ergebnisse des Modells zur Sentimentanalyse übersteigen in allen Bereichen deutlich die Resultate der Baseline. Alle Modellergebnisse liegen dabei um den Wert von 0.88 ± 0.02 (siehe Tabelle 5.3). Aus der Tabelle 5.5 geht deutlich hervor, dass die Prognose der neutralen Sentimentpolarität am schlechtesten abschneidet, was den Ergebnissen der Evaluation des feinabgestimmten Modells im Abschnitt 5.1.1 entspricht. Der Recall und die Precision der neutralen Sentimentpolarität deuten darauf hin, dass übermäßig viele Reviews als neutral eingestuft wurden, was wiederum den Recall-Score der übrigen Sentimentpolaritäten beeinträchtigt hat.

Metrik		positive	neutral	negative
Modell	F1-Score	0.9231	0.8182	0.9091
	Precision	1.0000	0.6923	1.0000
	Recall	0.8333	1.0000	0.8571
Baseline	F1-Score	-	-	0.7368
	Precision	-	-	0.5833
	Recall	0.0000	0.0000	1.0000

Tabelle 5.5: Attribute des Klassifizierers der feinabgestimmten Sentimentanalyse

SentiStrength

Die Baseline der SentiStrength-Analyse verfolgt das gleiche Prinzip wie die Sentimentanalyse, indem alle Reviews als negativ klassifiziert werden. Die Ergebnisse des SentiStrength-Modells schneiden grundlegend schlechter ab als die der Baseline, mit Ausnahme der Macro-Recall-Metrik, in der das Modell besser abschneidet (siehe Tabelle 5.3). Das unbefriedigende

Ergebnis könnte auf die Methode zur Ableitung der Sentimentpolarität während der SentiStrength-Berechnung zurückzuführen sein, die sämtliche SentiStrength-Werte von jedem Wort innerhalb der Reviews aufsummiert. Eine verbesserte Methode zur Ermittlung der Sentimentpolarität über SentiStrength wäre wünschenswert. Eine Möglichkeit könnte beispielsweise sein, alle SentiStrength-Werte zu summieren und dann durch die Anzahl aller SentiStrength-Werte, die ungleich Null sind, zu teilen. Aus der Tabelle 5.6 geht hervor, dass der Recall für die positive Sentimentpolarität bei eins liegt. Dies lässt darauf schließen, dass mehr positive SentiStrength-Wörter pro Review gefunden wurden, wodurch ein Review eher als positiv bewertet wird. Ein möglicher Grund dafür könnte sein, dass SentiStrength nicht speziell auf die Domäne der App-Reviews zugeschnitten ist, weshalb positive Wörter mit einer Precision von 0.2608 nicht so präzise erkannt werden wie negative Wörter mit 0.8571.

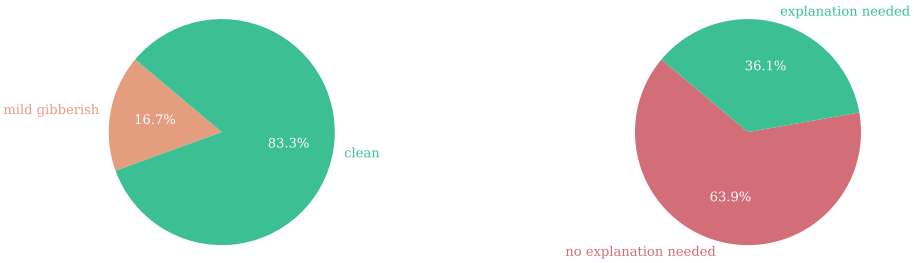
Metrik		**positive**	**neutral**	**negative**
Modell	F1-Score	0.4138	0.2667	0.4286
	Precision	0.2608	0.3333	0.8571
	Recall	1.0000	0.2222	0.2857
Baseline	F1-Score	-	-	0.7368
	Precision	-	-	0.5833
	Recall	0.0000	0.0000	1.0000

Tabelle 5.6: Attribute des Klassifizierers der SentiStrength-Analyse

Qualitätsanalyse

Die Erkennung der sprachlichen Qualität, innerhalb der Applikation auch als „Gibberish“ bezeichnet, nimmt als Baseline an, dass sämtliche Reviews mit dem Attribut „clean“ gekennzeichnet sind. Diese Zusammensetzung ergibt sich aus der Untersuchung der erhobenen „Ground Truth“-Daten, bei der festgestellt wurde, dass die Kategorie „clean“ dominiert (siehe Abbildung 5.2a). Die Überlegenheit der Kategorie „clean“ entsteht aufgrund der minimalen Anforderung von 190 Zeichen pro Review. Es ist auffällig, dass Reviews, die besonders lang, meistens auch sehr gut geschrieben sind. Daher ist die Auswertung der Modellergebnisse im Vergleich zur Baseline nicht universell gültig, da kürzere Bewertungen ausgenommen sind. Dennoch liefert die Stichprobenanalyse folgende Ergebnisse: Im Durchschnittstyp Micro zeigen die Modellergebnisse und die Baseline gleiche Werte, nur bei den Macro-Metriken treten Unterschiede auf (siehe Tabelle 5.3). Der Macro-Recall ist für das Modell wesentlich höher als für die Baseline. Der Macro-F1-Score und die Macro-Precision sind hingegen bei der Baseline höher. Die Tabelle 5.7 zeigt, dass das Modell anhand der F1-Scores für die Kategorie

„clean“ besser abschneidet als für „mild gibberish“. Es könnte angenommen werden, dass die beiden Kategorien „clean“ und „mild gibberish“, die einem Review eine gute bis sehr gute sprachliche Qualität zuschreiben, sich oft schwer voneinander abgrenzen lassen. Bei der Betrachtung eines Reviews könnten kleine grammatikalische oder andere Fehler häufig unbemerkt bleiben.



(a) Verteilung der sprachlichen Qualität in den aus der Stichprobe resultierenden annotierten Daten

(b) Verteilung des Erklärungsbedarfs in den aus der Stichprobe resultierenden annotierten Daten

Abbildung 5.2: Attributverteilung innerhalb des „Ground Truth“ der Stichprobe, insbesondere im Kontext der Qualitätsanalyse und der Erklärungsbedarfsanalyse

Metrik		noise	word salad	mild gibberish	clean
Modell	F1-Score	-	-	0.6667	0.8889
	Precision	-	-	0.5000	1.0000
	Recall	-	-	1.0000	0.8000
Baseline	F1-Score	-	-	-	0.9090
	Precision	-	-	-	0.8333
	Recall	-	-	0.0000	1.0000

Tabelle 5.7: Attribute des Klassifizierers der Qualitätsanalyse

Erklärungsbedarfsanalyse

Die Baseline für die Analyse des Erklärungsbedarfs legt fest, dass alle Reviews voraussichtlich keinen Erklärungsbedarf haben. Diese Entscheidung basiert auf der überwiegenden Tendenz der Stichprobe, die besagt, dass ein Review eher keinen Erklärungsbedarf aufweist (siehe Abbildung 5.2b). Aus der Tabelle 5.3 geht hervor, dass die Modellergebnisse die Baseline in allen Metrikwerten übertreffen und mit Werten nahe 0,8 ein gutes Resultat erzielen. Die Tabelle 5.8 zeigt, dass die Nichterfassung anhand

des besseren F1-Scores zuverlässiger erfolgt, als die Identifizierung eines Erklärungsbedarfs in einem Review. Die Precision dieser Nichterfassung in den Modellergebnissen ist mit 0,9 besonders hoch. Aufgrund der Zuweisung der Baseline, dass Reviews ausschließlich keinen Erklärungsbedarf enthalten, ergibt sich trivialerweise ein maximaler Recall von eins.

Metrik		no explanation needed	explanation needed
Modell	F1-Score	0.8372	0.7586
	Precision	0.9000	0.6875
	Recall	0.7826	0.8462
Baseline	F1-Score	0.7797	-
	Precision	0.6389	-
	Recall	1.0000	0.0000

Tabelle 5.8: Attribute des Klassifizierers der Erklärungsbedarfsanalyse

Eigenschaftsanalyse

Die Baseline zur Bestimmung der Merkmale eines Reviews weist jedem einzelnen Review eine zufällige Auswahl an Eigenschaften zu, wie zum Beispiel „user experience“ und „text rating“. Die Tabelle 5.3 zeigt, dass das Modell bei der Durchschnittsbildung Micro eine bessere Performance als Macro erzielt. Dies könnte darauf zurückzuführen sein, dass häufiger auftretende Merkmale besser klassifiziert werden können als solche, die seltener vorkommen. Die Abbildung 5.3 zeigt eine ungleichmäßige Häufigkeit bei der Zuordnung der Eigenschaften innerhalb des „Ground Truth“. Weiterhin wird die Eigenschaft

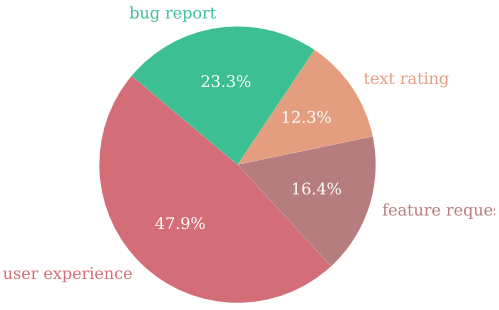


Abbildung 5.3: Verteilung der Merkmale aus der Eigenschaftsanalyse in den aus der Stichprobe resultierenden annotierten Daten

„user experience“ perfekt vorhergesagt, was mit einem Anteil von 47,9% an allen annotierten Reviews den größten Teil ausmacht (siehe Tabelle 5.9).

Ebenfalls überdurchschnittlich gut ist die Klassifizierung der Eigenschaft „bug report“ mit Metrikwerten von 94,12%, die mit 23,3% am zweithäufigsten im „Ground Truth“ vertreten ist. Diese beiden Kategorien nehmen zusammen 71,2% des „Ground Truth“ ein und weisen bessere Metrikwerte auf als die übrigen Kategorien „feature request“ und „text rating“. Die Accuracy der gesamten Eigenschaftsanalyse beträgt 0,5556. Dieser Wert, der auf den ersten Blick niedrig erscheinen könnte, erhält eine andere Bewertung, wenn in Erwägung gezogen wird, dass jedem Review bis zu vier Merkmale zugewiesen werden können. Im Kontext der Multilabel-Klassifikation erscheint das Gesamtergebnis demzufolge positiver. Die Eigenschaft „user experience“ schneidet besonders gut ab, da praktisch jedes Review eine solche Eigenschaft aufweist und somit ein hoher Metrik-Wert schnell zustande kommen kann. Bei „feature request“ und „text rating“ ist auffällig, dass der Recall wesentlich höher als die Precision ist. Dadurch neigt die Klassifizierung dazu, Reviews diesen Merkmalen zuzuweisen, obwohl sie laut der „Ground Truth“ eigentlich nicht dazu gehören.

Metrik		**user experience**	**feature request**	**text rating**	**bug report**
Modell	F1-Score	1.0000	0.7143	0.6087	0.9412
	Precision	1.0000	0.625	0.5000	0.9412
	Recall	1.0000	0.8333	0.7778	0.9412
Baseline	F1-Score	0.5600	0.4444	0.3846	0.3636
	Precision	0.9333	0.4000	0.2941	0.375
	Recall	0.4000	0.5000	0.5556	0.3529

Tabelle 5.9: Attribute des Klassifizierers Characteristic

5.2 Grenzen und Gefahren der Validität

Die Beurteilung der Grenzen und Gefahren der Validität basiert auf den vier Validitätsarten: „Construct Validity“, „Internal Validity“, „Conclusion Validity“ und „External Validity“, die sich auf die Arbeit von Wohlin et al. [87] bezieht.

5.2.1 Construct Validity

Für die Feinabstimmung des Klassifizierers „Sentimentanalyse“ wurde eine Umfrage zur Annotation von Reviews durchgeführt. Üblicherweise wird bei einer solchen Datenerfassung eine doppelte oder dreifache Annotation desselben Datensatzes angestrebt, um Fehler bei der Annotation durch einzelne Personen auszuschließen [70]. Beispielsweise können für das Training

des Modells nur die Daten verwendet werden, bei denen alle Personengruppen übereinstimmend annotiert haben. Zusätzlich kann bei Auftreten von Diskrepanzen in der Annotation der Reviews eine Kontrollinstanz zur Verfügung stehen, welche die Reviews erneut annotiert, um die Annotation zu überdenken oder die betroffenen Reviews nicht in das Training des Modells einzubeziehen. Der Einsatz einer solchen doppelten oder dreifachen Annotation, zusammen mit einer zusätzlichen Kontrollinstanz, hätte allerdings den Arbeitsaufwand zusätzlich erhöht. Um jedoch falschen Annotationen entgegenzuwirken, wurde die Auswahl der Personen, welche die Annotation vorgenommen haben, sorgfältig getroffen. Hierfür wurden ausschließlich Informatikstudierende ausgewählt, die sich zum Zeitpunkt der Umfrageerhebung im sechsten Bachelorsemester befanden und somit über domänenspezifisches Hintergrundwissen in der Softwareentwicklung verfügten. Zudem wurde mit dem „StratifiedGroupKFold“ sichergestellt, dass der Trainingsdatensatz und der Testdatensatz möglichst vielfältig zusammengesetzt sind, um spezifische Fehler von Personengruppen zu minimieren [78].

Darüber hinaus bedarf die Annotation der Stichprobe zur Evaluation aller verwendeten Klassifizierer einer kritischen Betrachtung, da die Reviews ausschließlich vom Verfasser dieser Arbeit annotiert wurden. Eine Annotation durch mehrere Probanden wäre ratsam, um die durch Einzelannotation entstehende Subjektivität zu reduzieren.

5.2.2 Internal Validity

Die Annotation der Reviews für die Sentimentanalyse könnte möglicherweise unzureichend sein, da die Probanden unsicher waren, welche Gesamtpolarität sie dem Review zuordnen sollten. Aus diesem Grund wurde nach der Umfrage zur Annotation der Reviews eine zweite Umfrage durchgeführt, um die Zuverlässigkeit der erhobenen annotierten Daten festzustellen. Basierend auf dieser Umfrage (siehe Abschnitt 4.4.2) gaben der Großteil der Probanden an, sich bei der Annotation der Reviews sicher zu fühlen. Die Unsicherheit lag größtenteils darin begründet, dass verschiedene Sichtweisen ein jeweiliges Review unterschiedlich bewerten lassen. Zudem führten widersprüchliche Emotionen, die aufgrund des informellen Sprachstils im Internet allgemein üblich sind, ebenfalls zu Unsicherheit [6]. Zwei Nutzer merkten zusätzlich an, dass in den Kommentaren eher ein Ratschlag für die Community verfasst war und es daher schwierig war, die Sentimentpolarität dem Verfasser oder der Community zuzuordnen. Weiterhin wurden die Englischkenntnisse der Teilnehmenden erfragt, da die zu annotierenden Reviews ausschließlich in englischer Sprache verfasst waren. Es lässt sich annehmen, dass aufgrund der großen Anzahl an Probanden, die sich gute bis sehr gute Englischkenntnisse zuschrieben, Fehler bei der Annotation nicht der englischen Sprache geschuldet waren. Zusammenfassend kann festgestellt werden, dass die

Probanden in ihren Aussagen zuversichtlich waren, sodass den annotierten Daten vertraut werden kann. Dennoch wären mehr annotierte Reviews und eine größere Anzahl von Probanden wünschenswert, um die Diversität des Trainingsdatensatzes weiter zu erhöhen.

Abgesehen von einer Erweiterung der Datenbasis könnte die Methode der Annotation eines Reviews hinsichtlich der Sentimentpolaritäten überdacht werden. Es kann herausfordernd sein, ein Review zu klassifizieren, das sowohl positive als auch negative Sentiments aufweist [70]. Eine solche Gesamteinschätzung kann sehr subjektiv ausfallen. Eine mögliche Lösung könnte sein, Annotationen auf Satzebene durchzuführen, auch wenn das Problem der Bestimmung der Gesamtpolarität des Reviews dann weiterhin bestehen würde. Ein anderer Ansatz könnte darin bestehen, dem Nutzer die Genauigkeit der Vorhersage mitzuteilen, sodass dieser die Verlässlichkeit der Information einschätzen kann. Alternativ könnte die neutrale Sentimentpolarität in die zwei Kategorien „no sentiment“ und „mixed sentiment“‘ unterteilt werden [70]. Dies würde es ermöglichen, Reviews, die sowohl positive als auch negative Sentiments aufweisen, eindeutig zu klassifizieren und somit die Subjektivität bei der Sentimentzuweisung zu reduzieren. Allerdings könnten solche Methoden die Komplexität der Anwendung erhöhen und zusätzliche Risiken darstellen, beispielsweise eine überproportionale Annotation der Klasse „mixed sentiment“.

5.2.3 Conclusion Validity

Für die Analyse aller Klassifizierer wurde eine Stichprobenanalyse mit 36 Reviews erstellt, die eine Mindestlänge von 190 Zeichen aufweisen (siehe Abschnitt 4.4.1). Die Mindestlänge bei der Auswertung der Metriken erwies sich besonders für den Klassifizierer der Qualitätsanalyse als unpassend, da kürzere Reviews häufiger informellen Text enthalten als die in der Stichprobenanalyse erhobenen längeren Reviews. Diese Erkenntnisse könnten auch bei anderen Klassifizierern aufgetreten sein, fallen aber nicht so stark ins Gewicht. Gleichwohl sollte angemerkt werden, dass die festgelegte Mindestlänge von 190 Zeichen durchaus berechtigt ist. Sie ermöglichte die Analyse gehaltvollerer Reviews und bot die Chance, die Leistungsfähigkeit der Klassifizierer genauer zu testen. Somit war es die richtige Entscheidung, für eine kleinere Stichprobe eine solche Mindestlänge miteinzubeziehen. Es wäre jedoch erstrebenswert, die Anzahl der Reviews in der Stichprobe zu erhöhen, allerdings hätte dies den Zeitaufwand erheblich erhöht.

5.2.4 External Validity

Die Evaluation der Klassifizierer wurde ausschließlich in den beiden populärsten App Stores durchgeführt: dem Apple App Store und dem Google Play Store. Ob eine weitere Anwendung auf andere App Stores oder Review-

Portale möglich ist, bleibt fraglich. Es ist jedoch anzunehmen, dass andere App Stores oder soziale Netzwerke einen vergleichbaren sprachlichen Stil und eine ähnliche Satzstruktur aufweisen, die sich auf informale Ausdrucksweisen und kurze Texte beschränken. Daher könnte eine Anwendung der Applikation auch für andere Bereiche des Internets in Erwägung gezogen werden.

Die Sentimentanalyse wurde speziell auf App-Reviews feinabgestimmt. Somit könnten andere Marktplätze, wie beispielsweise die Bewertung der Sentimentpolarität von Amazon-Reviews, möglicherweise falsche Ergebnisse liefern. Bei Amazon-Produkten handelt es sich eher um Hardware als um Software, was eine andere Grundlage für die Bewertung von Reviews bedeutet.

Die Klassifizierer, welche nicht spezifisch feinabgestimmt, sondern ausschließlich auf umfangreichen Internet-Datensätzen trainiert wurden, stellen eine ausgewogene Lösung für den Einsatz in verschiedenen Internetbereichen dar. Allerdings ist eine jeweilige Evaluation im entsprechenden Anwendungsbereich notwendig, welche in dieser Arbeit ebenfalls für die Domäne der App-Reviews erfolgreich durchgeführt wurde. Durch die Nutzung einer fortschrittlicheren Baseline, wie dem „State of the Art“-Ansatz, könnte jedoch eine präzisere Bewertung für jeden einzelnen Klassifizierer erzielt werden.

5.3 Schlussfolgerung

Die Evaluation der Anwendung *feelio* wurde umfassend durchgeführt und präsentiert Klassifizierer, die im Bereich der Anforderungsanalyse oder der allgemeinen App-Analyse vorteilhaft sind. Alle Klassifizierer, mit Ausnahme der SentiStrength-Analyse, übertreffen die jeweilige Baseline und bieten insgesamt ein solides Ergebnis für die praktische Anwendung. In der Softwareentwicklung lag der Fokus auf der Modularität, sodass jederzeit Verbesserungen vorgenommen und Klassifizierer ausgetauscht werden können, um flexibel auf aktuelle Marktentwicklungen zu reagieren.

Kapitel 6

Verwandte Arbeiten

6.1 Reviewanalyse

Maalej et al. [49] präsentieren Techniken zur Klassifizierung von App-Reviews in die Kategorien „bug reports“, „feature requests“, „user experience“ und „text ratings“ für den Einsatz im Bereich des Requirements Engineering (RE). Die verwendeten Techniken vereinen unterschiedliche NLP-Methoden (wie „stopword“-Entfernung und „stemming“), Review-Metadaten (wie Sternebewertung und Erstellungszeit), Sentimentanalyse, den Basis-Klassifizierer „string matching“ und den Supervised-Learning-Ansatz „bag of words“. Für die Sentimentanalyse kam speziell die Anwendung SentiStrength [83] zum Einsatz. Abgesehen von der Klassifizierung der Reviews erstellten die Autoren eine zugehörige Benutzeroberfläche, die es dem Nutzer ermöglicht, Reviews zu importieren, deren Trends anzuzeigen, einen App-Store-Vergleich durchzuführen und einzelne Reviews sowie deren Klassifizierungen nachzuvollziehen. Um die Software auf ihre Praxistauglichkeit zu überprüfen, wurde zudem ein Interview mit Entwicklern und Analysten durchgeführt. Die Teilnehmer des Interviews unterstrichen den Wunsch nach einem automatisierten Tool, welches uninformative Reviews herausfiltert und „bug reports“ sowie „feature requests“ leichter erkennbar macht. In Bezug auf die evaluierte Software wurde angemerkt, dass der Klassifizierer noch zuverlässiger arbeiten könnte. Wichtige Informationen sollten für den Nutzer auf den ersten Blick erkennbar sein. Ein webbasiertes Tool, das ein responsives Layout für mobile Geräte und den Desktop-Einsatz bietet, würde zudem die Handhabung der Software erheblich erleichtern.

Dhinakaran et al. [14] präsentierten einen Ansatz, um den Aufwand der manuellen Auswertung von Reviews mithilfe der Methode „active learning“ zu reduzieren. Sie klassifizierten Reviews in die Kategorien „features“, „bugs“, „rating“ und „user experience“. Dabei stellten sie fest, dass „active learning“ in ihren Untersuchungen besser abschnitt als ihre Baseline. Bei der Baseline wurden zufällige Elemente zum Trainingsdatensatz hinzugefügt, während bei

„active learning“ die spezifische „uncertainty sampling“-Strategie zum Einsatz kam.

Walid und Hadeer [24] widmeten sich ebenfalls der Klassifizierung von Reviews, und zwar in die vordefinierten Kategorien „bug report“, „feature strength“, „feature shortcoming“, „user request“, „praise“, „complaint“ und „usage scenario“. Im Rahmen ihrer Studie verglichen sie die Leistung verschiedener Machine-Learning-Ansätze, darunter „Naïve Bayes“, „support vector machines“, „logistic regression“, „neural network“, und auch Kombinationen dieser Methoden.

Cen et al. [6] haben sich die Identifizierung von Reviews mit sicherheitsrelevanten Aspekten zur Aufgabe gemacht. Für das Training der Supervised-Learning-Methode „bag of words“ wurden Reviews aus dem Google Play Store gesammelt, die bestimmte sicherheitsrelevante Schlüsselwörter enthielten. Diese Reviews wurden annotiert und für das Training des Modells verwendet.

Henao et al. [27] verfolgten den Ansatz des „transfer learning“, speziell das BERT-Modell, und verglichen es mit traditionellen Machine-Learning-Methoden. Die Reviews wurden in Kategorien wie „problem report“, „feature request“ und „irrelevant“ klassifiziert. Dabei wurden sowohl ein monolinguales als auch ein multilinguales BERT-Modell zur Betrachtung herangezogen. Es zeigte sich, dass das monolinguale BERT-Modell bzw. der „transfer learning“-Ansatz die traditionellen Machine-Learning-Ansätze übertraf.

Chen et al. [9] präsentierten das Framework „AR-Miner“, das informative von irrelevanten Reviews unterscheidet. Anschließend gruppiert es diese mittels „topic modeling“ und stellt sie in einer Rangfolge nach ihrem informativen Gehalt sortiert visuell dar.

Guzman et al. [25] extrahierten Features aus Reviews, wobei jedem Feature ein Sentiment zugewiesen wurde. Ähnliche Features wurden dann zu einer Gruppe zusammengefasst. Für die Sentimentanalyse wurde SentiStrength [83] verwendet.

Carreño und Winbladh [22] stellten einen Ansatz vor, mit dem anhand von App-Reviews neue oder modifizierte Anforderungen innerhalb der Softwareentwicklung ermittelt werden können. Sie gingen davon aus, dass die Relevanz eines Themas direkt mit der Häufigkeit seines Auftretens in den Reviews zusammenhängt. Auf dieser Grundlage wurde eine Methode zur Themenextraktion entwickelt. Diese Methode stellt eine komprimierte Liste von Anforderungen bereit, die speziell für Anforderungsanalysten konzipiert ist. Der Analyst kann aus dieser Liste die für ihn relevanten Anforderungen auswählen. Dieser Ansatz ist wesentlich zeitsparender, als sämtliche Reviews im Detail durchzulesen.

Jeong und Kim [40] nutzen die Sentimentanalyse als Indikator, um besonders informative App-Reviews herauszufiltern und so eine kompakte Auswahl für spätere, detaillierte Analysen bereitzustellen. Die Sentimentbewertung, die auf einer 5-Punkt-Likert-Skala basiert, wurde mit der „topic modeling“-Methode kombiniert. Es stellte sich heraus, dass besonders informative

Reviews überwiegend in den Kategorien sehr negativ, negativ und sehr positiv auftreten, Kategorien, die insgesamt nur selten in der Gesamtheit der Reviews vorkommen.

Panichella et al. [67] ordnen App-Reviews den Kategorien „Information Seeking“, „Information Giving“, „Feature Request“ und „Problem Discovery“ zu, um effizienter auf Feedback aus den Reviews innerhalb der Softwareentwicklung reagieren zu können. Ihr Ansatz kombiniert verschiedene NLP-Methoden, die sowohl die Sentimentanalyse als auch die reine Textanalyse mittels Mustererkennung einschließen.

Die Arbeiten von Jacobsen [39] und Kuhnke [44] beschäftigen sich beide mit der Erkennung von Erklärungsbedarf in App-Reviews.

Jacobsen [39] stellt eine grafische Anwendung vor, die Erklärungsbedarf in Reviews mittels der „rule-based matching“-Methode identifiziert. Der Prozess der App-Analyse folgt einem bestimmten Ablauf. Zunächst wird eine Analyse der Reviews durchgeführt, woraufhin dem Nutzer die Reviews mit markiertem Erklärungsbedarf präsentiert werden. Der Nutzer überprüft diese dann innerhalb der Anwendung und nimmt gegebenenfalls Anpassungen vor. Im Anschluss erfolgt eine weitere Analyse der Reviews mithilfe des Programms. Mit seinem iterativen Ansatz ermöglicht das Programm eine kontinuierliche Optimierung der Ergebnisse, die sich vorteilhaft auf die Analyse später hinzugefügter Reviews auswirkt.

Kuhnke [44] hingegen stellt sechs Methoden zur Erkennung des Erklärungsbedarfs vor. Darunter befinden sich eine einfache Heuristik, welche die Wortanzahl in Sätzen mit häufig vorkommenden Wörtern mit Erklärungsbedarf zählt, ein „Naïve Bayes“-Ansatz, drei verschiedene Arten von neuronalen Netzwerken und eine kombinierte Variante. Letztere zählt, wie viele Segmente eines Satzes von einem neuronalen Netzwerk als Erklärungsbedarf klassifiziert werden.

6.2 Sentimentanalyse

Imtiaz et al. [37] bewerteten populäre, etablierte Tools sowohl für die Sentimentanalyse als auch zur Bestimmung des Freundlichkeitsgrades. Dafür erstellten sie einen Datensatz basierend auf GitHub-Kommentaren, der entsprechend annotiert wurde. Nach einer Evaluierung der Leistung kamen sie zu dem Schluss, dass die bestehenden Tools größtenteils unzuverlässige Vorhersagen lieferten. Sie stellten zudem fest, dass sich die Annotationen je nach Personengruppe erheblich unterscheiden können. Aufgrund dieser Erkenntnisse sprachen sie sich für ein standardisiertes Annotationsverfahren aus, um in Zukunft qualitativ bessere annotierte Daten zu erhalten.

Ding et al. [15] legten ebenfalls einen annotierten Datensatz an, der auf GitHub-Kommentaren basiert. Daraufhin entwickelten sie das Analyse-Tool *SentiSW*, das trinäre Sentimentpolaritäten sowie deren Ziel-Kontext (Person

oder Projekt) vorhersagt. Sie fanden heraus, dass ihre Supervised Learning Methode besser performte als der „State of the Art“-Ansatz.

Novielli et al. [58] annotierten Fragen, Antworten und Kommentare von Stack Overflow basierend auf sechs Emotionen, um einen neuen Datensatz zu erstellen. In einer weiteren Arbeit [57] untersuchten Novielli et al. vier Sentimentanalyse-Anwendungen, die speziell für das Software Engineering konzipiert wurden. Lexikonbasierte Ansätze zeigten bei bereichsübergreifenden Analysen bessere Ergebnisse als Supervised Learning Methoden. Allerdings erwiesen sich die Supervised Learning Methoden in ihrer speziellen Trainingsdomäne als überlegen gegenüber den lexikonbasierten Verfahren. Weiterhin führten sie eine Evaluation bezüglich der Fehlklassifikationen der Annotationen durch und entwickelten daraus Leitlinien für den zuverlässigen Einsatz von Analyse-Tools im Software Engineering.

Lin et al. [46] präsentierten einen wenig erfolgreichen Versuch, bei dem ein Modell auf selbstannotierten Daten von Stack Overflow trainiert wurde, welches in der Evaluation nicht die gewünschte Performance erzielte. Infolgedessen entschlossen sie sich, andere vergleichbare Sentimentanalyse-Tools zu überprüfen. Dabei stellten sie fest, dass diese Sentimentanalysen ähnlich unbefriedigende Ergebnisse lieferten. Daher empfiehlt es sich, den Einsatz dieser Tools in der Praxis sorgfältig abzuwägen. Zudem evaluierten sie einen verhältnismäßig kleinen App-Review-Datensatz, bei dem die neutrale Kategorie im Vergleich zu den beiden anderen Polaritäten deutlich schlechter abschnitt.

Uddin et al. [84] untersuchten unter anderem Sentimentanalysen, die speziell auf das Software Engineering von Lin et al. [45, 46] ausgerichtet waren. Bei der Betrachtung einer kombinierten Fassung ihrer Modelle stellte sich heraus, dass diese im Vergleich zu den einzelnen Modellen von Lin et al. weniger effektiv war. Vor diesem Hintergrund entwickelten sie das Supervised Learning Tool *Sentisead*. Durch den Einsatz des Transfer Learning-Ansatzes gelang eine weitere Optimierung, wodurch *Sentisead* bessere Ergebnisse als die Modelle von Lin et al. erreichte.

Ortu et al. [63] annotierten Fehlerberichte von Entwicklern hinsichtlich ihrer Emotionen. Im ersten Schritt wurden die Berichte anhand von sechs Emotionen kategorisiert. Nach der Auswertung stellte sich heraus, dass von diesen sechs Emotionen nur „love“, „joy“ und „sadness“ konsistente Übereinstimmungen bei wiederholten Annotationen des gleichen Fehlerberichts aufwiesen. In der zweiten Phase der Annotation wurden daher nur diese Emotionen sowie eine zusätzliche neutrale Kategorie berücksichtigt. Die beiden erhobenen Datensätze ergänzten eine frühere Arbeit [62] der gleichen Autoren, die für ihre Annotationen die Attribute „joy“, „love“, „sadness“ und „anger“ verwendet hatten.

Potts et al. [70] präsentieren *DynaSent*, welches zur Bewertung der Leistung von Sentimentmodellen eingesetzt werden kann. Es wird detailliert dargelegt, auf welche Schritte bei der Annotation von Daten für ein

nachfolgendes Sentimentmodell geachtet werden sollte. Eine ihrer zentralen Erkenntnisse ist, dass trinäre Sentimentpolaritäten nicht lediglich in „positiv“, „neutral“ und „negativ“ kategorisiert werden sollten. Stattdessen wird empfohlen, anstelle der neutralen Polarität die Kategorien „no sentiment“ und „mixed sentiment“ zu berücksichtigen.

6.3 Abgrenzung zu verwandten Arbeiten

Um die geschaffene Arbeit von den zuvor erwähnten Arbeiten zu differenzieren, bietet die Anwendung *feelio* ein umfangreicheres Analysewerkzeug, welches sowohl App-Reviews aus den beiden beliebtesten App Stores, dem Google Play Store und Apple App Store crawlt, als auch den Import per CSV-Datei ermöglicht, um eine komplexere Analyse der App-Reviews für einen App-Analysten durchzuführen. Statt nur einen Klassifizierer zu entwickeln, welcher Techniken wie die Sentimentanalyse und „bag of words“ kombiniert, um eine Eigenschaft wie „feature request“ möglichst korrekt zu bestimmen, lag der Fokus der Anwendung *feelio* darauf, mehrere einzelne Klassifizierer wie Emotionsanalyse, Sentimentanalyse, SentiStrength-Analyse, Qualitätsanalyse, Eigenschaftsanalyse und Erklärungsbedarfanalyse zu implementieren. Diese können vom Nutzer innerhalb der Benutzeroberfläche mit weiteren Review-Metadaten, wie der Sternebewertung, der Länge eines Reviews etc., kombiniert werden. Dadurch wird es ermöglicht, einen personalisierten Filter für die eigenen Bedürfnisse zu erstellen und die gewünschten Statistiken und Grafiken anzuzeigen. Auf Usability-Aspekte wurde zudem vermehrt Wert gelegt. Die Anwendung *feelio* kann mit wenig Aufwand auf einen Server portiert werden, sodass sie sowohl von unterwegs über das Smartphone als auch im Büro am Desktop abgerufen werden kann. Im Vergleich zu anderen Studien, die sich auf spezifische Methoden wie „active learning“ oder „bag of words“ zur Klassifizierung von App-Reviews konzentrieren, widmet sich diese Arbeit unter anderem mehreren von Hugging Face stammenden Modellen, die auf der Transformer-Architektur basieren. SentiStrength und „string matching“ kamen ebenso für bestimmte einzelne Klassifizierer zum Einsatz.

Ein wesentlicher Aspekt dieser Arbeit ist zudem die Anwendung der Transfer-Learning-Methode zur Feinabstimmung eines Modells für die Sentimentanalyse. Dabei mussten App-Reviews annotiert werden, da kein Datensatz dieser Größe im Internet vorhanden war. Nach dem Stand vom 17.08.2023 handelt es sich um den größten App-Review-Datensatz, der die Sentimentpolaritäten „positiv“, „neutral“ und „negativ“ als Attribute beinhaltet.

Die multifunktionale Analyse-Anwendung *feelio* stellt ein universelles Werkzeug zur App-Review-Analyse dar und hebt sich damit zusammenfassend von bisherigen Herangehensweisen ab.

Kapitel 7

Fazit

Das abschließende Kapitel Fazit widmet sich einer kurzen Zusammenfassung und beleuchtet die ausstehenden Tätigkeiten, die noch in Betracht gezogen werden können, um die Forschung weiterzuführen.

7.1 Zusammenfassung

Die beiden beliebtesten App Stores: der Google Play Store und der Apple App Store umfassen eine Vielzahl von Apps, die von den Nutzern durch geschriebene Reviews bewertet werden können [2]. Besonders beliebte Apps erhalten in kürzester Zeit viele Reviews [28]. Um auf das Nutzerfeedback reagieren zu können und möglicherweise das Requirements Engineering zu optimieren, bedarf es eines automatisierten Analysewerkzeugs. Hier setzt das Analysewerkzeug *feelio* an. Es wurde entwickelt, um App-Analysten und interessierten Nutzern bei der Auswertung von App-Reviews zu unterstützen. Mithilfe von *feelio* kann der Nutzer eine umfassende App-Analyse erstellen, indem er Apps aus dem Google Play Store, dem Apple App Store crawlt oder Reviews aus einer CSV-Datei importiert. Alle Review-Datenquellen können kombiniert werden, um spätere Vergleiche zwischen den Quellen zu ziehen. Jedes Review wird anhand verschiedener Klassifizierer analysiert:

1. **Die Emotionsanalyse** ordnet ein Review den sechs Basisemotionen nach Ekman [17] zu: Zorn („anger“), Abscheu („disgust“), Furcht („fear“), Freude („joy“), Traurigkeit („sadness“), Überraschung („surprise“), ergänzt durch eine neutrale Klasse.

2. **Die Sentimentanalyse** besteht aus einem selbst feinabgestimmten Modell, welches die Stimmung eines Reviews (Sentimentpolarität) als positiv, neutral oder negativ bewertet. Parallel dazu markiert das NLP-Tool SentiStrength in einem Review positiv und negativ gewichtete Begriffe und ordnet jedem Review ebenfalls eine entsprechende Sentimentpolarität zu.

3. **Die Qualitätsanalyse** ordnet die sprachliche Qualität eines Reviews den Kategorien „noise“, „word salad“, „mild gibberish“ oder „clean“ zu.
4. **Die Erklärungsbedarfsanalyse** entscheidet, ob ein Review weiteren Erklärungsbedarf aufweist.
5. **Die Eigenschaftsanalyse** kennzeichnet ein Review in den Kategorien „user experience“, „feature request“, „text rating“ oder „bug report“.

Die Analysemethoden: Emotionsanalyse, Sentimentanalyse, Qualitätsanalyse, Erklärungsbedarfsanalyse und Eigenschaftsanalyse stammen jeweils von einem Modell von Hugging Face.

Für die Sentimentanalyse wurde speziell ein Modell von Hugging Face feinabgestimmt, um eine höhere Genauigkeit für die Domäne der App-Reviews zu erzielen. Dabei wurde unter anderem ein annotierter Review-Datensatz mit 1080 Einträgen erhoben. Alle Klassifizierer, mit Ausnahme der SentiStrength-Analyse, übertreffen die festgelegte Baseline und bieten einem App-Analysten erheblichen Mehrwert. Mithilfe der zahlreich verwendeten Klassifizierer ermöglicht die Applikation *feelio* beispielsweise, verschiedene Korrelationen innerhalb der Review-Daten nachzuvollziehen oder spezifische Anforderungen zu definieren. Die Benutzerfreundlichkeit der Applikation wurde anhand der „10 Usability Heuristics for User Interface Design“ von Jakob Nielsen überprüft, um sicherzustellen, dass das Design den Anforderungen entspricht. Zusätzlich wurde die entworfene Anwendung *feelio* mit dem Gedanken an einen langlebigen Produktlebenszyklus entwickelt. Dabei war es von entscheidender Bedeutung, eine Modularität der Software zu gewährleisten. Dies ermöglicht beispielsweise das einfache Austauschen von Klassifizierern oder das Verändern von Filtermethoden, um die Handhabung so unkompliziert wie möglich zu gestalten.

7.2 Ausblick

Die Anwendung bildet eine gute Grundlage, um spätere Erweiterungen durchführen zu können. Um den Nutzen der Software zu maximieren, wäre das Hinzufügen oder Austauschen von Komponenten notwendig, um schnell die neuesten Analysemethoden, die auf dem Markt verfügbar sind, zu integrieren. Bestehende Klassifizierer können zudem kontinuierlich verbessert werden. Beispielsweise könnte für die Sentimentanalyse ein weiterer umfangreicher, annotierter Datensatz erhoben werden, um die Methode noch spezifischer auf App-Reviews auszurichten und so die Leistung des Modells weiter zu steigern. Außerdem könnte in regelmäßigen Abständen ein neues Modell mit den bereits vorhandenen annotierten Daten trainiert werden, um zu prüfen, ob dieses das aktuelle Modell übertrifft. Eine alternative Konfiguration der Hyperparameter des feinabgestimmten Modells könnte ebenfalls zu verbesserten Ergebnissen führen.

Die anderen bestehenden Klassifizierer könnten ebenso mit dem Transfer-Learning-Ansatz feinabgestimmt werden, um noch bessere Metrikwerte zu erzielen. Dafür wäre jedoch die Erhebung großer annotierter Datensätze für den jeweiligen Trainings- und Testdatensatz eines feinabgestimmten Klassifizierers erforderlich. Die SentiStrength-Methode ist hilfreich, um positive und negative Wörter in einem Review zu erkennen. Die Bestimmung der Gesamtpolarität eines Reviews mit Hilfe des NLP-Tools SentiStrength ist jedoch nicht zufriedenstellend. Eine Alternative zur Berechnung der Gesamtpolarität oder eine modifizierte Version von SentiStrength, analog zu SentiStrength-SE [38], wäre wünschenswert.

Weiterhin könnte zur Steigerung der Leistungsfähigkeit von wörterbuchbasierten Klassifizierern, wie sie in der Erklärungsbedarfsanalyse zum Einsatz kommen, innerhalb der Benutzeroberfläche eine Funktion integriert werden, die es ermöglicht, Begriffe zum Wörterbuch hinzuzufügen oder daraus zu entfernen. Dadurch ließe sich ein personalisierter Klassifizierer für eine App-Analyse erstellen.

Ein weiterer Vorschlag könnte die Leistungssteigerung bestimmter, neu zu erstellender Klassifizierer betreffen, indem mehrere einzelne Klassifizierer zu einem umfassenderen Klassifizierer kombiniert werden. Dies könnte dazu dienen, bestimmte Korrelationen innerhalb der Reviews zu nutzen und so einen Synergieeffekt zu erzielen. Die Nutzerfreundlichkeit der Anwendung *feelio* wurde bislang ausschließlich durch die „10 Usability Heuristics for User Interface Design“ von Jakob Nielsen und durch persönliche Rückmeldungen einer kleinen Nutzergruppe (Kommilitonen und ausgewählte Personen) evaluiert. Es wäre ratsam, eine umfassendere Evaluation der Applikation durchzuführen, indem eine größere Nutzerstudie zur Usability durchgeführt wird. In einer solchen Studie könnten auch Fragen nach dem Mehrwert der Software, dem bisherigen Einsatz von App-Review-Analyse-Methoden und der Effektivität sowie Effizienz der Software im realen Arbeitsumfeld berücksichtigt werden. Die daraus resultierenden Erkenntnisse könnten mögliche Schwachstellen aufdecken und zur weiteren Optimierung von *feelio* beitragen. Zu guter Letzt ist geplant, die Software auf einen Server zu portieren, um den Zugriff auf die Software weltweit zu ermöglichen. Dies könnte die Anwendung auch kommerziell attraktiver machen.

Anhang A

Anhang

A.1 Use Cases

USE CASE 1	**Anmeldungsprozess**
Erläuterung	Beim ersten Aufruf der Webseite wird dem Nutzer eine Anmeldeoberfläche angezeigt.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Unteraufgabe
Stakeholder und Interessen	Systemnutzer: Der Nutzer möchte das Analysewerkzeug *feelio* nutzen und sich daher mit seinem Konto anmelden.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer ist aktuell nicht in einer Sitzung oder befindet sich im ausgeloggten Zustand.
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Dem Nutzer wird eine Anmeldemaske angezeigt, über die er sich auf der Webseite mittels E-Mail und Passwort einloggen kann.
Auslöser	Der Nutzer besucht die Webseite erstmals oder wird nach 24h durch ein Session-Timeout abgemeldet.
Beschreibung	1. Der Nutzer startet den Browser. 2. Der Nutzer ruft die Webseite auf. 3. Automatisch erfolgt eine Weiterleitung zur Adresse “<IP-Adresse>:8080/User/signIn“.
Erweiterungen	3.a. WENN der Nutzer zuvor eine andere Unterseite (z.B. „<IP-Adresse>:8080/AppLibrary“) aufgerufen hat, DANN wird er automatisch zur Adresse „<IP-Adresse>:8080/User/signIn“ weitergeleitet.

Tabelle A.1 Fortsetzung von vorheriger Seite

USE CASE 1	**Anmeldungsprozess**
Technologie	-
Priorität	wünschenswert
Verwendungshäufigkeit	regelmäßig

Tabelle A.1: Use Case zum Anmeldeprozess innerhalb der Applikation *feelio*

USE CASE 2	**Eingabe bei der Anmeldung**
Erläuterung	Der Nutzer kann sich beim Analysetool mit seiner E-Mail-Adresse und einem selbstgewählten Passwort anmelden.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Unteraufgabe
Stakeholder und Interessen	Systembediener: Der Nutzer möchte ein Konto für das Analysetool anlegen, damit er bei späteren Sitzungen auf seine gespeicherten Daten zugreifen kann.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer ist aktuell nicht in einer Sitzung oder befindet sich im ausgeloggten Zustand. • Der Nutzer hat sich bereits mit seiner E-Mail-Adresse und einem selbstgewählten Passwort registriert. • Der Nutzer ist abgemeldet (siehe gegebenenfalls Use Case 1 (Tabelle A.1) und 3 (Tabelle A.3)).
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Das System meldet den Nutzer an.
Auslöser	Der Nutzer möchte sich mit seinem bereits angelegten Konto anmelden, um auf seine erstellten Inhalte zuzugreifen.
Beschreibung	1. Dem Nutzer wird eine Anmeldemaske für E-Mail-Adresse und Passwort präsentiert. 2. Der Nutzer gibt eine korrekt formatierte E-Mail-Adresse in das vorgesehene Feld ein (z.B. „contact@timokurtz.de“). 3. Der Nutzer gibt das bei der Registrierung gewählte Passwort in das entsprechende Feld ein (z.B. „qufdy8-somkak-tumnEq“). 4. Anschließend klickt der Nutzer auf den Button „SIGN IN“. 5. Der Nutzer wird automatisch zur Adresse „<IP-Adresse>:8080/AppLibrary“ weitergeleitet.

Tabelle A.2 Fortsetzung von vorheriger Seite

USE CASE 2	**Eingabe bei der Anmeldung**
Erweiterungen	4.a. WENN das E-Mail-Feld leer ist, DANN zeigt das System die Fehlermeldung „E-mail is required.“. 4.b. WENN die E-Mail-Adresse nicht korrekt formatiert ist, DANN zeigt das System die Fehlermeldung „E-mail must be valid.“. 4.c. WENN das Passwort-Feld leer ist, DANN zeigt das System die Fehlermeldung „Password is required.“. 4.d. WENN das Passwort oder die E-Mail nicht korrekt eingegeben wurden, DANN zeigt das System die Fehlermeldung „Sign In Failed: The email or password you entered is incorrect. Please try again.“.
Technologie	-
Priorität	wünschenswert
Verwendungshäufigkeit	regelmäßig

Tabelle A.2: Use Case für die Dateneingabe während des Anmeldeprozesses in der Applikation *feelio*

USE CASE 3	**Session-Timeout**
Erläuterung	Nach 24 Stunden wird der Nutzer automatisch vom System abgemeldet.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Unteraufgabe
Stakeholder und Interessen	Systembediener: Der Nutzer möchte, dass sein Konto automatisch abgemeldet wird, um unbefugten Zugriff zu vermeiden.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand.
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Das System meldet den Nutzer automatisch ab.
Auslöser	Nach 24 Stunden im angemeldeten Zustand.
Beschreibung	1. Der Nutzer meldet sich erfolgreich an (vgl. Use Case 2, Tabelle A.2). 2. Entweder bleibt der Nutzer 24 Stunden inaktiv, oder er verwendet das Analysetool über diesen Zeitraum. 3. Nach Ablauf dieser 24 Stunden wird der Nutzer bei der nächsten Interaktion zur Adresse „<IP-Adresse>:8080/User/signIn“ weitergeleitet und muss sich erneut anmelden.

Tabelle A.3 Fortsetzung von vorheriger Seite

USE CASE 3	**Session-Timeout**
Erweiterungen	3.a. WENN keine Interaktion mit der Webseite stattfindet, DANN bleibt der Nutzer auf der aktuellen Seite.
Technologie	-
Priorität	wünschenswert
Verwendungshäufigkeit	regelmäßig

Tabelle A.3: Use Case zum Session-Timeout nach 24 Stunden Nutzung der Applikation *feelio*

USE CASE 4	**Registrierungsprozess**
Erläuterung	Der Nutzer kann zur Maske für die Kontoerstellung wechseln, indem er in der Anmeldemaske auf „SIGN UP“ klickt.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Unteraufgabe
Stakeholder und Interessen	Systembediener: Der Nutzer möchte das Analysetool *feelio* nutzen und möchte sich daher mit einem Konto registrieren.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer ist aktuell nicht in einer Sitzung oder befindet sich im ausgeloggten Zustand. • Der Nutzer befindet sich auf der Anmeldeseite „<IP-Adresse>:8080/User/signIn“ (Ergebnis von Use Case 3, siehe Tabelle A.1).
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Der Nutzer wird zur Registrierungsmaske weitergeleitet.
Auslöser	Der Nutzer besitzt noch kein Konto und möchte sich daher für das Analysetool registrieren.
Beschreibung	1. Der Nutzer klickt auf den „SIGN UP“-Button. 2. Der Nutzer wird zur Registrierungsmaske unter der Adresse „<IP-Adresse>:8080/User/signUp“ weitergeleitet.
Erweiterungen	-
Technologie	-
Priorität	wünschenswert
Verwendungshäufigkeit	regelmäßig

Tabelle A.4: Use Case zum Registrierungsprozess innerhalb der Applikation *feelio*

USE CASE 5	**Eingabe bei der Registrierung**
Erläuterung	Der Nutzer kann sich im Analysetool mit seiner E-Mail-Adresse und einem selbstgewählten Passwort registrieren.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Unteraufgabe
Stakeholder und Interessen	Systembediener: Der Nutzer möchte ein Konto für das Analysetool erstellen, um bei zukünftigen Sitzungen auf seine Daten zugreifen zu können.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer ist aktuell nicht in einer Sitzung oder befindet sich im ausgeloggten Zustand. • Der Nutzer befindet sich auf der Registrierungsseite (Ergebnis von Use Case 4, siehe Tabelle A.4).
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Der Nutzer hat erfolgreich ein Konto für das Analysetool angelegt und kann dieses Tool nutzen.
Auslöser	Der Nutzer beabsichtigt, ein Konto für das Analysetool anzulegen.
Beschreibung	1. Dem Nutzer wird ein Registrierungsformular mit Feldern für die E-Mail-Adresse und das Passwort präsentiert. 2. Der Nutzer trägt eine korrekt formatierte E-Mail-Adresse in das vorgesehene Feld ein, beispielsweise „contact@timokurtz.de“. 3. Der Nutzer gibt ein ausreichend sicheres Passwort, wie zum Beispiel „qufdy8-somkak-tumnEq“, in das Passwort-Feld ein. 4. Der Nutzer klickt auf den Button „SIGN UP“. 5. Bei erfolgreicher Registrierung wird der Nutzer automatisch zur Adresse „<IP-Adresse>:8080/App-Library“ weitergeleitet.

Tabelle A.5 Fortsetzung von vorheriger Seite

USE CASE 5	**Eingabe bei der Registrierung**
Erweiterungen	4.a. WENN das E-Mail-Feld nicht ausgefüllt wurde, DANN erscheint die Fehlermeldung „E-mail is required.“. 4.b. WENN die E-Mail-Adresse nicht korrekt formatiert ist, DANN erscheint die Fehlermeldung „E-mail must be valid.“. 4.c. WENN das Passwort-Feld nicht ausgefüllt wurde, DANN erscheint die Fehlermeldung „Password is required.“. 4.d. WENN das Passwort nicht den Anforderungen entspricht (mindestens ein Kleinbuchstabe, ein Großbuchstabe und mindestens eine Zahl), DANN erscheint die Fehlermeldung „Password must be valid. (Password must contain at least one lowercase letter, one uppercase letter, and one digit).“. 4.e. WENN bereits ein Konto mit dieser E-Mail-Adresse existiert, DANN erscheint die Fehlermeldung „This email address is already registered. Please use a different email address.“.
Technologie	-
Priorität	wünschenswert
Verwendungshäufigkeit	regelmäßig

Tabelle A.5: Use Case für die Eingabe bei der Registrierung in der Applikation *feelio*

USE CASE 6	**Beginn einer App-Analyse**
Erläuterung	Auf der Seite „App Library“ angekommen, besteht die Möglichkeit, eine neue App-Analyse zu starten.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Hauptaufgabe
Stakeholder und Interessen	Systembediener: Der Nutzer möchte die Reviews einer App seiner Wahl analysieren.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand. • Der Nutzer befindet sich auf der Seite „<IP-Adresse>:8080/AppLibrary“.
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Der Nutzer wird zum Formular für die Erstellung einer Analyse weitergeleitet.
Auslöser	Der Nutzer möchte Reviews für eine App analysieren.

Tabelle A.6 Fortsetzung von vorheriger Seite

USE CASE 6	**Beginn einer App-Analyse**
Beschreibung	1. Der Nutzer klickt auf den „+“-Button. 2. Der Nutzer wird zur Seite für die Erstellung einer Analyse weitergeleitet („<IP-Adresse>:8080/AppLibrary/ProjectCreation“). 3. Dort wird dem Nutzer ein Eingabeformular für App-Daten präsentiert, mit den möglichen Eingabetypen: CSV-Datei, Google Play Store und Apple App Store.
Erweiterungen	-
Technologie	-
Priorität	unverzichtbar
Verwendungshäufigkeit	ständig

Tabelle A.6: Use Case für den Start einer App-Analyse in der Applikation *feelio*

USE CASE 7	**Eingabe für den App-Analyse-Prozess**
Erläuterung	Der Nutzer kann zwischen drei verschiedenen Eingabequellen für die App-Analyse wählen: CSV-Datei, Google Play Store und Apple App Store.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Unteraufgabe
Stakeholder und Interessen	Systembenutzer: Der Nutzer möchte Reviews aus verschiedenen Datenquellen erfassen.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand. • Der Nutzer befindet sich auf der Seite „<IP-Adresse>:8080/AppLibrary/ProjectCreation“.
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Der Nutzer hat alle Informationen für seine App-Analyse eingegeben.
Auslöser	Der Nutzer möchte Reviews analysieren, die aus verschiedenen Datenquellen stammen.

Tabelle A.7 Fortsetzung von vorheriger Seite

USE CASE 7	**Eingabe für den App-Analyse-Prozess**
Beschreibung	1. Dem Nutzer werden drei Tabs zur Auswahl der App-Typen (CSV-Datei, Google Play Store und Apple App Store) präsentiert. 2. Je nachdem, welchen App-Typ der Nutzer auswählen möchte, kann er über die entsprechenden Tabs zur jeweiligen Eingabemaske wechseln. 3. Mit dem „+“-Button kann der Nutzer eine neue App-Analyse initiieren. 4. Nach erfolgreicher Eingabe (siehe Use Case) wird der Nutzer zur analysierten App unter der Adresse „<IP-Adresse>:8080/AppLibrary/App“ weitergeleitet.
Erweiterungen	4.a. WENN das Namensfeld im CSV-Tab befüllt ist („CSV App“) und keine CSV-Datei hochgeladen wurde, DANN erscheint die Fehlermeldung „Since you named a CSV file, you also need to upload a CSV file.“. 4.b. WENN das Namensfeld im CSV-Tab leer ist und eine CSV-Datei erfolgreich hochgeladen wurde, DANN erscheint die Fehlermeldung „Since you want to upload a CSV file, you need to give the project a name.“. 4.c. WENN das App-ID-Feld im Google Play Store Tab befüllt ist („com.Slack“) und keine Eingabe für das Feld „Review Count“ getätigt wurde, DANN erscheint die Fehlermeldung „Please enter a number.“. 4.d. WENN das App-ID-Feld im Google Play Store Tab nicht korrekt formatiert ist („.com.Slack“), DANN erscheint die Fehlermeldung „Please enter a valid Google Play Store App ID.“. 4.e. WENN das Feld „Review Count“ im Google Play Store Tab nicht korrekt formatiert ist („abc“), DANN erscheint die Fehlermeldung „Please enter a number.“.

Tabelle A.7 Fortsetzung von vorheriger Seite

USE CASE 7	**Eingabe für den App-Analyse-Prozess**
Erweiterungen	4.f. WENN im Apple App Store Tab das App-ID-Feld korrekt befüllt ist („618783545“) und keine Eingabe für „Review Count“ erfolgt ist, DANN erscheint die Fehlermeldung „Please enter a review count.“. 4.g. WENN das App-ID-Feld im Apple App Store Tab falsch formatiert ist („jdhs829“), DANN erscheint die Fehlermeldung „Invalid App ID. Ensure it’s a valid number.“. 4.h. WENN das App-Name-Feld im Apple App Store Tab leer ist, aber eine valide App-ID („618783545“) oder ein korrekter Wert für „Review Count“ („20“) eingegeben wurde, DANN erscheint die Fehlermeldung „Please provide an App Name.“. 4.i. WENN das Feld „Review Count“ im Apple App Store Tab falsch formatiert ist („abc“), DANN erscheint die Fehlermeldung „Ensure you enter a valid number.“.
Technologie	-
Priorität	unverzichtbar
Verwendungshäufigkeit	ständig

Tabelle A.7: Use Case für die Eingabemaske bei der App-Analyse in der Applikation *feelio*

USE CASE 8	**Upload einer CSV-Datei**
Erläuterung	Über den CSV-Tab kann der Nutzer eine Datei hochladen und dem CSV-App-Typ einen Namen zuweisen.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Hauptfunktion
Stakeholder und Interessen	Systembediener: Der Nutzer plant, eine CSV-Datei mit App-Review-Daten zur Analyse hochzuladen.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand. • Der Nutzer befindet sich auf der Seite „<IP-Adresse>:8080/AppLibrary/ProjectCreation“. • Der Nutzer hat den CSV-Tab geöffnet. • Der Nutzer verfügt über eine CSV-Datei, die mindestens die Spalten „review“, „rating“ und „date“ enthält.
Garantie	Das Backend bleibt in einem konsistenten Zustand, und die CSV-Datei wird nicht verändert.
Erfolgsfall	Der Nutzer lädt eine CSV-Datei hoch, gibt der App einen Namen und bekommt daraufhin eine Analyse der Reviews aus der CSV-Datei präsentiert.
Auslöser	Der Nutzer möchte Review-Daten einer App aus einer CSV-Datei analysieren.
Beschreibung	1. Der Nutzer gibt seiner CSV-Datei-Analyse den Namen „CSV Datei Test“. 2. Der Nutzer hat die Möglichkeit, aus mehreren vordefinierten Genres auszuwählen, z.B. „Games“ oder „Entertainment“ (Auswahl des Genres „Games“). 3. Der Nutzer lädt seine CSV-Datei über das Eingabeformular hoch. 4. Mit einem Klick auf den „+“-Button startet der Nutzer die App-Analyse. 5. Bei korrekter Eingabe wird der Nutzer zur analysierten App unter der Adresse „<IP-Adresse>:8080/AppLibrary/App“ weitergeleitet.

Tabelle A.8 Fortsetzung von vorheriger Seite

USE CASE 8	**Upload einer CSV-Datei**
Erweiterungen	3.a. WENN die hochgeladene Datei keine CSV-Datei ist, DANN erscheint die Fehlermeldung „The file you uploaded is not a CSV file.“. 3.b. WENN die CSV-Datei nicht die Spalten „review“, „rating“ und „date“ enthält, DANN erscheint die Fehlermeldung „The CSV file is missing the required fields:...“. 4.a. WENN im CSV-Tab ein Name eingegeben wurde (z. B. „CSV App“), aber keine CSV-Datei hochgeladen wurde, DANN erscheint die Fehlermeldung „Since you named a CSV file, you also need to upload a CSV file.“. 4.b. WENN im CSV-Tab kein Name eingegeben wurde, aber eine CSV-Datei erfolgreich hochgeladen wurde (z.B. „reviewsDemo.csv“), DANN erscheint die Fehlermeldung „Since you want to upload a CSV file, you need to give the project a name.“.
Technologie	-
Priorität	unverzichtbar
Verwendungshäufigkeit	ständig

Tabelle A.8: Use Case für den Upload einer CSV-Datei bei der App-Analyse in der Applikation *feelio*

USE CASE 9	**Analyse einer Google Play Store App**
Erläuterung	Der Nutzer kann im Tab „Google Play Store“ Kennzahlen eingeben, um eine App aus dem Google Play Store zu analysieren.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Hauptfunktion
Stakeholder und Interessen	Systembediener: Der Nutzer möchte die Reviews einer App aus dem Google Play Store analysieren.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand. • Der Nutzer befindet sich auf der Seite „<IP-Adresse>:8080/AppLibrary/ProjectCreation“. • Der Nutzer hat den Google Play Store Tab geöffnet.
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Der Nutzer erhält eine Analyse zu seiner App („Slack“).
Auslöser	Der Nutzer möchte Reviews einer Google Play Store App analysieren.

Tabelle A.9 Fortsetzung von vorheriger Seite

USE CASE 9	**Analyse einer Google Play Store App**
Beschreibung	1. Der Nutzer trägt die App-ID („com.Slack“) in das vorgesehene Feld „App-ID“ ein. 2. Im Feld „Review Count“ gibt der Nutzer die gewünschte Anzahl der Reviews („300“) ein. 3. Um die App-Analyse zu starten, klickt der Nutzer auf den „+“-Button. 4. Nach erfolgreicher Eingabe wird er zur analysierten App unter „<IP-Adresse>:8080/AppLibrary/App“ weitergeleitet.
Erweiterungen	4.a. WENN im Tab Google Play Store eine „App-ID“ („com.Slack“) eingegeben wurde, jedoch das Feld „Review Count“ nicht ausgefüllt ist, DANN wird die Fehlermeldung „Please enter a number“ angezeigt. 4.b. WENN die „App-ID“ im Tab Google Play Store nicht korrekt formatiert ist („.com.Slack“), DANN erscheint die Fehlermeldung „Please enter a valid Google Play Store App ID.“. 4.c. WENN das Feld „Review Count“ im Tab Google Play Store falsch formatiert ist („abc“), DANN zeigt das System die Fehlermeldung „Please enter a number.“ an.
Technologie	-
Priorität	unverzichtbar
Verwendungshäufigkeit	ständig

Tabelle A.9: Use Case für die Analyse einer Google Play Store App in der Applikation *feelio*

USE CASE 10	**Analyse einer Apple App Store App**
Erläuterung	Der Nutzer kann im Tab „Apple App Store“ Kennzahlen zur Analyse einer App aus dem Apple App Store eingeben.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Hauptfunktion
Stakeholder und Interessen	Systembediener: Der Nutzer beabsichtigt, die Reviews einer App aus dem Apple App Store zu analysieren.

Tabelle A.10 Fortsetzung von vorheriger Seite

USE CASE 10	**Analyse einer Apple App Store App**
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand. • Der Nutzer befindet sich auf der Seite „<IP-Adresse>:8080/AppLibrary/ProjectCreation“. • Der Nutzer hat den Apple App Store Tab geöffnet.
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Der Nutzer erhält eine Analyse für seine App („Slack“).
Auslöser	Der Nutzer möchte App-Review-Daten einer Apple App Store App analysieren.
Beschreibung	1. Der Nutzer trägt die App-ID („618783545“) in das entsprechende Feld „App-ID“ ein. 2. Der Nutzer gibt die Anzahl der Reviews („300“) in das Feld „Review Count“ an. 3. Durch Klicken auf den „+“-Button startet der Nutzer die App-Analyse. 4. Bei korrekter Eingabe erfolgt eine automatische Weiterleitung zur analysierten App unter „<IP-Adresse>:8080/AppLibrary/App“.
Erweiterungen	4.a. WENN im „Apple App Store“-Tab eine App-ID („618783545“) eingegeben, aber keine Review-Anzahl angegeben wurde, DANN erscheint die Fehlermeldung „Please enter a review count.“. 4.b. WENN im „Apple App Store“-Tab eine nicht korrekt formatierte App-ID („jdhs829“) eingegeben wurde, DANN erscheint die Fehlermeldung „Invalid App ID. Please enter a valid number.“. 4.c. WENN das „App-Name“-Feld im „Apple App Store“-Tab leer ist, obwohl entweder eine App-ID („618783545“) oder eine Review-Anzahl („20“) angegeben wurde, DANN erscheint die Fehlermeldung „Please enter an App Name.“. 4.d. WENN das Feld „Review Count“ im „Apple App Store“-Tab nicht korrekt formatiert wurde, DANN erscheint die Fehlermeldung „Please enter a number.“.
Technologie	-
Priorität	unverzichtbar
Verwendungshäufigkeit	ständig

Tabelle A.10: Use Case für die Analyse einer Apple App Store App in der Applikation *feelio*

USE CASE 11	**Darstellung der Review-Analyse**
Erläuterung	Der Nutzer kann ausgewählte Apps nach verschiedenen Kriterien analysieren.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Hauptfunktion
Stakeholder und Interessen	Systembediener: Der Nutzer beabsichtigt, die Bewertungen einer App zu analysieren.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand. • Der Nutzer hat in der Analysemaske folgende Eingaben vorgenommen: (CSV-Typ: App-Name „CSV-Test 1“, CSV-Demo-Datei hochgeladen), (Google Play Store: App-ID „com.Slack“, Review-Anzahl: 300), (Apple App Store: App-ID „618783545“, App-Name „Slack“, Review-Anzahl: 300). • Der Nutzer befindet sich auf der Seite „<IP-Adresse>:8080/AppLibrary/App“.
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Der Nutzer bekommt eine Analyse der von ihm zuvor ausgewählten App-Art und hat die Möglichkeit, verschiedene App-Typen miteinander zu vergleichen.
Auslöser	Der Nutzer möchte App-Reviews analysieren.
Beschreibung	1. Das System präsentiert dem Nutzer vier Tabs: „Comparison“, „Apple App Store“, „Google Play Store“ und „CSV Datei“. 2. Der Nutzer kann zwischen diesen Tabs wechseln und so die Analysen der verschiedenen App-Typen einsehen. 3. In jedem App-Typ werden dem Nutzer die Hauptkategorien „Statistics“, „Screenshots“, „Reviews“ und „Charts“ dargestellt.
Erweiterungen	3.a. Unter „Statistics“ erhält der Nutzer Informationen über die Durchschnittsbewertung, die durchschnittliche Länge der Reviews, das durchschnittliche Sentiment und den Zeitraum der erfassten Reviews. 3.b. Im Bereich „Screenshots“ werden die Screenshots präsentiert, die direkt aus dem jeweiligen App Store stammen. 3.c. Im Abschnitt „Reviews“ sind alle Reviews des ausgewählten App-Typs einsehbar. 3.d. Unter „Charts“ findet der Nutzer diverse Grafiken, darunter beispielsweise eine Visualisierung der Sentimentverteilung.

Tabelle A.11 Fortsetzung von vorheriger Seite

USE CASE 11	**Darstellung der Review-Analyse**
Technologie	-
Priorität	unverzichtbar
Verwendungshäufigkeit	ständig

Tabelle A.11: Use Case für die Darstellung einer Review-Analyse in der Applikation *feelio*

USE CASE 12	**Filteroptionen innerhalb der Review-Analyse**
Erläuterung	Der Nutzer kann auf der Analyse-Seite der App verschiedene Filter setzen, um seine App detailliert zu analysieren.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Hauptfunktion
Stakeholder und Interessen	Der Nutzer möchte die Reviews seiner App durch den Einsatz verschiedener Filter untersuchen.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand. • Der Nutzer hat in der Analysemaske folgende Eingaben vorgenommen: (CSV-Typ: App-Name „CSV-Test 1“, CSV-Demo-Datei hochgeladen), (Google Play Store: App-ID „com.Slack“, Review-Anzahl: 300), (Apple App Store: App-ID „618783545“, App-Name „Slack“, Review-Anzahl: 300). • Der Nutzer befindet sich auf der Seite „<IP-Adresse>:8080/AppLibrary/App“.
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Je nach den gewählten Filtern aktualisieren sich die Werte in den Bereichen „Statistics“, „Reviews“ und „Charts“ in Echtzeit.
Auslöser	Der Nutzer möchte eine App basierend auf den ausgewählten Filtern analysieren.
Beschreibung	1. Der Nutzer klickt auf den Button „Filter“. 2. Ein Menü mit verschiedenen Filteroptionen öffnet sich. 3. Durch Antippen eines Filters wird die zugehörige Checkbox aktiviert. 4. Hierbei öffnet sich ein detailliertes Untermenü, das spezifische Filtereinstellungen anbietet. 5. Nachdem ein Filter ausgewählt und dessen Einstellungen angepasst wurden, aktualisieren sich die Inhalte in den Bereichen „Statistics“, „Reviews“ und „Charts“ in Echtzeit.

Tabelle A.12 Fortsetzung von vorheriger Seite

USE CASE 12	**Filteroptionen innerhalb der Review-Analyse**
Erweiterungen	5.a. WENN kein Filter angewendet wird, DANN bleibt die Anzahl der Reviews unverändert.
Technologie	-
Priorität	unverzichtbar
Verwendungshäufigkeit	ständig

Tabelle A.12: Use Case für die Möglichkeiten des Filterns in der Review-Analyse der Applikation *feelio*

USE CASE 13	**Sortieroptionen innerhalb der Review-Analyse**
Erläuterung	Der Nutzer kann auf der App-Analyseseite Reviews nach verschiedenen Kategorien sortieren.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Hauptfunktion
Stakeholder und Interessen	Systembediener: Der Nutzer möchte seine Reviews übersichtlicher darstellen, indem er sie nach einer gewählten Kategorie aufsteigend oder absteigend ordnet.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand. • Der Nutzer hat in der Analysemaske folgende Eingaben vorgenommen: (CSV-Typ: App-Name „CSV-Test 1“, CSV-Demo-Datei hochgeladen), (Google Play Store: App-ID „com.Slack“, Review-Anzahl: 300), (Apple App Store: App-ID „618783545“, App-Name „Slack“, Review-Anzahl: 300). • Der Nutzer befindet sich auf der Seite „<IP-Adresse>:8080/AppLibrary/App“.
Garantie	Das Backend bleibt konsistent.
Erfolgsfall	Je nach gewählter Sortieroption des Nutzers ändert sich die Reihenfolge der Reviews im Bereich „Reviews“ in Echtzeit.
Auslöser	Der Nutzer möchte Reviews nach verschiedenen Sortierkriterien anordnen.

Tabelle A.13 Fortsetzung von vorheriger Seite

USE CASE 13	**Sortieroptionen innerhalb der Review-Analyse**
Beschreibung	1. Der Nutzer klickt auf den Button „Sort“. 2. Es öffnet sich ein Menü mit verschiedenen Sortierkategorien, die vom Nutzer ausgewählt werden können. 3. Bei Auswahl einer Sortierkategorie wird diese sofort aktiviert. 4. Sobald eine Sortierung ausgewählt wurde, werden die Reviews im Bereich „Reviews“ in Echtzeit entsprechend sortiert.
Erweiterungen	3.a. WENN eine Sortierkategorie aktiv ist, DANN wird diese durch einen Haken markiert. 3.b. WENN die gleiche Sortierkategorie erneut gewählt wird, DANN wechselt die Sortierreihenfolge (zwischen aufsteigend und absteigend).
Technologie	-
Priorität	unverzichtbar
Verwendungshäufigkeit	ständig

Tabelle A.13: Use Case für die Sortiermöglichkeiten der Reviews in der Review-Analyse der Applikation *feelio*

USE CASE 14	**Allgemeine Navigation**
Erläuterung	Der Nutzer hat über die obere Navigationsleiste Zugriff auf die drei Hauptseiten: „HOME“, „APPLICATIONS“ und „PROFILE“. Wählt er „HOME“ aus, gelangt er zu einem Vergleich aller hinzugefügten Apps. Unter „APPLICATIONS“ findet er eine Übersicht der zuvor erstellten App-Analysen. Durch Klick auf „PROFILE“ gelangt er zur Kontoübersicht, in der er sich abmelden kann.
Systemgrenzen	Gesamtsystem
Hauptakteur	Systembediener
Ebene	Hauptfunktion
Stakeholder und Interessen	Systembediener: Wählt der Nutzer die „HOME“-Webseite aus, möchte er einen Vergleich aller Apps sehen. Bei Auswahl von „APPLICATIONS“ wünscht er sich eine Übersicht seiner App-Analysen. Entscheidet er sich für „PROFILE“, erwartet er eine Kontoübersicht.
Vorbedingung	• Das Backend und das Frontend wurden erfolgreich gestartet. • Der Nutzer befindet sich im angemeldeten Zustand.
Garantie	Das Backend bleibt konsistent.

Tabelle A.14 Fortsetzung von vorheriger Seite

USE CASE 14	**Allgemeine Navigation**
Erfolgsfall	Der Nutzer wird zum jeweiligen Menüpunkt weitergeleitet.
Auslöser	Der Nutzer möchte Reviews von Apps analysieren.
Beschreibung	1. Der Nutzer wählt einen Menüpunkt (entweder „HOME“, „APPLICATIONS“ oder „PROFILE“) aus. 2. Er wird zur entsprechenden Seite weitergeleitet.
Erweiterungen	2.a. WENN der Nutzer „HOME“ auswählt, DANN wird er zur Seite „<IP-Adresse>:8080“ weitergeleitet. 2.b. WENN der Nutzer „APPLICATIONS“ auswählt, DANN wird er zur Seite „<IP-Adresse>:8080/AppLibrary“ weitergeleitet. 2.c. WENN der Nutzer „PROFILE“ auswählt, DANN wird er zur Seite „<IP-Adresse>:8080/User“ weitergeleitet.
Technologie	-
Priorität	unverzichtbar
Verwendungshäufigkeit	ständig

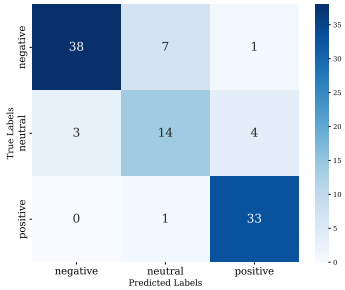
Tabelle A.14: Use Case für die allgemeine Navigation innerhalb der Applikation *feelio*

A.2 Feinabgestimmtes Sentimentanalysemodell

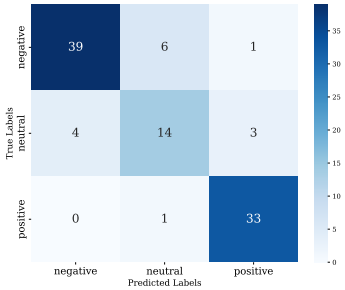
A.2.1 Datensatzerhebung

Abbildung A.1: Auswahl der Apps, deren Reviews für die Annotation verwendet wurden

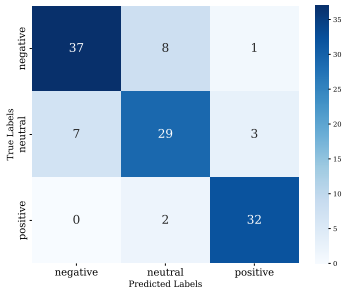
A.2.2 Performance



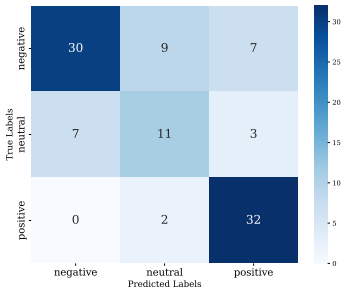
(a) Standard



(b) Loss Function



(c) Oversampling



(d) Twitter-Roberta-Base

Abbildung A.2: Vergleich der Konfusionsmatrizen der unterschiedlichen Konfigurationen

A.3 Stichprobenanalyse

A.3.1 Datensatzerhebung

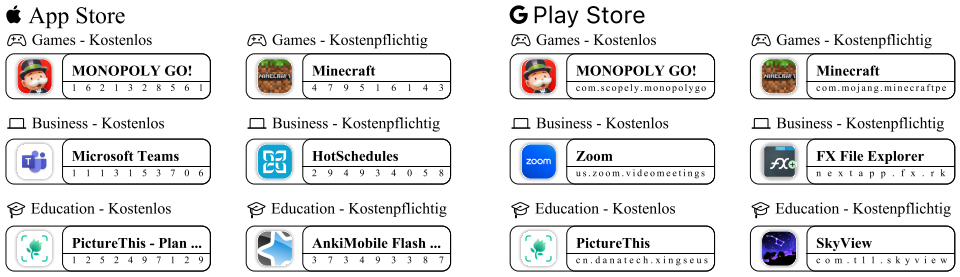
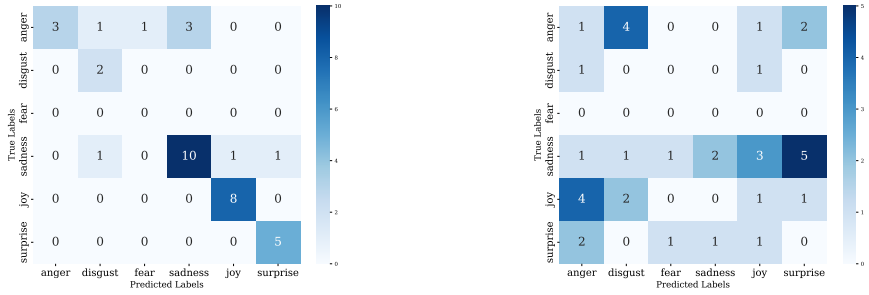


Abbildung A.3: Auswahl der Apps, deren Reviews für die Evaluation verwendet wurden

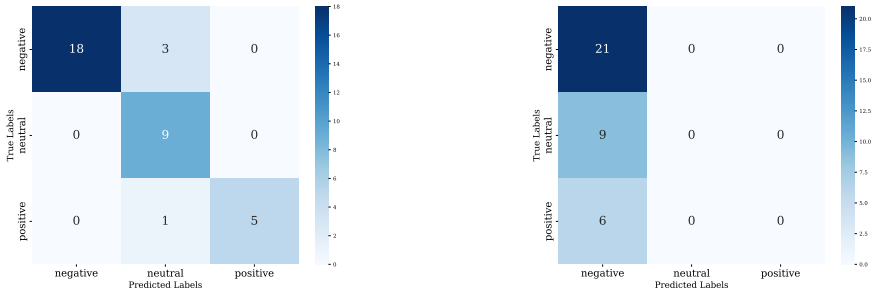
A.3.2 Performance



(a) Modellergebnisse

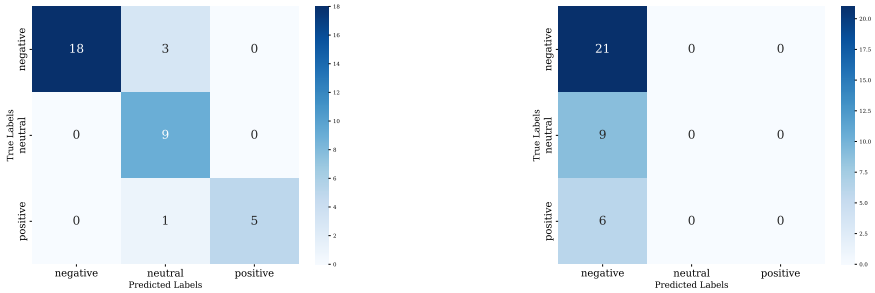
(b) Baseline

Abbildung A.4: Vergleich der Konfusionsmatrizen für die Emotionsanalyse (Klassifizierer: „Emotion“)



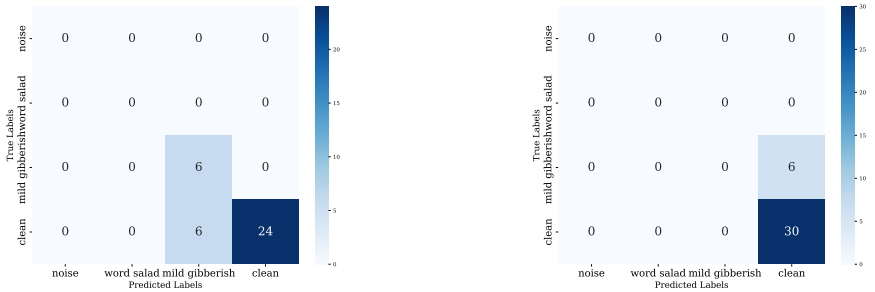
(a) Modellergebnisse (b) Baseline

Abbildung A.5: Vergleich der Konfusionsmatrizen für die feinabgestimmte Sentimentanalyse (Klassifizierer: „Sentiment“)



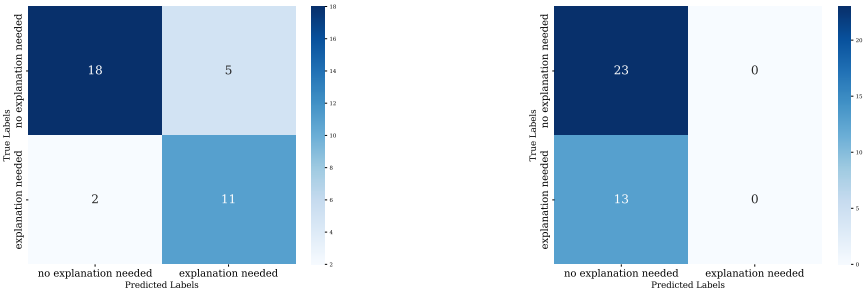
(a) Modellergebnisse (b) Baseline

Abbildung A.6: Vergleich der Konfusionsmatrizen für die „SentiStrength“-Analyse (Klassifizierer: „SentiStrength“)



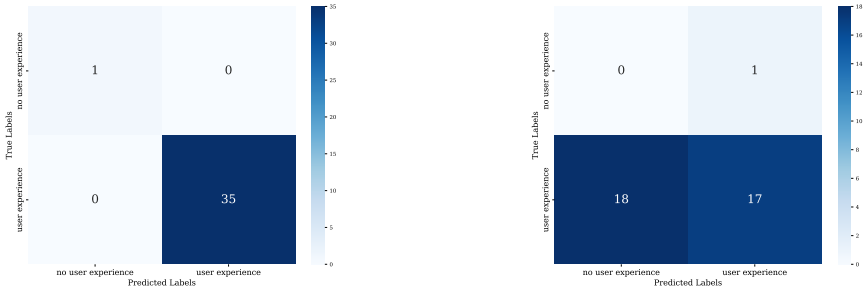
(a) Modellergebnisse (b) Baseline

Abbildung A.7: Vergleich der Konfusionsmatrizen für die Qualitätsanalyse (Klassifizierer: „Gibberish“)



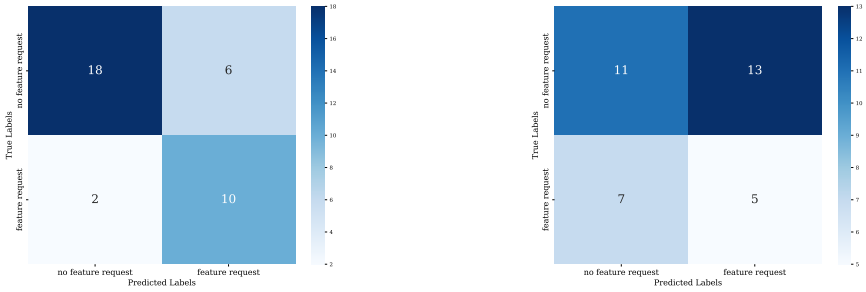
(a) Modellergebnisse (b) Baseline

Abbildung A.8: Vergleich der Konfusionsmatrizen für die Erklärungsbedarfsanalyse (Klassifizierer: „ExplanationNeeded“)



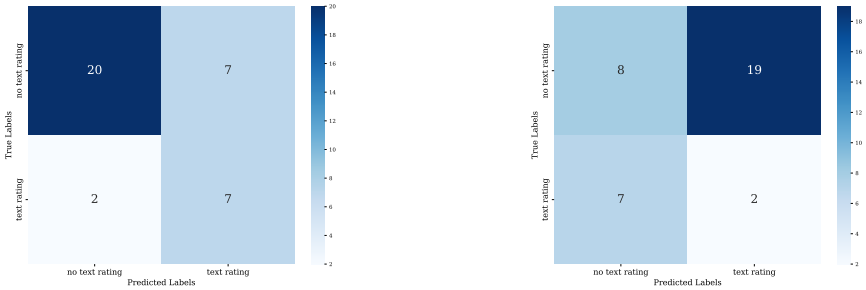
(a) Modellergebnisse (b) Baseline

Abbildung A.9: Vergleich der Konfusionsmatrizen für das Attribut „user experience“ der Eigenschaftsanalyse (Klassifizierer: „Characteristic“)



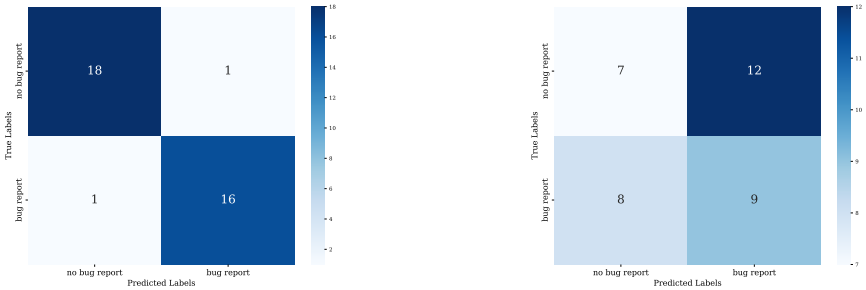
(a) Modellergebnisse (b) Baseline

Abbildung A.10: Vergleich der Konfusionsmatrizen für das Attribut „feature request“ der Eigenschaftsanalyse (Klassifizierer: „Characteristic“)



(a) Modellergebnisse (b) Baseline

Abbildung A.11: Vergleich der Konfusionsmatrizen für das Attribut „text rating“ der Eigenschaftsanalyse (Klassifizierer: „Characteristic“)



(a) Modellergebnisse (b) Baseline

Abbildung A.12: Vergleich der Konfusionsmatrizen für das Attribut „bug report“ der Eigenschaftsanalyse (Klassifizierer: „Characteristic“)

A.3.3 Hugging Face Ranglisten

Klassifizierer aus der Kategorie „Text Classification“

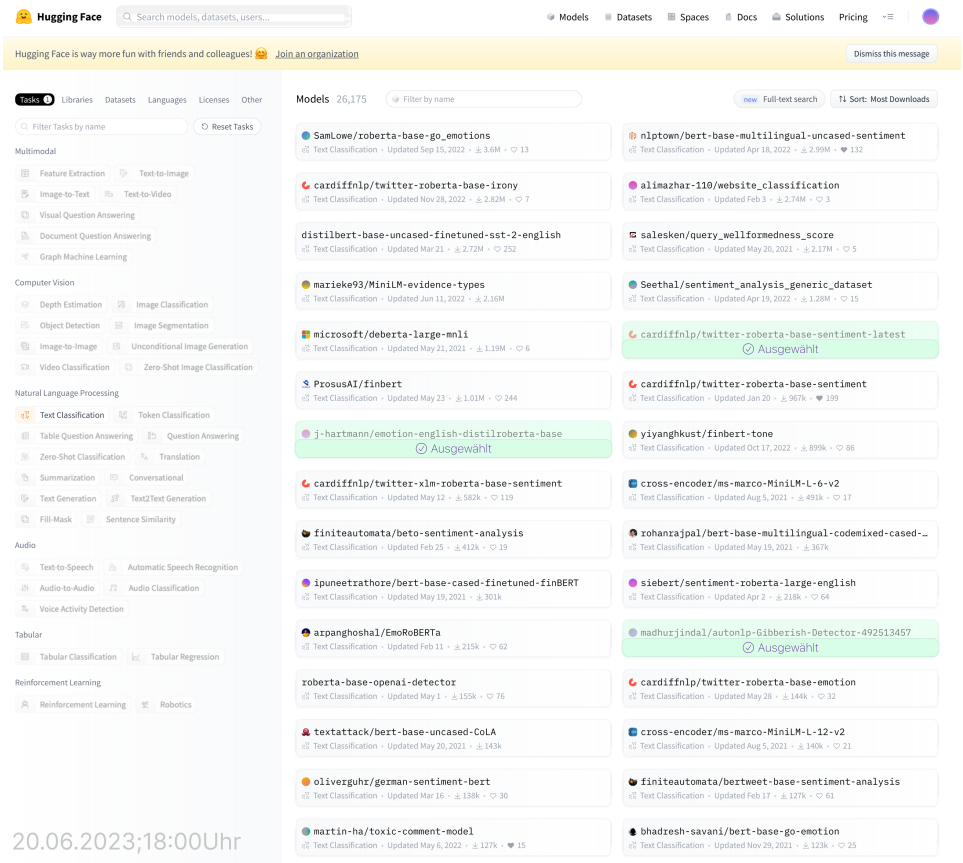


Abbildung A.13: Auswahl der Klassifizierer innerhalb der Kategorie „Text Classification“, die zur Klassifikation der Reviews herangezogen wurden (Stand: 20.06.2023, 18:00 Uhr)

Klassifizierer für die Emotionsanalyse

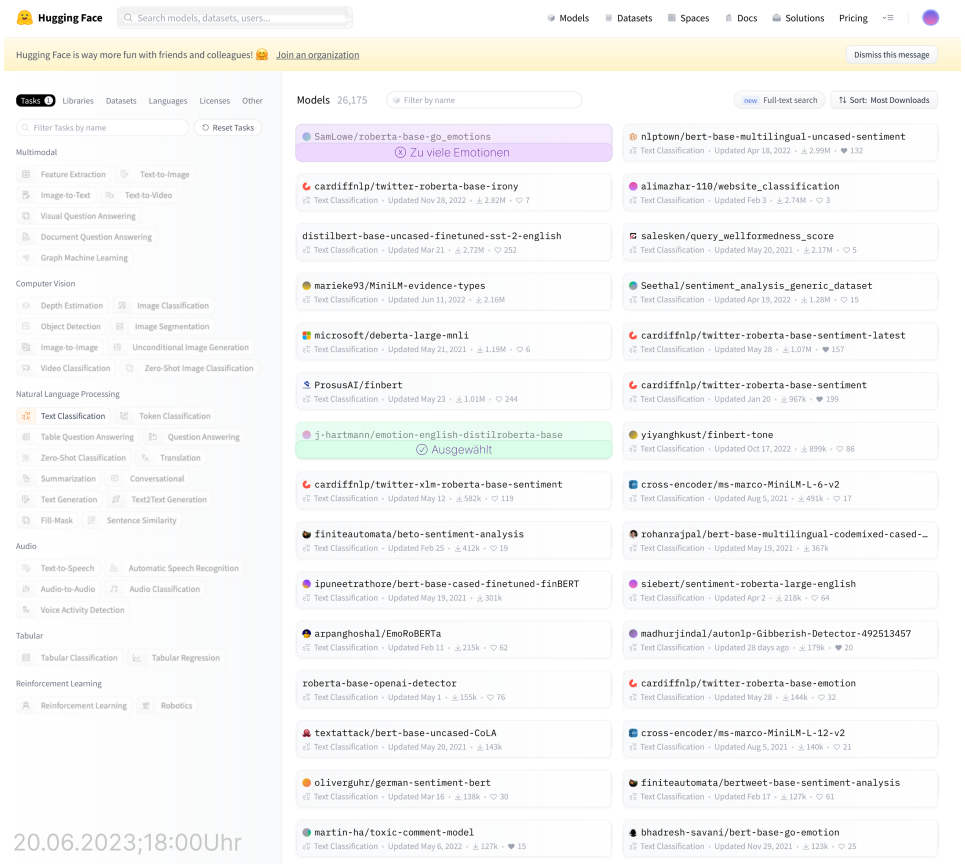


Abbildung A.14: Auswahl des Klassifizierers für die Emotionsanalyse (Stand: 20.06.2023, 18:00 Uhr)

Klassifizierer für die feinabgestimmte Sentimentanalyse

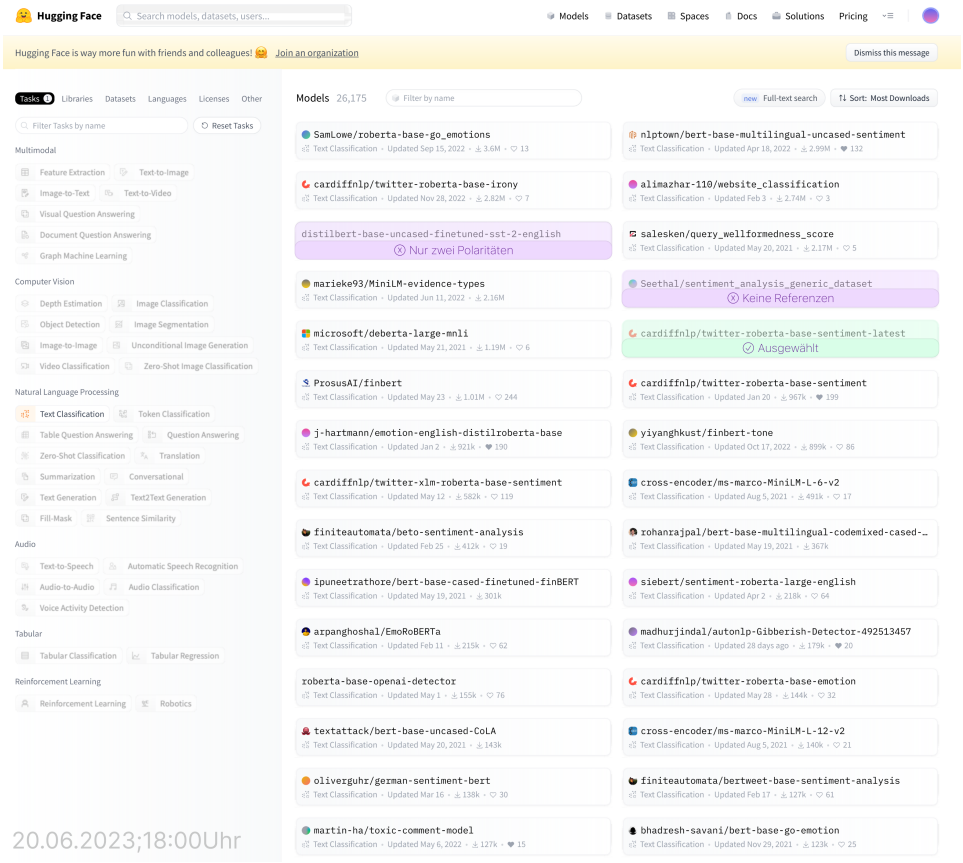


Abbildung A.15: Auswahl des Klassifizierers für das Ausgangsmodell der feinabgestimmten Sentimentanalyse (Stand: 20.06.2023, 18:00 Uhr)

Klassifizierer für die Qualitätsanalyse

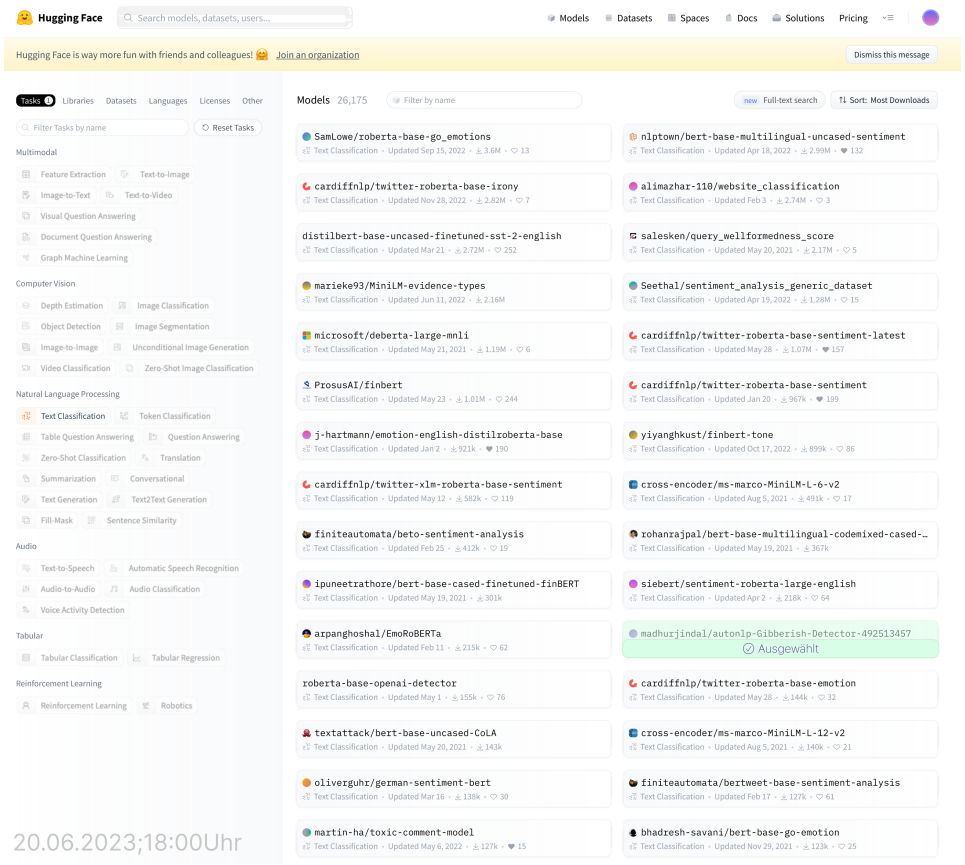


Abbildung A.16: Auswahl des Klassifizierers für die Qualitätsanalyse (Stand: 20.06.2023, 18:00 Uhr)

Klassifizierer für die Eigenschaftsanalyse

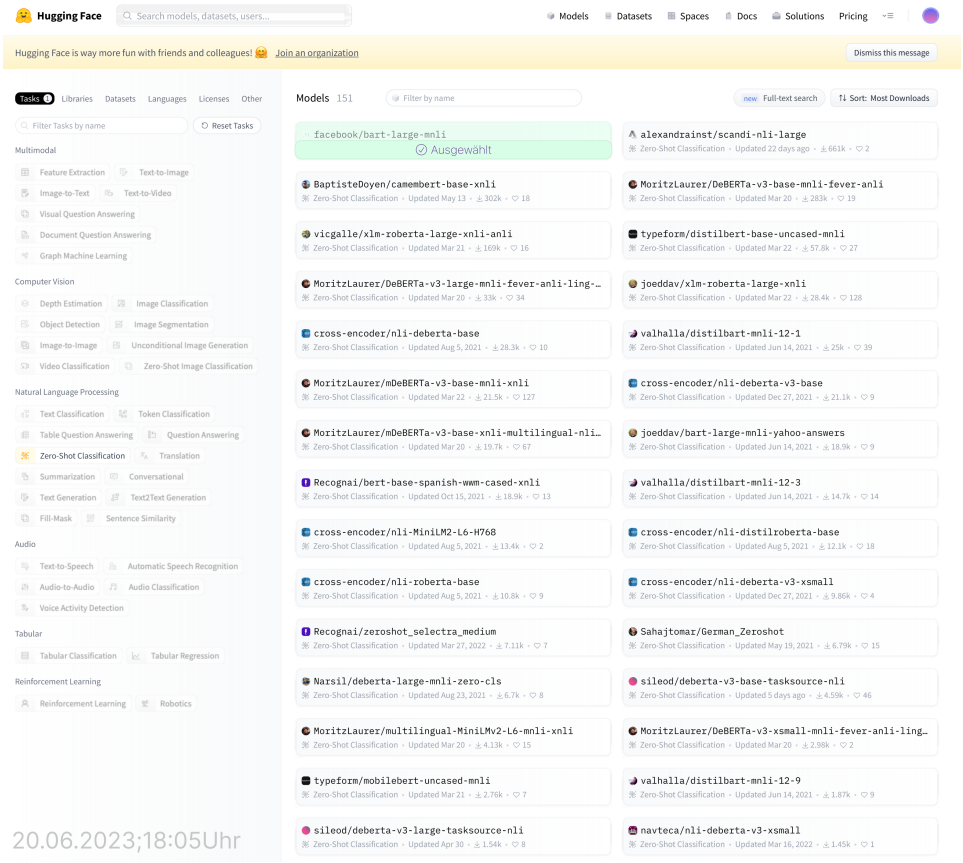


Abbildung A.17: Auswahl des Klassifizierers für die Eigenschaftsanalyse (Stand: 20.06.2023, 18:05 Uhr)

A.4 Datenbank Schema

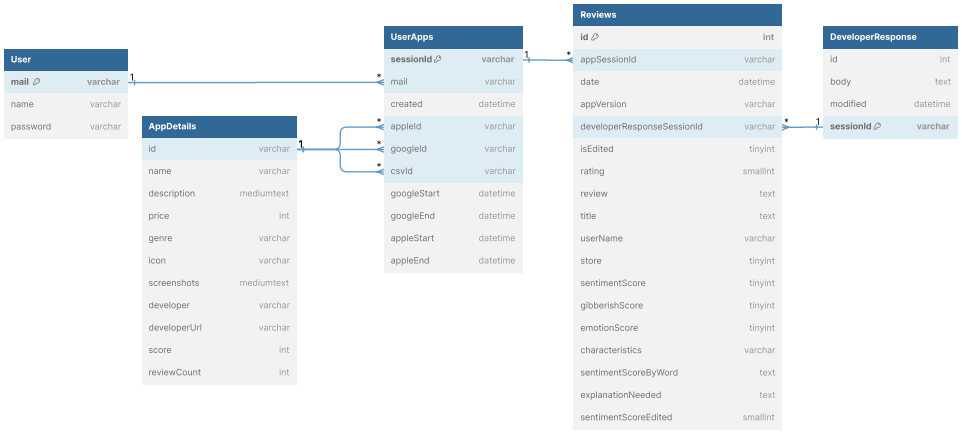


Abbildung A.18: ER-Diagramm der Anwendung *feelio* mit den Tabellen User, UserApps, AppDetails, Reviews, DeveloperResponse

A.5 Produktpräsentation

A.5.1 Video

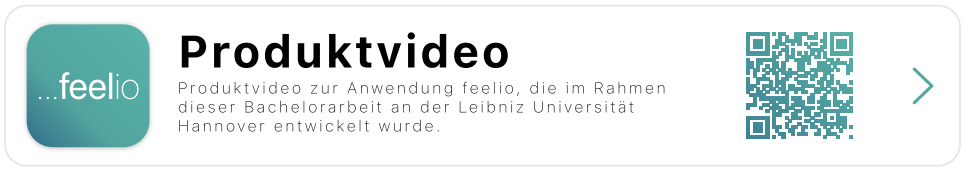


Abbildung A.19: Produktvideo für die neu entwickelte Anwendung *feelio* Analyse von App-Reviews

A.5.2 Interaktiver Prototyp

Abbildung A.20: Interaktiver Prototyp für die neu entwickelte Anwendung *feelio* Analyse von App-Reviews

A.5.3 Screenshots

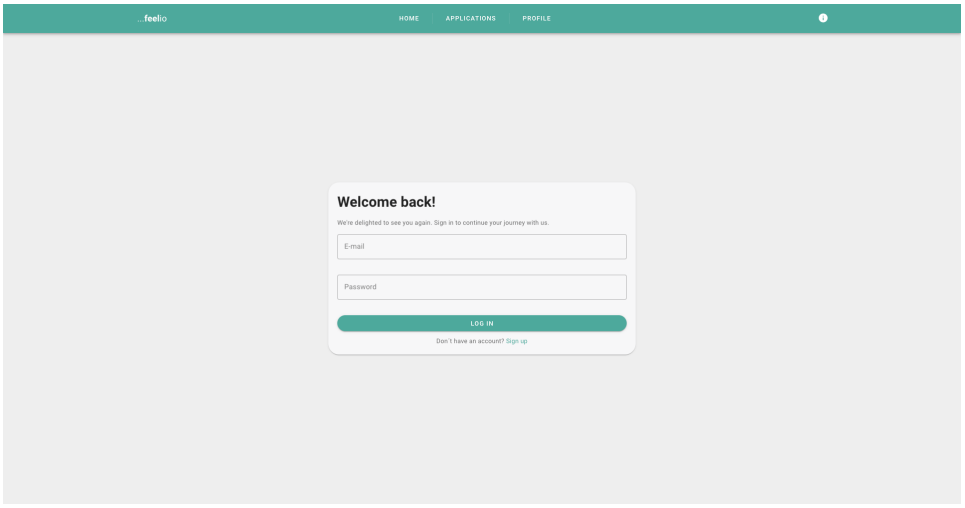


Abbildung A.21: Anmeldemaske innerhalb der Anwendung *feelio*

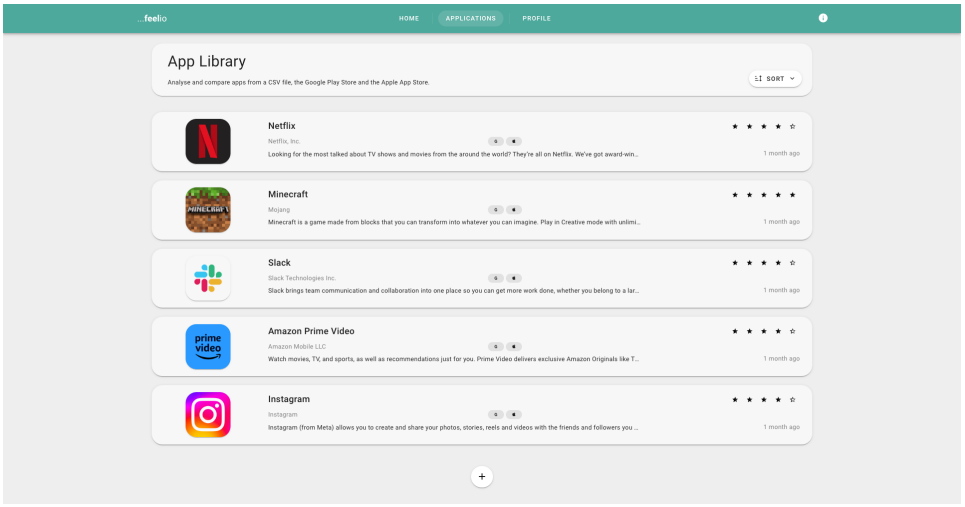


Abbildung A.22: Übersicht aller für die Analyse kreierten Apps innerhalb der Anwendung *feelio*

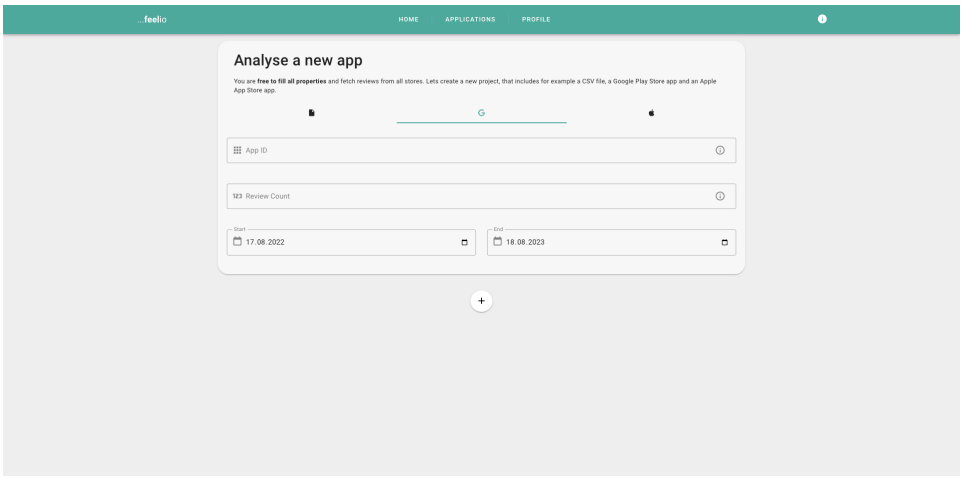


Abbildung A.23: Erstellung einer neuen App-Analyse innerhalb der Anwendung *feelio*

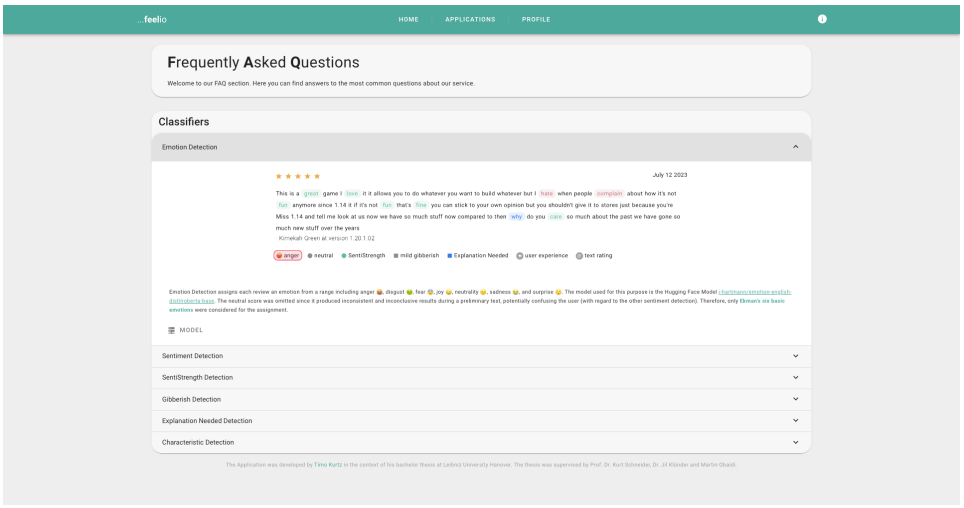


Abbildung A.24: FAQ („Frequently Asked Questions“) innerhalb der Anwendung *feelio*

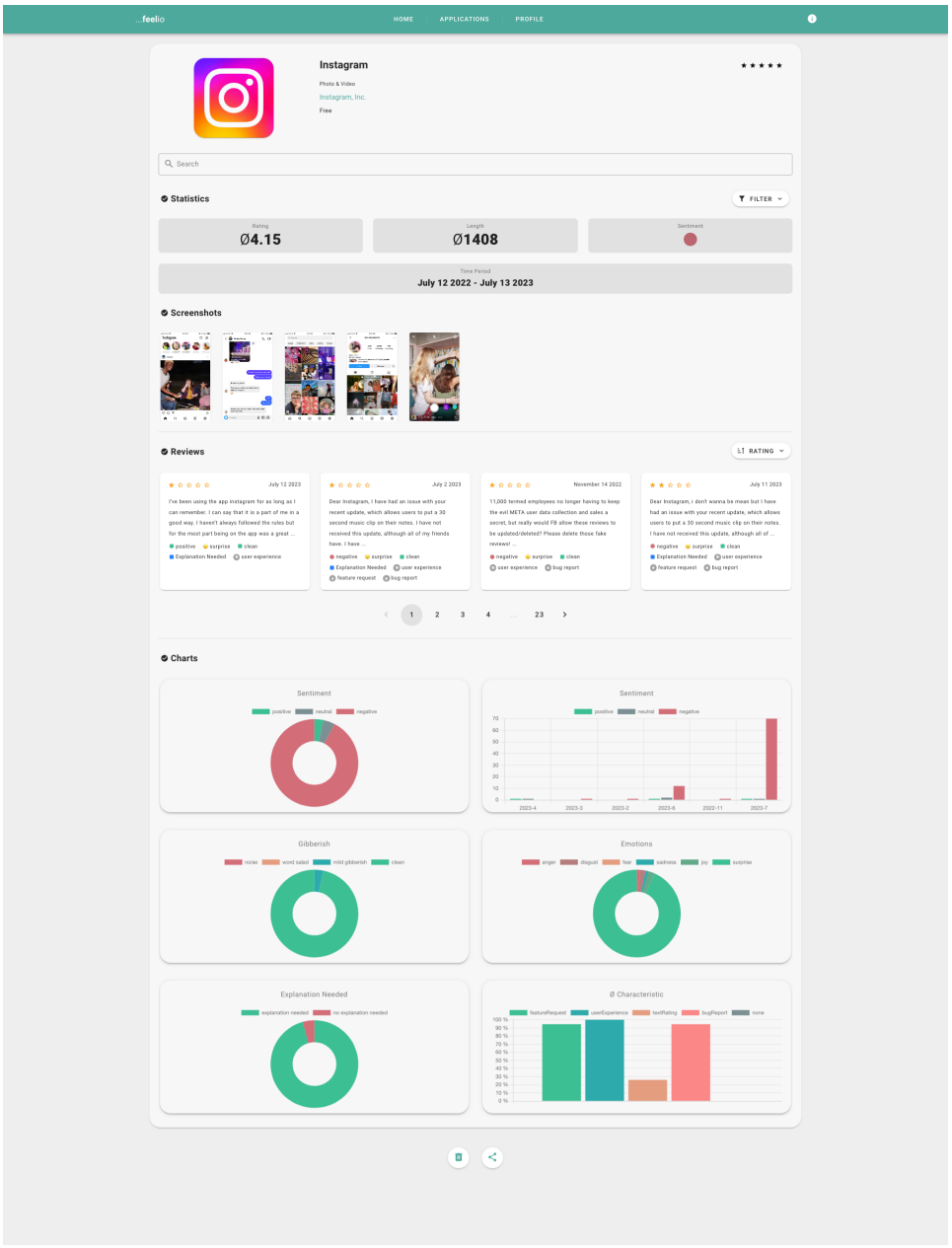


Abbildung A.25: Detailansicht für eine App-Analyse innerhalb der Anwendung *feelio*

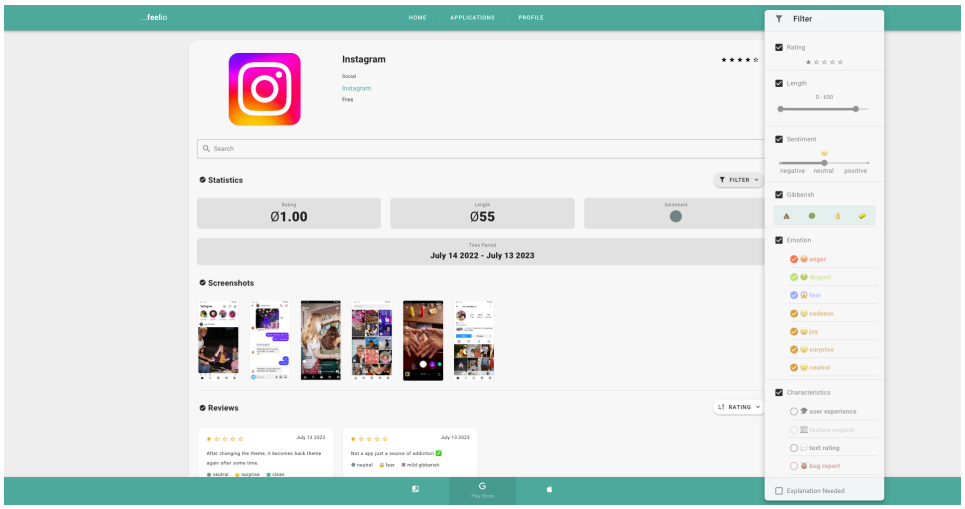


Abbildung A.26: Filteroptionen für die App-Analyse innerhalb der Anwendung *feelio*

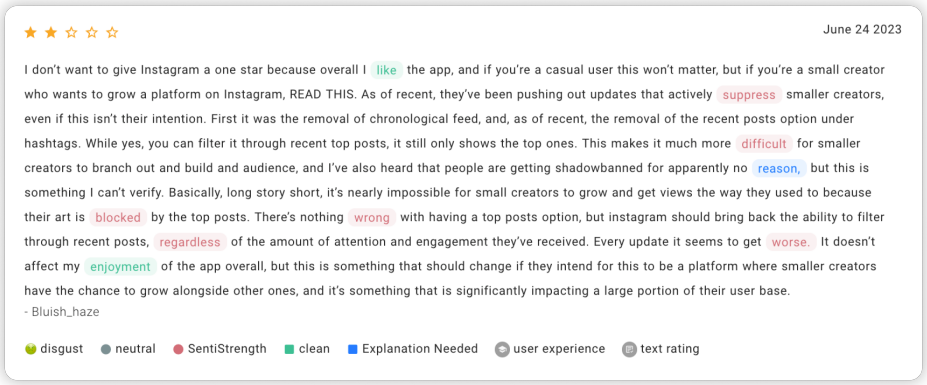


Abbildung A.27: Detailansicht eines Reviews inklusive seiner Klassifikation innerhalb der Anwendung *feelio*

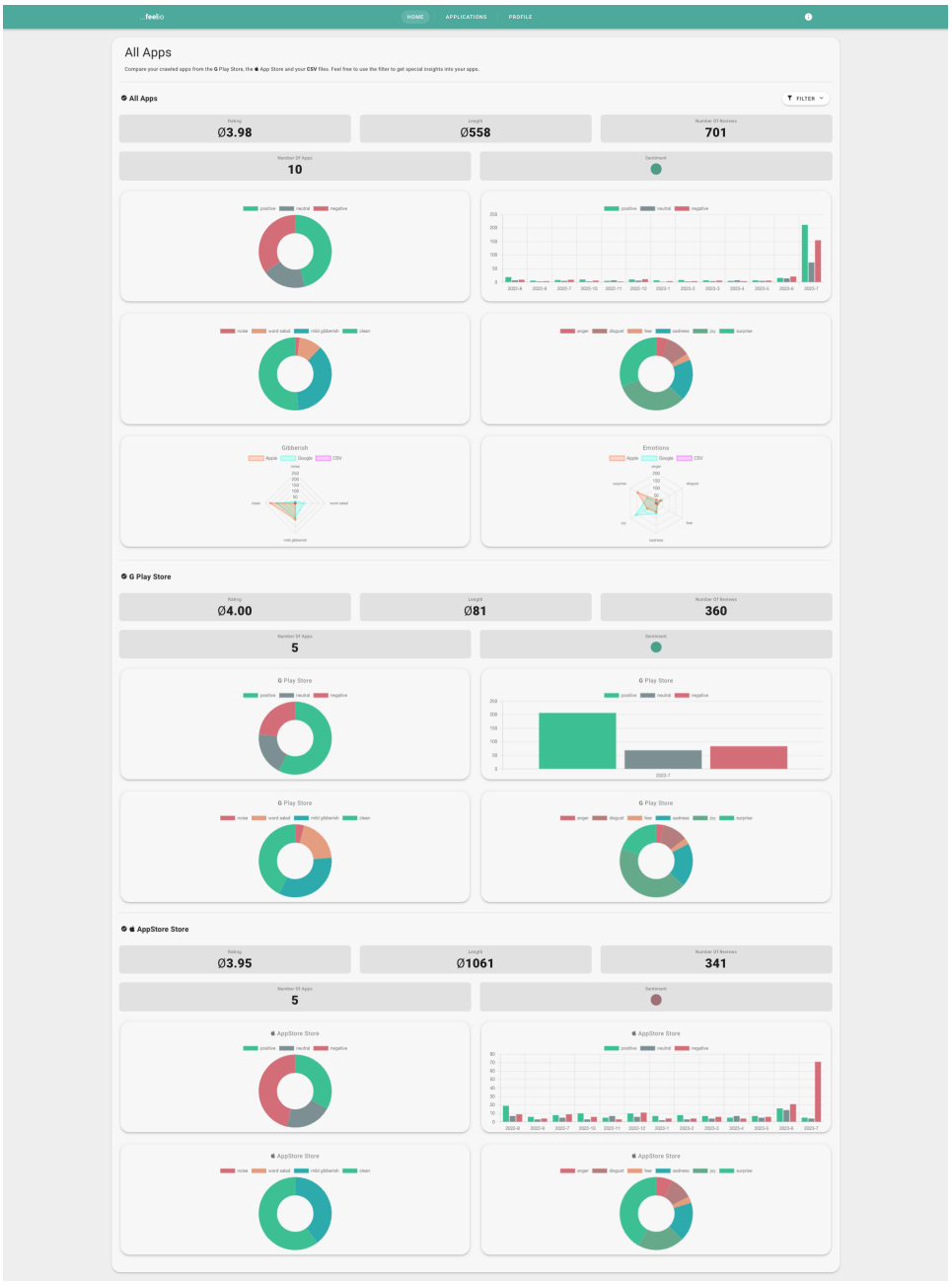


Abbildung A.28: Vergleich aller Reviews, die innerhalb der unterschiedlichen Datenquellen (App Stores) verglichen werden können

Literaturverzeichnis

[1] Appfigures. Global google play and apple app store app revenues 2022. Zitiert nach de.statista.com Abruf am 26.07.2023, 20.00 Uhr `https://de.statista.com/statistik/daten/studie/208599/umfrage/anzahl-der-apps-in-den-top-app-stores`.

[2] Appfigures. Total apps by quarter-category 3q2022. Zitiert nach de.statista.com Abruf am 02.05.2023, 18.00 Uhr `https://de.statista.com/statistik/daten/studie/208599/umfrage/anzahl-der-apps-in-den-top-app-stores/`.

[3] E. M. Bender. 100 things you always wanted to know about semantics & pragmatics but were afraid to ask. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, page 1, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[4] S. Butt, S. Sharma, R. Sharma, G. Sidorov, and A. Gelbukh. What goes on inside rumour and non-rumour tweets and their reactions: A psycholinguistic analyses. *Computers in Human Behavior*, 135:107345, 2022.

[5] P. Buxmann and H. Schmidt. *Künstliche Intelligenz: Mit Algorithmen zum wirtschaftlichen Erfolg*. Springer-Verlag, 01 2019.

[6] L. Cen, L. Si, N. Li, and H. Jin. User comment analysis for android apps and cspi detection with comment expansion. *CEUR Workshop Proceedings*, 1225:25–30, 01 2014.

[7] Chai.js. Assertion library for node. Zitiert nach scikit-learn.org Abruf am 31.07.2023, 23.00 Uhr `https://www.chaijs.com`.

[8] M. Chemuturi. *Requirements Engineering and Management for Software Development Projects*. SpringerLink : Bücher. Springer New York, 2012.

[9] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang. Ar-miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on*

Software Engineering, ICSE 2014, page 767–778, New York, NY, USA, 2014. Association for Computing Machinery.

[10] Cowboy-Bebug. App-Store-Scraper Python. Zitiert nach github.com Abruf am 07.08.2023, 15.00 Uhr `https://github.com/cowboy-bebug/app-store-scraper`.

[11] M. Del Tredici, R. Fernández, and G. Boleda. Short-term meaning shift: A distributional exploration. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2069–2075, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[13] Devoteam. Digital transformation | empowering businesses. Zitiert nach devoteam.com Abruf am 15.08.2023, 20.00 Uhr `https://www.devoteam.com`.

[14] V. T. Dhinakaran, R. Pulle, N. Ajmeri, and P. K. Murukannaiah. App review analysis via active learning: Reducing supervision effort without compromising classification accuracy. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 170–181, 2018.

[15] J. Ding, H. Sun, X. Wang, and X. Liu. Entity-level sentiment analysis of issue comments. In *2018 IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pages 7–13, 2018.

[16] Docker Inc. The most-used tool in stack overflow's 2023 developer survey. Zitiert nach docker.com Abruf am 07.08.2023, 15.00 Uhr `https://www.docker.com`.

[17] P. Ekman, W. Friesen, M. O'Sullivan, A. Chan, I. Diacoyanni-Tarlatzis, K. Heider, R. Krause, W. LeCompte, T. Pitcairn, and P. Ricci Bitti. Universals and cultural differences in the judgments of facial expressions of emotion. *Journal of personality and social psychology*, 53:712–7, 11 1987.

[18] Facundoolano. App-Store-Scraper Node.js. Zitiert nach github.com Abruf am 07.08.2023, 15.00 Uhr `https://github.com/facundoolano/app-store-scraper`.

[19] Facundoolano. Google-Play-Scraper. Zitiert nach github.com Abruf am 07.08.2023, 15.00 Uhr `https://github.com/facundoolano/google-play-scraper`.

[20] Figma. The collaborative interface design tool. Zitiert nach figma.com Abruf am 15.08.2023, 12.00 Uhr `https://www.figma.com`.

[21] G. Fischer. Symmetry of ignorance, social creativity, and meta-design. *Knowl. Based Syst.*, 13:527–537, 2000.

[22] L. V. Galvis Carreño and K. Winbladh. Analysis of user comments: An approach for software requirements evolution. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, page 582–591. IEEE Press, 2013.

[23] E. Gonzalez-Holland, D. Whitmer, L. Moralez, and M. Mouloua. Examination of the use of nielsen's 10 usability heuristics & outlooks for the future. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 61(1):1472–1475, 2017.

[24] E. Guzman, M. El-Haliby, and B. Bruegge. Ensemble methods for app review classification: An approach for software evolution (n). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 771–776, 2015.

[25] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 153–162, 2014.

[26] M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: Msr for app stores. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, MSR '12, page 108–111. IEEE Press, 2012.

[27] P. R. Henao, J. Fischbach, D. Spies, J. Frattini, and A. Vogelsang. Transfer learning for mining feature requests and bug reports from tweets and app store reviews, 2021.

[28] L. Hoon, R. Vasa, J.-G. Schneider, and J. C. Grundy. An analysis of the mobile app review landscape: Trends and implications. Technical report, Swinburne University of Technology, 2013.

[29] Hugging Face Inc. bart-large-mnli. Zitiert nach huggingface.co Abruf am 25.07.2023, 16.00 Uhr `https://huggingface.co/facebook/bart-large-mnli`.

[30] Hugging Face Inc. Distilroberta base model. Zitiert nach huggingface.co Abruf am 15.07.2023, 12.00 Uhr `https://huggingface.co/distilroberta-base`.

[31] Hugging Face Inc. Emotion english distilroberta-base. Zitiert nach huggingface.co Abruf am 15.07.2023, 12.00 Uhr `https://huggingface.co/j-hartmann/emotion-english-distilroberta-base`.

[32] Hugging Face Inc. Gibberish detection. Zitiert nach huggingface.co Abruf am 03.08.2023, 15.00 Uhr `https://huggingface.co/madhurjindal/autonlp-Gibberish-Detector-492513457`.

[33] Hugging Face Inc. Roberta base model. Zitiert nach huggingface.co Abruf am 15.07.2023, 12.00 Uhr `https://huggingface.co/roberta-base`.

[34] Hugging Face Inc. Transformers: State-of-the-art machine learning for pytorch, tensorflow, and jax. Zitiert nach huggingface.co Abruf am 26.07.2023, 10.00 Uhr `https://huggingface.co/docs/transformers/index`.

[35] Hugging Face Inc. Twitter-roberta-base for sentiment analysis - updated (2022). Zitiert nach huggingface.co Abruf am 25.07.2023, 16.00 Uhr `https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest`.

[36] IEEE. Systems and software engineering – life cycle processes – requirements engineering. *ISO/IEC/IEEE 29148:2011(E)*, 2011.

[37] N. Imtiaz, J. Middleton, P. Girouard, and E. Murphy-Hill. Sentiment and politeness analysis tools on developer discussions are unreliable, but so are people. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, SEmotion '18, page 55–61, New York, NY, USA, 2018. Association for Computing Machinery.

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

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

[40] J. Jeong and N. Kim. Does sentiment help requirement engineering: exploring sentiments in user comments to discover informative comments. *Automated Software Engineering*, 28, 11 2021.

[41] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall series in artificial intelligence. Pearson Prentice Hall, 2009.

[42] J. Kanwal, K. Smith, J. Culbertson, and S. Kirby. Zipf's law of abbreviation and the principle of least effort: Language users optimise a miniature lexicon for efficient communication. *Cognition*, 165:45–52, 2017.

[43] Z. Kuang, S. Zong, J. Zhang, J. Chen, and H. Liu. Music-to-text synaesthesia: Generating descriptive text from music recordings, 2023.

[44] M. Kuhnke. Identifzierung von Erklärungsbedarf via User-Feedback-Analyse. Master's thesis, Leibniz Universität Hannover, 2020.

[45] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, and M. Lanza. Pattern-Based Mining of Opinions in Q&A Websites. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 548–559, 2019.

[46] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto. Sentiment analysis for software engineering: How far can we go? In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 94–104, 2018.

[47] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[48] D. Loureiro, F. Barbieri, L. Neves, L. E. Anke, and J. Camacho-Collados. Timelms: Diachronic language models from twitter, 2022.

[49] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik. On the automatic classification of app reviews. *Requir. Eng.*, 21(3):311–331, sep 2016.

[50] W. Maalej and D. Pagano. On the socialness of software. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 864–871, 2011.

[51] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[52] Microsoft. Typescript is javascript with syntax for types. Zitiert nach typescriptlang.org Abruf am 07.08.2023, 15.00 Uhr `https://www.typescriptlang.org`.

[53] S. Mohammad. A practical guide to sentiment annotation: Challenges and solutions. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 174–179, 2016.

[54] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, Cambridge, MA, 2012.

[55] J. Nielsen. 10 usability heuristics for user interface design. Zitiert nach nngroup.com Abruf am 05.05.2023, 12.00 Uhr `nngroup.com/articles/ten-usability-heuristics`.

[56] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, page 249–256, New York, NY, USA, 1990. Association for Computing Machinery.

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

[58] N. Novielli, F. Calefato, and F. Lanubile. A gold standard for emotion annotation in stack overflow. In *Proceedings of the 15th International Conference on Mining Software Repositories*, MSR '18, page 14–17, New York, NY, USA, 2018. Association for Computing Machinery.

[59] OpenJS Foundation. Express application. Zitiert nach expressjs.com Abruf am 07.08.2023, 15.00 Uhr `https://expressjs.com/en/api.html`.

[60] OpenJS Foundation. Node.js is an open-source, cross-platform javascript runtime environment. Zitiert nach nodejs.org Abruf am 07.08.2023, 15.00 Uhr `https://nodejs.org/en`.

[61] Oracle. MySQL - Die populärste Open-Source-Datenbank der Welt. Zitiert nach mysql.com Abruf am 07.08.2023, 15.00 Uhr `https://www.mysql.com`.

[62] M. Ortu, G. Destefanis, B. Adams, A. Murgia, M. Marchesi, and R. Tonelli. The jira repository dataset: Understanding social aspects of software development. In *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering*, PROMISE '15, New York, NY, USA, 2015. Association for Computing Machinery.

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

[64] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *2013 21st IEEE International Requirements Engineering Conference (RE)*, pages 125–134, 2013.

[65] Pallets. Flask documentation. Zitiert nach flask.palletsprojects.com Abruf am 07.08.2023, 15.00 Uhr `https://flask.palletsprojects.com/en/2.3.x/`.

[66] F. Palomba, M. L. Vásquez, G. Bavota, R. Oliveto, M. D. Penta, D. Poshyvanyk, and A. D. Lucia. User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps. In *Proceedings of the 31st International Conference on Software Maintenance and Evolution*, pages 291–300. IEEE, 2015.

[67] S. Panichella, A. Di 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)*, pages 281–290, 2015.

[68] O. Pieczul, S. Foley, and M. E. Zurko. Developer-centered security and the symmetry of ignorance. In *Proceedings of the 2017 New Security Paradigms Workshop*, NSPW '17, page 46–56, New York, NY, USA, 2017. Association for Computing Machinery.

[69] Postman. API Platform. Zitiert nach postman.com Abruf am 25.08.2023, 21.00 Uhr `https://www.postman.com`.

[70] C. Potts, Z. Wu, A. Geiger, and D. Kiela. Dynasent: A dynamic benchmark for sentiment analysis, 2020.

[71] D. Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness and correlation. *Mach. Learn. Technol.*, 2, 01 2008.

[72] Python Software Foundation. Python is a programming language that lets you work quickly and integrate systems more effectively. Zitiert nach python.org Abruf am 07.08.2023, 15.00 Uhr `https://www.python.org`.

[73] A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[74] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2nd edition, 1979.

[75] D. Rozado, R. Hughes, and J. Halberstadt. Longitudinal analysis of sentiment and emotion in news media headlines using automated labelling with transformer language models. *PloS one*, 17:e0276367, 10 2022.

[76] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 10 2019.

[77] Scikit Learn. Groupkfold. Zitiert nach scikit-learn.org Abruf am 31.07.2023, 22.00 Uhr `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupKFold.html`.

[78] Scikit Learn. Stratifiedgroupkfold. Zitiert nach scikit-learn.org Abruf am 31.07.2023, 22.00 Uhr `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedGroupKFold.html`.

[79] Scikit Learn. Stratifiedkfold. Zitiert nach scikit-learn.org Abruf am 31.07.2023, 22.00 Uhr `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html`.

[80] N. Seyff, F. Graf, and N. Maiden. Using mobile re tools to give end-users their own voice. In *2010 18th IEEE International Requirements Engineering Conference*, pages 37–46, 2010.

[81] E. Stickel, H.-D. Groffmann, and K.-H. Rau. *I*, pages 321–356. Gabler Verlag, Wiesbaden, 1997.

[82] TechCrunch. Global google play and apple app store app revenues 2022. Zitiert nach de.statista.com Abruf am 02.05.2023, 18.40 Uhr `https://www.statista.com/statistics/183469/app-stores-global-revenues/`.

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

[84] G. Uddin, Y.-G. Guéhénuc, F. Khomh, and C. K. Roy. An empirical study of the effectiveness of an ensemble of stand-alone sentiment detection tools for software engineering datasets. *ACM Trans. Softw. Eng. Methodol.*, 31(3), apr 2022.

[85] Vuetify LLC. Vuetify. Zitiert nach vuetifyjs.com Abruf am 07.08.2023, 15.00 Uhr `https://vuetifyjs.com/en/`.

[86] H. Wachsmuth. *Text Analysis Pipelines*, pages 19–53. Springer, 12 2015.

[87] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, and B. Regnell. *Experimentation in Software Engineering*. Springer, 2012.

[88] E. Yourdon and L. Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Yourdon Press computing series. Prentice Hall, 1979.

[89] P. Zatko, I. Poupyrev, R. El Guerrab, and R. Dugan. Google Deepmind: Ground-breaking AlphaGo masters the game of Go. `https://www.youtube.com/watch?v=SUbqykXVx0A`, 05 2023.

[90] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.