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

App Review Management and Identification of Explanation Need in Privacy Issues from App Reviews

Bachelorarbeit

im Studiengang Informatik

von

Sebastian Raabe

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Dr. Jil Klünder
Betreuer: M. Sc. Wasja Brunotte

Hannover, 29.08.2022

Erklärung der Selbstständigkeit

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

Hannover, den 29.08.2022

Sebastian Raabe

Zusammenfassung

Menschen verwenden täglich stundenlang Smartphones und hinterlassen dabei viele digitale Spuren, da die meisten Apps in irgendeiner Form vertrauliche Daten der Nutzer erheben. Benutzer sollten das Recht haben zu erfahren, welche Daten in welcher Form verarbeitet, gespeichert und verwendet werden. Da dies nicht immer leicht ersichtlich ist, wird Benutzer-Feedback in Form von App Reviews analysiert. In diesen Bewertungen werden oft Privatsphäreprobleme hervorgehoben, die Nutzern und Entwicklern gezielt angezeigt werden könnten, um Transparenz in diesem Bereich zu fördern. Zu diesem Zweck wird in dieser Arbeit eine grafische Nutzeroberfläche entwickelt, in der App Reviews erstellt, verändert, in einer Datenbank gespeichert, gezielt übersichtlich angezeigt, gelabelt oder gelöscht werden können. Weiterhin werden Bewertungen hinsichtlich Privatsphäreproblemen und Erklärungsbedarf in diesen untersucht. Dazu wurde ein Word2Vec model auf App Reviews trainiert, um daraus auf Keywords basierende Heuristiken zu entwickeln. Außerdem wurden verschiedene NLP Schritte auf App Reviews durchgeführt, um darin einfacher Keywords zu erkennen. Der Ansatz zur Identifizierung von Privacy Issues erreicht eine gute Accuracy von 85 % und einen F1-Score von 67,5 %. Erklärungsbedarf in Privacy Issues konnte mit einer Accuracy von 94,9 % und einem F1-Score von 42,4 % identifiziert werden. Die entstandene Datenbank und GUI können in Zukunft genutzt werden, um App Reviews zu verwalten und mit verschiedenen Heuristiken zu analysieren.

Abstract

People use smartphones for hours every day, leaving behind many digital traces, as most apps collect users' confidential data in some form. Users should have the right to know what data is processed, stored and used, and in what form. Since this is not always readily apparent, user feedback is analyzed in the form of App Reviews. These reviews often highlight privacy issues that could be specifically displayed to users and developers to promote transparency in this area. To this end, this thesis develops a graphical user interface in which App Reviews can be created, modified, stored in a database, displayed in a well-structured targeted manner, labeled, or deleted. Furthermore, reviews are analyzed with respect to privacy issues and the explanation need in them. For this purpose, a Word2Vec model was trained on App Reviews to develop keyword-based heuristics. Also, various NLP steps were performed on App Reviews to detect keywords in them. The approach for identifying privacy issues achieved a good accuracy of 85 % and an F1 score of 67.5 %. Explanation need in Privacy Issues could be identified with an Accuracy of 94.9 % and a F1-Score of 42.4 %. The resulting database and GUI can be used in the future to manage App Reviews and analyze them with different heuristics.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Problemstellung . . . 1
- 1.2 Lösungsansatz . . . 1
- 1.3 Struktur der Arbeit . . . 2

2 Grundlagen **3**
- 2.1 Privatsphäre . . . 3
  - 2.1.1 Definition Privatsphäre . . . 3
  - 2.1.2 Definition Privacy Issue . . . 3
- 2.2 Definition Erklärungsbedarf . . . 4
- 2.3 Natural Language Processing . . . 4
  - 2.3.1 Tokenisierung . . . 5
  - 2.3.2 Lemmatisierung . . . 5
  - 2.3.3 N-Gramme . . . 5
  - 2.3.4 Stop Words . . . 5
  - 2.3.5 Word2Vec . . . 6
  - 2.3.6 Continuous Bag of Words . . . 6
- 2.4 Performance . . . 6

3 Konzept **9**
- 3.1 Ansatz . . . 9
- 3.2 Datensatz und Datenbank . . . 10
  - 3.2.1 Allgemein . . . 10
  - 3.2.2 Datenbankschema . . . 10
- 3.3 Grafische Nutzeroberfläche (GUI) . . . 12
  - 3.3.1 Anforderungsanalyse . . . 12
  - 3.3.2 Mockups . . . 14
- 3.4 Ground Truth . . . 15
- 3.5 Heuristiken . . . 15
  - 3.5.1 Privacy Issues identifizieren . . . 15
  - 3.5.2 Erklärungsbedarf identifizieren . . . 15

4 Implementierung **17**
4.1 Architektur . . . 17
4.2 Datenbank . . . 17
4.2.1 JSON-Format . . . 18
4.2.2 Reviews importieren . . . 19
4.2.3 Herausforderungen . . . 19
4.3 Grafische Nutzeroberfläche (GUI) . . . 20
4.3.1 Interface . . . 20
4.3.2 Labelmode . . . 22
4.4 Heuristiken . . . 23
4.4.1 Datenvorverarbeitung . . . 23
4.4.2 Word2Vec . . . 24
4.4.3 Keywords . . . 24
4.4.4 Identifizierung . . . 25
4.4.5 Optimierungen . . . 25

5 Evaluation **29**
5.1 Ergebnisse . . . 29
5.2 Interpretation . . . 30
5.3 Limitationen . . . 31

6 Verwandte Arbeiten **33**

7 Zusammenfassung und Ausblick **35**
7.1 Zusammenfassung . . . 35
7.2 Ausblick . . . 36

A Anhang **39**
A.1 Mockups . . . 39
A.2 GUI-Screenshots . . . 42
A.3 Stop Words . . . 43

Kapitel 1

Einleitung

1.1 Problemstellung

Menschen benutzen täglich digitale Systeme. Vor allem Smartphones werden intensiv genutzt, sei es zur Arbeit oder in der Freizeit. Im Durchschnitt verwenden Männer 154,26 Minuten und Frauen sogar 166,87 Minuten am Tag ihr Handy [1]. Fast jede App erhebt persönliche Daten des Nutzers und viele Nutzer wissen nicht genau, welche Daten sie in welchem Umfang preisgeben. Das kann der Standort einer Person sein, Zugriff auf Kameras, Fotos, Kontakte, E-Mails oder es werden Tastatureingaben verfolgt, durch die beispielsweise Passwörter erfasst werden könnten. Dabei sollte verantwortungsvoll mit diesen kritischen Informationen eines Menschen umgegangen werden, da so nicht nur Sicherheitslücken vermieden werden können, sondern die persönliche Freiheit eines jeden Menschen respektiert und geschützt werden sollte. Allerdings können Herausgeber einer App das Vertrauen in ihre Software stärken, wenn sie transparent und leicht zugänglich darstellen, wie vertrauliche Daten erhoben und verwendet werden.

Benutzer-Feedback in Form von App Reviews von Apps aus dem Google Play Store und dem Apple App Store sollte analysiert werden, um zu schauen, ob es mit einer App mutmaßlich Privatsphäreprobleme gibt oder nicht. Des Weiteren kann auch überprüft werden, ob die Nutzer nach Erklärungen für Probleme mit der Privatsphäre suchen oder nicht. Eine Verwaltung und übersichtliche Darstellung von App Reviews stellt eine Grundlage für diese Analyse dar. Dies ist Aufgabe der vorliegenden Arbeit.

1.2 Lösungsansatz

Um diese Aufgabe zu lösen, werden eine Datenbank, eine grafische Oberfläche und Heuristiken zur Identifizierung von Privacy Issues (s. Abschnitt 2.1.2) und Erklärungsbedarf (s. Abschnitt 2.2) benötigt. In der Datenbank können App Reviews sowie weitere Daten wie Apps und Kategorien gespeichert

werden. Diese Datenbank wird mit einer grafischen Benutzeroberfläche verbunden, in der die Verwaltung der Daten umgesetzt wird. Dies beinhaltet unter anderem das Importieren, Verändern, Sortieren, Löschen oder Labeln von Daten und das Eingeben eigener SQL-Abfragen. Außerdem können die Heuristiken gestartet werden. Die Heuristiken basieren auf einer Erkennung von Keywords, die teilweise einem trainiertem Word2Vec model (s. Abschnitt 2.3.5) entstammen. So wurden einzelne Review-Sätze analysiert, nachdem diese mithilfe von Natural Language Processing 2.3 Schritten vorverarbeitet wurden. Durch diese Arbeit können Privatsphäreprobleme entdeckt und abgespeichert werden, um weitere Arbeiten mit den analysierten Reviews durchzuführen.

1.3 Struktur der Arbeit

Diese Arbeit ist in sieben Kapitel und einen Anhang unterteilt. In Kapitel 2 werden wichtige Grundlagen zu Privacy Issues, Erklärungsbedarf und Natural Language Processing erläutert. In Kapitel 3 wird das Konzept der gesamten Arbeit vorgestellt, inklusive Datenbank, grafischer Nutzeroberfläche und Heuristiken zur Identifizierung von Privacy Issues und Erklärungsbedarf. In Kapitel 4 geht es dann um die einzelnen Schritte der Umsetzung des Konzepts. Anschließend werden in Kapitel 5 die Ergebnisse meiner Ansätze präsentiert und evaluiert. Auch Limitationen und Herausforderungen werden diskutiert. Des Weiteren wird in Kapitel 6 ein Blick auf verwandte Arbeiten geworfen und es findet eine Abgrenzung der Arbeit statt. Zum Schluss wird eine Zusammenfassung gegeben und ein Ausblick auf weitere Vertiefungsmöglichkeiten meiner Arbeit gezeigt.

Kapitel 2

Grundlagen

In diesem Kapitel werden Grundlagen von Natural Language Processing, Privatsphäre und Erklärungsbedarf dargestellt sowie die Definitionen für wichtige Begriffe gegeben.

2.1 Privatsphäre

2.1.1 Definition Privatsphäre

Für die Definition der Privatsphäre nutze ich die Definition aus der Arbeit von Baruth [2]:

„Privacy is the faculty and right that a person has to define, preserve and control the boundaries that limit the extent to which the rest of society can interact with or intrude upon. At the same time, he or she retains full control over information generated by, and related to, him or her.“ [21]

Diese Definition fasst verschiedene Prinzipien zusammen. Privatsphäre ist das grundsätzliche Recht einer Person, persönliche Grenzen selbst festlegen und kontrollieren zu können. Des Weiteren sollten alle vertraulichen Informationen eines Menschen von diesem kontrolliert werden. Im Softwarekontext sollten Nutzer das Recht haben zu erfahren, welche Daten wie verarbeitet, gespeichert und verwendet werden [3]. Durch Offenlegung solcher Informationen kann Vertrauen gestärkt und so die Qualität der Software erhöht werden.

2.1.2 Definition Privacy Issue

Eine App Review wird als Privacy Issue markiert, wenn die oben genannte Definition der Privatsphäre verletzt wird.

In Tabelle 2.1 sind beispielhaft Sätze dargestellt, die entweder als Privacy Issue eingestuft werden oder nicht. Hierbei ist wichtig anzumerken, dass

eine negative Konnotation eines Satzes gegeben sein muss, damit dieser als Privacy Issue eingestuft werden kann [2]. Der zweite Satz aus der Tabelle enthält zwar das Wort „privacy“, aber ist als Lob an die App zu verstehen und nicht als Verletzung der Definition der Privatsphäre.

Satz	Privacy Issue
One of my favorite apps!	Nein
This app values my privacy.	Nein
Bad privacy policy.	Ja
They monitor everything you type!	Ja

Tabelle 2.1: Beispielsätze für Privacy Issues

2.2 Definition Erklärungsbedarf

Es gibt zwei verschiedene Arten von Erklärungsbedarf, impliziten und expliziten Erklärungsbedarf [14]. Expliziter Erklärungsbedarf äußert sich durch eine klare Frage eines Nutzers, auf die dieser eine Antwort braucht. Impliziter Erklärungsbedarf ist keine direkte Frage, sondern eine Äußerung, bei der deutlich wird, dass der Nutzer ein Verständnisproblem hat.

In dieser Arbeit wird nur Erklärungsbedarf in Privacy Issues betrachtet. Wenn eine App Review Erklärungsbedarf enthält, aber kein Privacy Issue darstellt, wird diese Review nicht als Erklärungsbedarf eingestuft. Der dritte Satz in Tabelle 2.2 ist dafür ein Beispiel.

Satz	Erklärungsbedarf
One of my favorite apps!	Nein
They monitor everything you type!	Nein
Sometimes the app just changes the route and I don’t understand why.	Nein
I see no reason why simple reminder app should refuse to run without location permission.	Ja
Why does it require my phone number to validate service?	Ja
Why do you need permission for my storage?	Ja

Tabelle 2.2: Beispielsätze für Erklärungsbedarf

2.3 Natural Language Processing

Natural Language Processing (NLP) ist ein Teilgebiet der Informatik, in dem menschliche Sprache durch Computer verarbeitet wird [10]. NLP kann

beispielsweise eingesetzt werden für maschinelle Übersetzung, Dialogsysteme oder Spracherkennung.

2.3.1 Tokenisierung

Bei der Tokenisierung wird Text in kleinere Sinnesabschnitte unterteilt, sogenannte Tokens [13]. Token können aus Wörtern, Zahlen oder Satz- und Sonderzeichen bestehen. Es kann definiert werden, wie Text genau getrennt werden soll. Oft werden Sätze in Wörter unterteilt, indem man sie durch Leerzeichen trennt:

```
Ich lebe in Hannover => ['Ich', 'lebe', 'in', 'Hannover']
```

Durch diese Tokenisierung werden auch Wörter wie „New York“ getrennt, obwohl diese kontextuell zusammen gehören. Dieses Problem kann wiederum mit N-Grammen (s. Abschnitt 2.3.3) gelöst werden.

2.3.2 Lemmatisierung

Die Lemmatisierung wird verwendet, um Wörter auf ihre grammatikalische Grundform (Lemma) zurückzuführen [13]. Beispielsweise würde man aus „better“ das Lemma „good“ erhalten.

Das Stemming ist ein weiteres Verfahren, das Wörter Dabei werden Wörter auf den Worstamm zurückgeführt, zum Beispiel würden „change“ und „changing“ zu „chang“ verändert werden. Anders als beim Stemming sind die Ergebnisse bei der Lemmatisierung immer vollständige Wörter [13].

2.3.3 N-Gramme

N-Gramme sind ein Sequenzen von Wörtern der Länge N [13]. „New York City“ ist ein Beispiel für ein Trigramm und „Los Angeles“ ein Beispiel für ein Bigramm. So kann der Kontext des Satzes in vielen Fällen besser erhalten bleiben, da die Beziehung zwischen benachbarten Wörtern sehr wichtig ist.

2.3.4 Stop Words

Als Stop Words werden die Wörter einer Sprache bezeichnet, die am häufigsten darin vorkommen [13]. Diese Wörter kann man entfernen, ohne die Bedeutung des Satzes zu verändern. So würde zum Beispiel das Wort „aber“ aus einem Satz entfernt werden, da dieses keinen Mehrwert in der weiteren Verarbeitung erzeugt. Durch dieses Verfahren können viele Token entfernt werden, was eine große Zeitersparnis für die weitere Verarbeitung bedeutet.

2.3.5 Word2Vec

Word2Vec ist ein Verfahren, in dem Wörter eine Vektorrepräsentation erhalten. Dazu wird ein neuronales Netz auf einem Textdatensatz trainiert, das in der Lage ist, Beziehungen zwischen einzelnen Wörtern herzustellen und so die semantische und syntaktische Bedeutung in den Vektoren beizubehalten [13]. Die Dimension der Vektoren ist groß, oftmals 300 [7]. In diesen Vektorräume sind ähnliche Wörter nah zueinander. Durch die „Cosine Similarity“ kann die Distanz und damit die Ähnlichkeit von zwei Vektoren bestimmt werden. Durch die Darstellung der Wörter als Vektoren ist es möglich, arithmetische Operationen auf diesen durchzuführen. Beispielsweise ist so folgende Rechnung möglich:

```
King - Man + Woman = Queen
```

Word2Vec kann mit dem Continuous Bag of Words model oder dem Skip-Gram model implementiert werden. Nachfolgend wird nur das Continuous Bag of Words model erklärt.

2.3.6 Continuous Bag of Words

Das Continuous Bag of Words (CBOW) model besteht aus einem input layer, einem projection layer und einem output layer [17]. Dabei wird ein Wort vorhergesagt, basierend auf einem Kontextfenster, das beispielsweise aus zwei Wörtern links und rechts des gesuchten Wortes besteht [13]. Im Beispiel

This	app	harvests	all	your	personal	data

wird das Wort „harvests“ mithilfe der hellblauen Wörter vorhergesagt. Die grauen Wörter werden erst in nachfolgenden Schritten verwendet.

2.4 Performance

Bei einer binären Klassifikation wird einem Objekt eine von zwei vorher definierten Klassen zugewiesen [15]. Im Rahmen dieser Arbeit entscheiden Heuristiken, ob ein Review-Satz ein Privacy Issue (und zusätzlich Erklärungsbedarf) darstellt oder nicht. Um die Performance von Klassifikationen zu ermitteln, gibt es Metriken wie Recall, Precision, Accuracy und F1 Score [13].

Wenn zwei Klassen als positiv und negativ definiert werden und man das Ergebnis eines Modells mit den korrekten Werten aus dem Testdatensatz vergleicht, gibt es 4 mögliche Szenarios [13, 11]:

- True Positive: Das korrekte Ergebnis ist positiv und das Modell hat positiv ausgegeben

- True Negative: Das korrekte Ergebnis ist negativ und das Modell hat negativ ausgegeben
- False Positive: Das korrekte Ergebnis ist negativ und das Modell hat positiv ausgegeben
- False Negative: Das korrekte Ergebnis ist positiv und das Modell hat negativ ausgegeben

Aus diesen Szenarios ergibt sich eine Konfusionsmatrix, wie in Tabelle 2.3 zu sehen ist. Mit diesen Messungen können die oben genannten Metriken wie folgt berechnet werden [13, 11]:

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

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

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

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

Precision misst den Anteil der positiven korrekten Ergebnisse aus allen als positiv vorhergesagten Ergebnissen. Recall misst den Anteil der positiven korrekten Ergebnisse aus allen tatsächlich positiv Ergebnissen. Accuracy gibt den Anteil der korrekt vorhergesagten Ergebnisse aus allen Ergebnissen an. Der F1 Score bildet ein Gleichgewicht aus Recall und Precision [11].

		Predicted	
		P	N
Actual	P	True Positive (TP)	False Negative (FN)
	N	False Positive (FP)	True Negative (TN)

Tabelle 2.3: Konfusionsmatrix für binäre Klassifikation

Kapitel 3

Konzept

Dieses Kapitel beschäftigt sich mit den Ideen, dem Ablauf und dem Konzept der gesamten Arbeit, die in drei Arbeitspakete aufgeteilt worden ist. Dementsprechend wird zuerst der generelle Ansatz diskutiert und folgend systematisch das Vorgehen der einzelnen Abschnitte erklärt.

3.1 Ansatz

Diese Arbeit lässt sich grob in drei Arbeitspakete unterteilen. Zuerst sollte ein großer Datensatz bestehend aus App Reviews im JSON-Format (s. Abschnitt 3.2.1) in eine Datenbank geladen werden. Dafür wurde von mir ein Datenbankschema entwickelt sowie ein Database Management System (DBMS, s. Abschnitt 4.2) ausgesucht und vorbereitet. Danach konnten Reviews, Kategorien und Apps importiert werden. Anschließend sollte eine Software programmiert werden, die mit der Datenbank interagiert und alle Daten über eine grafische Oberfläche nutzbar macht. Auch kann man die Aktionen aus den anderen beiden Arbeitspaketen wie zum Beispiel der Import von Reviews oder das Finden von Privacy Issues hier durchführen. Das letzte Arbeitspaket bestand darin, Heuristiken zu entwickeln und zu implementieren, um Privacy Issues und Erklärungsbedarf in Privacy Issues aus App Reviews erkennen zu können. Einen genauen Überblick über die Arbeitspakete zeigt Abbildung 3.1.

Weiterhin baut diese Arbeit auf der Arbeit von Baruth [2] auf. Baruth hat für die Identifizierung von Privacy Issues in App Reviews Klassifizierungs- und Deep Learning Algorithmen eingesetzt. Darüber hinaus wurden Daten aus den App Stores von Apple und Google heruntergeladen, Methoden verglichen und eine Knowledge Base eingerichtet [2]. Im Gegensatz zu der Arbeit von Baruth wird hier eine Datenbank aufgebaut, eine GUI programmiert und zusätzlich zur Suche nach Privacy Issues auch nach Erklärungsbedarf gesucht. Der Datensatz, den ich zu Beginn meiner Arbeit erhalten habe, stammt auch aus der Arbeit von Baruth.

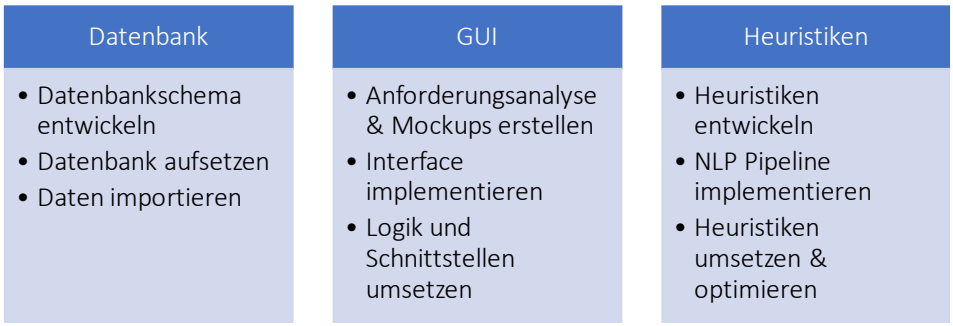

Abbildung 3.1: Aufteilung in drei Arbeitspakete

3.2 Datensatz und Datenbank

3.2.1 Allgemein

Zu Beginn meiner Arbeit erhielt ich einen Datensatz mit App Reviews aus dem Google Play Store und dem Apple App Store. Dieser bestand aus Kategorien wie z.B. „COMMUNICATION“, diese hatten wiederum Unterkategorien, beispielsweise „Recommended for you“ und in diesen befanden sich mit einer JSON-Datei pro App viele Dateien. In den JSON-Dateien ist immer ein großes Array mit beliebig vielen Reviews. Die Reviews bestanden aus einer AppId, einem score, einem datetime String, einem content string, einem array für sentences und ein privacy_issues flag. Die sentences hatten einzeln jeweils einen Satz und ein isPrivacyIssue flag. Dieses ist auf 1 gesetzt, falls ein Privacy Issue erkannt wurde und auf 0, wenn kein Privacy Issue erkannt wurde.

Ein Problem bestand aber darin, dass die Daten in einem nicht standardisierten JSON-Format abgespeichert waren. Deshalb mussten diese erst einmal normalisiert werden, bevor sie in eine Datenbank importiert werden konnten. Des Weiteren musste auch noch ein Datenbankschema entwickelt werden. Erst danach konnten alle Daten in eine Datenbank importiert werden.

3.2.2 Datenbankschema

Das Datenbankschema (s. Abbildung 3.2) ist ein Diagramm, das sich an UML-Syntax orientiert und aus den Reviews aus den JSON-Dateien herausgearbeitet wurde (R02, s. Anforderungstabelle 3.1). Es besteht aus fünf Tabellen: Review, ReviewSentence, App, Category und SubCategory. Grundsätzlich gibt es Apps, die immer genau eine Category und eine SubCategory besitzen. Diese beschreiben die Kategorien aus dem jeweiligen App Store, in der die App zu finden ist. Eine App besitzt außerdem eine

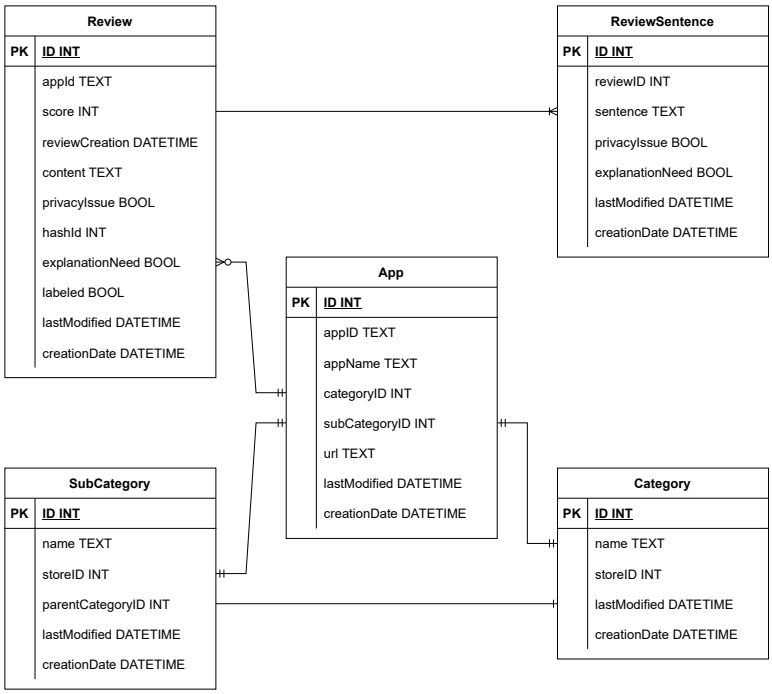

Abbildung 3.2: Datenbankschema

eindeutige ID, einen String mit dem Namen und eine URL, um diese App im Google Play Store oder Apple App Store aufzurufen. Die App ist mit der Review verbunden, da jede Review einen String mit dem Appnamen speichert. Eine Review hat außer einer ID noch den score, den Inhalt, eine berechnete HashId sowie Datum und Uhrzeit, zu dem die Review erstellt worden ist. Außerdem werden drei flags gespeichert, die anzeigen, ob die Bewertung schon gelabelt wurde, ob sie ein Privacy Issue darstellt und ob sie Erklärungsbedarf enthält. Als Nächstes ist jeder ReviewSentence über die ReviewID mit einer Review verbunden. Hier wird zusätzlich auch noch der Inhalt des jeweiligen Satzes, ob er ein Privacy Issue darstellt und ob er Erklärungsbedarf enthält, gespeichert. Abschließend wird von jedem Objekt aus jeder Tabelle jeweils das Datum und die Uhrzeit gespeichert, an dem es erstellt und zuletzt bearbeitet wurde.

In diesem Datenbankschema werden alle ID's als INT, alle flags als BOOL, alle Strings als TEXT und alle Datums- und Uhrzeit-Formate als DATETIME Objekte abgespeichert.

3.3 Grafische Nutzeroberfläche (GUI)

Der Erfolg einer Software hängt wesentlich davon ab, wie gut die Software auf die Anforderungen der Nutzer zugeschnitten ist [23]. Bevor man anfängt eine Software zu programmieren, sollte deshalb eine Anforderungsanalyse (s. Abschnitt 3.3.1) gemacht und Mockups (s. Abschnitt 3.3.2) erstellt werden, damit man die Anforderungen der Nutzer bestmöglich erfassen kann, um somit ein effektives Programm zu produzieren [23].

3.3.1 Anforderungsanalyse

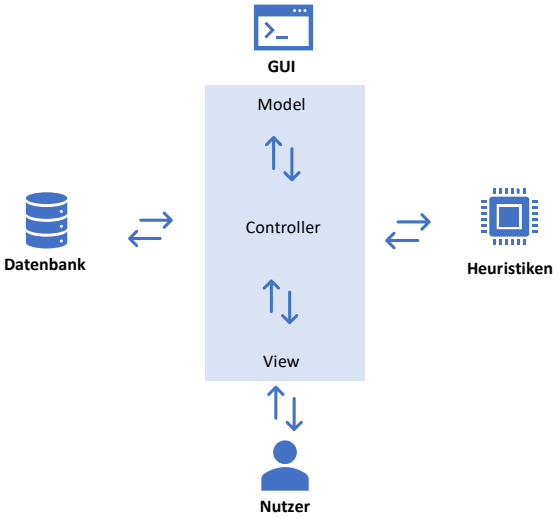

Abbildung 3.3: Interaktionen des MVC-Patterns mit dem Nutzer, der Datenbank und den Heuristiken

Es wurde zusammen mit dem Betreuer eine Liste mit Anforderungen erhoben, die im zweiten Arbeitspaket dieser Arbeit implementiert werden mussten.

Das Programm muss die gegebenen JSON-Dateien verarbeiten und alle Daten in die Datenbank importieren können. Sehr wichtig ist, dass man Daten in Tabellenform groß anzeigen lassen und diese beliebig sortieren kann. Des Weiteren soll man Zeilen auswählen, bearbeiten, löschen und hinzufügen können. Außerdem braucht man eine Pagination-Funktion, um durch die Daten durchblättern zu können. Wichtig ist auch die Möglichkeit, ein weiteres Fenster von der Anwendung öffnen zu können, ein Fenster nur für eigene SQL-Abfragen und ein Labelmode. Im Labelmode kann man dann einzelne Bewertungssätze durchgehen, erhält Informationen zu dem Satz angezeigt und kann anklicken, ob es sich um ein Privacy Issue oder

Erklärungsbedarf handelt oder nicht. Auch muss die Software die Funktion beinhalten, und Privacy Issues sowie Erklärungsbedarf selber erkennen zu können.

Zusätzlich ist gefordert, dass die Programmiersprache C# verwendet werden sollte und die Software nur auf dem Betriebssystem Windows laufen muss. Die Benutzeroberfläche wird komplett in Englisch gehalten. Abschließend sollte das Model-View-Controller (MVC) Pattern verwendet werden und das Programm muss die typischen Usability Richtlinien einhalten. MVC ist ein Design Pattern, dass ein Programm in drei wesentliche Bereiche unterteilt: Model, View und Controller. Das Model implementiert Datenobjekte, in dieser Arbeit beispielsweise Reviews oder Apps, beinhaltet aber keine Methoden für die Funktionalität der Software. Der View stellt die grafische Nutzerobfläche bereit und der Controller regelt die gesamte Programmlogik, oft initiiert durch Nutzeraktionen im View[5]. Ein Überblick über die Benutzung des MVC-Patterns in dieser Arbeit ist in Abbildung 3.3 zu sehen und eine tabellarische Auflistung der Anforderungen ist in Tabelle 3.1 dargestellt. Aspekte der Usability sind unter anderem, dass die Software angenehm zu bedienen, auf die Aufgabe zugeschnitten, übersichtlich und fehlertolerant ist.

ID	Anforderung
R01	JSON-Dateien standardisieren
R02	Es soll ein Datenbankschema modelliert werden
R03	Importieren der Daten in die Datenbank
R04	Anzeige von Daten in geeigneter, übersichtlicher Form
R05	Sortierung von Daten in der in der Anzeige aus R04
R06	Zeilen auswählen, bearbeiten, hinzufügen und löschen
R07	Aus Effizienzgründen soll immer nur ein Teil des Datensatzes aus der Datenbank geladen werden
R08	Es soll möglich sein, eine weiteres Fenster der Anwendung zu öffnen
R09	Es soll eine Möglichkeit geben, eigene SQL-Abfragen einzugeben und das Ergebnis übersichtlich darzustellen
R10	Möglichkeit zum effizienten Labeln von Reviewsätzen
R11	Programmiersprache C#
R12	Programm muss auf dem Betriebssytem Windows laufen
R13	MVC Pattern verwenden
R14	Usability Richtlinien einhalten
R15	Identifizierung von Privacy Issues
R16	Identifizierung von Erklärungsbedarf

Tabelle 3.1: Tabelle mit Anforderungen

3.3.2 Mockups

Mit einem Mockup kann man mit wenig Aufwand vorläufige Benutzeroberflächen entwerfen und verändern, bis sie allen Stakeholdern gefallen [22]. Bei meinem Mockup habe ich mich an gängige Konventionen gehalten, indem ich mich in der oberen linken Ecke für eine Navigationsleiste und unten links für eine Feedback-Textbox entschieden habe. Der Großteil des Interfaces wird jedoch von einem Bereich eingenommen, der Tabellen wie z.B. gelabelte Bewertungen anzeigen soll. Jeder der Buttons in der Navigationsleiste erzeugt ein Dropdown-Menü, von dem aus verschiedene Aktionen gestartet werden können. Wenn man auf File klickt, kann man beispielsweise Reviews importieren, ein SQL-Fenster für eigene Abfragen aufrufen oder den Labelmode starten. Im Show-Menü kann man sich viele unterschiedliche Daten anzeigen lassen. Unter anderem Apps, Reviews, Kategorien, nur gelabelte Reviews oder einzelne Review-Sätze mit Erklärungsbedarf. Im dritten Reiter kann man Privacy Issues sowie Erklärungsbedarf finden und die Performance der Heuristiken evaluieren. Bei Other kann man Zeilen hinzufügen, um selbst Daten einzugeben oder Zeilen komplett löschen.

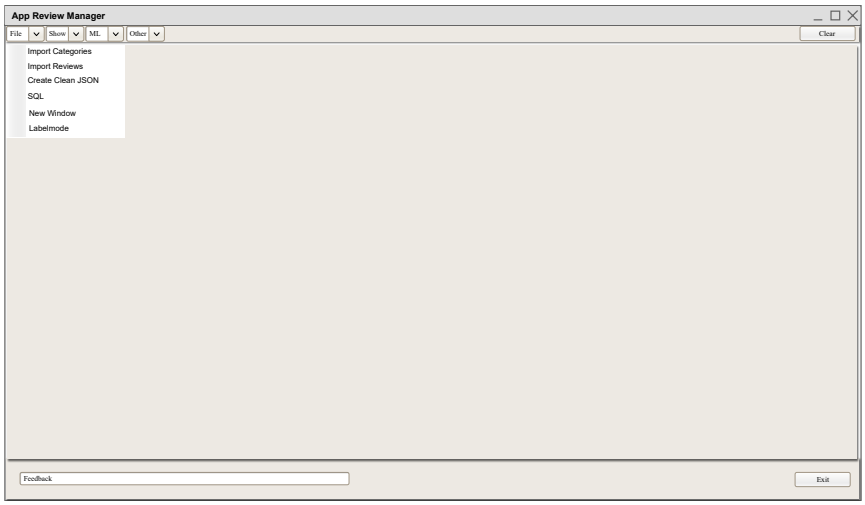

Abbildung 3.4: Mockup der Frontpage, mit Klick auf „File“

Weitere Mockup-Skizzen sind im Anhang zu finden (s. Abschnitt A.1).

3.4 Ground Truth

Eine Ground Truth stellt eine gelabelte Datengrundlage dar, die genutzt werden kann, um Modelle zu trainieren oder die Leistung zu evaluieren [16]. Dafür konnte ich auf gelabelte Daten aus der Arbeit von Baruth zurückgreifen. In dieser wurden bereits 1.743 Bewertungen hinsichtlich der Frage, ob sie ein Privacy Issue darstellen oder nicht, gelabelt [2]. Das sind 5.967 einzelne Bewertungssätze, wovon 1.291 als Privacy Issue gelabelt waren. Dies ergibt einen Anteil von ca. 21,6 %.

In der Phase der Entwicklung der Heuristiken habe ich die Bewertungen, die bereits als Privacy Issue gelabelt waren noch einmal gelabelt, diesmal wurde aber nach Erklärungsbedarf gesucht. Von den 1.291 Privacy Issue Sätzen wurden 150 auch als Erklärungsbedarf eingestuft, was einen Anteil von ca. 11,6 % bei den Privacy Issues ausmacht, aber nur ca. 2,5 % insgesamt.

3.5 Heuristiken

3.5.1 Privacy Issues identifizieren

In der Arbeit von Baruth wurden bei der Analyse von einzelnen Bewertungssätzen bessere Ergebnisse erzielt als bei der Analyse von ganzen Bewertungen auf einmal [2]. Deshalb wird in dieser Arbeit nur die Identifizierung von Privacy Issues und Erklärungsbedarf auf getrennten Review-Sätzen betrachtet. Darüber hinaus konnte Baruth keine eindeutige Empfehlung dafür aussprechen, eine automatische Rechtschreibkorrektur zu verwenden, um die Performance zu verbessern. Infolgedessen wird auf die Implementierung einer Rechtschreibkorrektur verzichet.

Es gibt viele verschiedene Möglichkeiten Privacy Issues in App Reviews zu finden. In der Arbeit von Baruth wurden verschiedene Varianten und Kombinationen aus Deep-Learning- und Klassifikationsalgorithmen verwendet. Ich beschränke mich in dieser Arbeit allerdings auf eine Erkennung über Keywords, wobei für die Keywords u.a. ein Word2Vec model eingesetzt werden soll. Die Keywords werden mit n-grams und Performancetests so gut wie möglich optimiert. Dieser Ansatz wurde gewählt, weil in dieser Arbeit der Fokus nicht nur auf der Identifizierung liegt und deshalb die Zeit für aufwändigere Heuristiken fehlt.

3.5.2 Erklärungsbedarf identifizieren

Die Identifizierung von Erklärungsbedarf in Privacy Issues funktioniert auch mittels einer Keyword Erkennung. Hier basieren die Keywords auf dem 5W1H-Ansatz (who, what, when, where, why, how), bei dem es darum geht, möglichst gezielt die wichtigsten Informationen bspw. eines Nachrichtenartikels zusammenzufassen. Diese Wörter werden jedoch auch

häufig verwendet von Menschen, die nach einer Erklärung für etwas suchen [8]. Hier kann man noch ein paar Keywords hinzufügen und optimieren und sollte dann hinreichend gut Erklärungsbedarf erkennen können.

Kapitel 4

Implementierung

In diesem Kapitel wird die Umsetzung des Konzepts thematisiert. Zuerst wird über die generelle Architektur der zu implementierenden Anwendung gesprochen. Anschließend gibt es für die Implementierung von Datenbank, GUI und Heuristiken jeweils ein separates Unterkapitel.

4.1 Architektur

Die Software wurde auf Basis des .NET Frameworks entwickelt und als Programmiersprache kam C# zum Einsatz (R11, R12). Bei der Entwicklung wurde konsequent das MVC-Pattern umgesetzt und sich an die Usability-Richtlinien gehalten (R13, R14). Hier können die komplette GUI sowie Schnittstellen programmiert werden. Die Anwendung interagiert mit einer Datenbank auf einem MariaDB Server, um dort Daten speichern, abrufen, verändern oder löschen zu können. Von der Anwendung aus werden Prozesse kreiert, die Python Scripts ausführen können. Diese werden genutzt, um dort Review Sätze zu verarbeiten und Privacy Issue sowie Erklärungsbedarf zu identifizieren (R15, R16). Python eignet sich dafür besonders gut, da es hier viele für diese Aufgabe geeignete NLP Bibliotheken gibt.

4.2 Datenbank

Ein DBMS ist Software, die genutzt wird, um die Verwaltung und Nutzung von großen Datensammlungen zu optimieren. Diese bieten viele Vorteile, unter anderem effizienterer Datenzugriff, Datenintegrität, Zugriffskontrolle mit Benutzerklassen, gleichzeitige Zugriffsplanung oder schnellere Entwicklung von neuen Applikationen [20].

Als DBMS wurde MariaDB[1] gewählt, da es bereits in der Arbeit von Baruth zur Implementierung einer Knowledge Base verwendet wurde.

[1] MariaDB: https://mariadb.org - letzter Zugriff am 05.08.22

Zusätzlich dazu wurde HeidiSQL verwendet. HeidiSQL[2] ist eine grafische Oberfläche für DBMS wie MariaDB, um einfacher mit den Datenbanken zu interagieren. Hier wurden die Tabellen passend zum Datenbankschema (s. Abbildung 3.2) erstellt.

4.2.1 JSON-Format

JSON oder JavaScript Object Notation ist ein für Menschen lesbares Format, um Daten strukturiert auszutauschen. Es benutzt Objekte in key-value-Paaren [18]. Dabei können Datentypen wie Zahlen, Strings oder Booleans verwendet und die Objekte in Arrays angeordnet werden.

Um die JSON-Dateien in ein standardisiertes Format zu bringen, musste jede Datei einzeln konvertiert werden (R01). Viele davon konnten direkt komplett eingelesen werden, aber manche Dateien mussten auch Stück für Stück verarbeitet werden, da die Größe der Dateien sehr stark variierte, von wenigen Kilobytes bis mehr als 1,5 Gigabyte. Das liegt daran, dass alle Bewertungen zu einer App immer in einer einzigen Datei abgespeichert werden und die Anzahl der Bewertungen sehr verschieden sind. Infolgedessen wurde auf den eingelesenen Daten eine Zeichenersetzung vollzogen:

```
{\"appId\": \"app.cartomizer\",
\"score\": 5,
\"datetime\": \"2020-10-03 09:59:01\",
\"content\": \"Awsome app! Very intuitive and usefull for me\",
\"sentences\": [{\"sentence\": \"Awsome app!\", \"isPrivacyIssue\": null},
{\"sentence\": \"Very intuitive and usefull for me\", \"isPrivacyIssue\": null}],
\"privacy_issues\": null}

Alle \" ersetzten durch " und danach alle \\ durch \ ersetzen

{"appId": "app.cartomizer",
"score": 5,
"datetime": "2020-10-03 09:59:01",
"content": "Awsome app! Very intuitive and usefull for me",
"sentences": [{"sentence": "Awsome app!", "isPrivacyIssue": null},
{"sentence": "Very intuitive and usefull for me", "isPrivacyIssue": null}],
"privacy_issues": null}
```

Zudem musste am Anfang und am Ende jeder Datei ein Zeichen abgeschnitten werden, da es sonst ungültige Anführungszeichen gäbe. Letztendlich wurden die neuen Dateien im JSON2-Dateiformat abgespeichert, um sie gut von den nicht standardisierten Dateien unterscheiden zu können.

Dieser erste Teilbereich wurde bereits in C# in der .NET Windows Forms App programmiert. So konnte man hier schon in einem simplen Dropdown-Menü auf „Create Clean JSON“ drücken und den Prozess starten.

[2]HeidiSQL: https://www.heidisql.com - letzter Zugriff am 05.08.22

4.2.2 Reviews importieren

Bevor man Daten aus dem C# Programm in die MariaDB Datenbank importieren kann, muss eine MySql Verbindung mithilfe eines MySql-Command[3] aufgebaut werden. Anschließend kann man zum Beispiel einen MySqlDataReader[4] nutzen, um SQL-Befehle auszuführen und somit Daten in die Datenbank zu laden (R03).

Als erstes mussten die Kategorien in die Datenbank geladen werden, damit die Eigenschaften Category und Subcategory der Apps automatisch auf die richtigen ID's gesetzt werden, wenn man Apps importiert. Alle möglichen Kategorien und Unterkategorien der Apps aus dem Google Play Store und dem Apple App Store konnten aus der jeweiligen Dateipfadstruktur des Datensatzes herausgelesen werden. Zum Beispiel befindet sich die App Skype im Pfad *google play store\COMMUNICATION\Recommended for you*, woraus sich die Kategorie *COMMUNICATION* und die Unterkategorie *Recommended for you* ergibt. Nun werden die JSON2-Dateien einzeln eingelesen. Bei dem Einlesen einer neuen Datei, muss sichergestellt werden, ob diese App bereits in der Datenbank existiert oder nicht. Wenn ja, dann wird die komplette Datei übersprungen, da man nun davon ausgehen kann, dass die Datei schon eingelesen wurde. Wenn nicht, dann wird die App in die Datenbank importiert und die vergebene ID kann für alle nachfolgenden Reviews verwendet werden.

Mithilfe der JsonConvert[5] Klasse kann man aus großen JSON-Dateien eine Liste an eigenen C# Objekten erstellen, in meinem Fall Reviews und Review-Sätze. Wenn man alle flags auf 0 setzt und für die Reviews eine HashId berechnet, können die Reviews und Review-Sätze in die Datenbank importiert werden.

4.2.3 Herausforderungen

In dem ersten Arbeitspaket kam es an zwei verschiedenen Stellen zu Problemen - einmal bei der JSON-Verarbeitung und einmal bei dem Import der App Reviews in die Datenbank.

Bei der Konvertierung von JSON zu Review-Objekten traten ein paar Probleme auf. Ich habe versucht, die JSON-Dateien einzulesen, zu verarbeiten, und anschließend direkt Reviews in die Datenbank zu importieren. Dabei gab es Fehler, die ich nicht direkt lösen konnte.

Die Lösung bestand darin, kleinschrittiger vorzugehen. Erst einmal habe

[3] MySqlCommand: https://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-sql-command.html - letzter Zugriff am 08.08.2022

[4] MySqlDataReader: https://dev.mysql.com/doc/dev/connector-net/6.10/html/T_MySql_Data_MySqlClient_MySqlDataReader.htm - letzter Zugriff am 17.08.22

[5] JsonConvert: https://www.newtonsoft.com/json/help/html/t_newtonsoft_json_jsonconvert.htm - letzter Zugriff am 08.08.2022

ich an kleineren Dateien getestet, ob die JSON-Verarbeitung ordnungsgemäß funktioniert. Nachdem ich an dieser Änderungen vorgenommen hatte, habe ich gemerkt, dass es mehr Sinn ergibt, komplett neue Dateien selbst zu erstellen, um direkt das standardisierte Format einzulesen und nicht immer erst durch den Verarbeitungsprozess zu gehen. Durch diese Schritte konnte ich auch andere Fehler klar zuordnen und lösen.

Ein anderes Problem war das Importieren der App Reviews. Dies war funktionell möglich, jedoch war die Geschwindigkeit dieses Prozesses zu langsam. Zuerst habe ich deshalb meinen Computer durchgehend Reviews hochladen lassen, aber da nur wenige Bewertungen pro Sekunde geschafft wurden, hat dies nicht ausgereicht. Deshalb habe ich nach einer Methode gesucht, mit der dieser Prozess beschleunigt werden kann. Unter anderem habe ich versucht, die Zahl der SQL-Abfragen zu verringern, indem ich deutlich mehr Review-Objekte in einer Abfrage importiere. Die Lösung des Problems war, die Kollisionsprüfung der HashId wegzulassen. Die HashId wurde für jede Bewertung berechnet um sicherzustellen, dass keine Bewertung doppelt in der Datenbank landet. Die Suche nach einer Kollision nahm leider den größten Teil eines Reviewimports ein, so dass dies anders gelöst werden musste. Das Problem, dass manche Bewertungen mehrfach in die Datenbank geladen werden, konnte auch simpler umgangen werden, in dem man sicherstellt, dass keine App oder keine Datei mehr als einmal eingelesen wird. Innerhalb der JSON-Dateien kommt keine Review doppelt vor, weshalb diese sukzessiv ohne Probleme eingelesen werden können. Nun können mehrere Hundert Bewertungen pro Sekunde importiert werden. Der einzige Nachteil ist, dass bei Problemen mit einer Datei, alle Reviews zu dieser App gelöscht werden müssen und bei der App wieder von vorne begonnen werden muss. Alternativ könnte man temporär wieder die Kollision der HashId's überprüfen und einen Geschwindigkeitsverlust hinnehmen.

4.3 Grafische Nutzeroberfläche (GUI)

Die Hauptbestandteile der GUI sind ein Hauptfenster, in dem Daten angeschaut sowie verwaltet und von dem aus die nachfolgenden Bestandteile ausgeführt werden können, ein Fenster zum Labeln von Bewertungssätzen (Labelmode), ein Fenster für eigene SQL-Abfragen (SQL-Fenster), die Interaktion mit der Datenbank und die Ausführung der entwickelten Heuristiken zur Identifizierung von Privacy Issues und Erklärungsbedarf.

4.3.1 Interface

In der Forms App exisitieren bereits vorgefertigte Textfelder, Buttons, Navigationsleisten, Dropdown-Menüs u. v. m. Man zieht die Elemente aus der Liste an die gewünschte Stelle im Interface. Properties, wie z.B. die

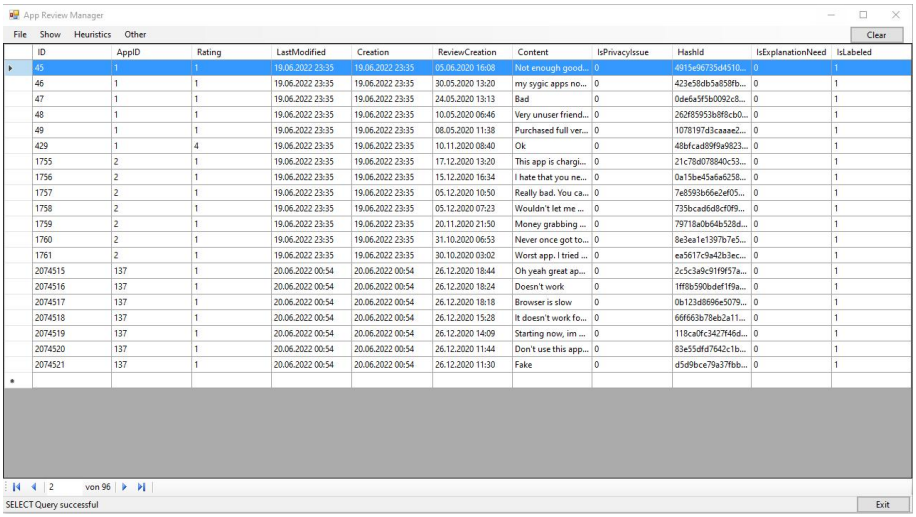

Abbildung 4.1: Hauptseite der GUI, die Apps anzeigt

Schriftgröße, können in einem eigenen Fenster oder programmierend eingestellt werden. Nur die Events eines Elements müssen zwingend programmiert werden.

Um Daten geeignet, übersichtlich darstellen zu können, eignet sich eine Tabellenform besonders gut (R04). Dafür wurde das Data Grid View Element ausgewählt, denn dieses ist speziell dafür geeignet, Daten in Tabellenform flexibel und übersichtlich darstellen zu können. Auch lassen sich darin Daten auswählen, bearbeiten, hinzufügen, löschen sowie sortieren (R05, R06).

Das Hauptfenster besteht aus einer Navigationsleiste, einem großen Data Grid View, einem Modul für Pagination und einer Statusleiste. Die Naviagationsleiste sieht anders aus als im Mockup, ein Menüpunkt heißt nun „Heuristics“ und nicht mehr „ML“, da letztendlich kein Machine Learning eingesetzt worden ist und „Heuristics“ einen besseren Namen darstellt. Wenn man bspw. auf „import categories“ klickt, wird aus dem View eine Funktion aus dem Controller aufgerufen, der diese Aufgabe übernimmt. Für das SQL-Fenster, den Labelmode und weitere Hauptfenster werden neue Forms gestartet und angezeigt (R08). Für die Unterpunkte aus dem „Show“-Menü werden verschiedene SQL-Abfragen an die Datenbank gemacht und das Ergebnis an das Data Grid View Element weitergeben. Für die Unterpunkte aus dem „Heuristics“-Menü werden Betriebssystem-Prozesse kreiert, denen man den Pfad der Python Installation, den Pfad des Python-Scripts sowie optional weitere Argumente übergibt. Nach der Ausführung werden an die Windows Forms App Ergebnisse oder Fehlermeldungen weitergegeben.

Pagination wird verwendet, um jeweils nur einen Teil des Datensatzes aus der Datenbank zu laden. Man kann so seitenweise weiterblättern oder eine Zielseite angeben (R07). Pagination ist deshalb angenehmer und praktischer

zu benutzen als nur eine Scrollbar am rechten Bildschirmrand, wie es noch im Mockup zu sehen ist.

In der Statusleiste befindet sich eine Textbox, die als Feedback für jegliche Aktionen in der GUI verwendet wird.

Das SQL-Fenster besteht aus einem großen Textfeld zur Eingabe von SQL-Abfragen, einer Statusleiste in der Feedback gegeben wird und zwei Buttons, „Execute“ und „Clear“ (R09). Mit dem ersten Button wird eine query an die Datenbank gemacht und mit dem zweiten Button wird das Textfeld geleert. Das Ergebnis einer Abfrage wird dann in dem Fenster angezeigt, von dem aus das SQL-Fenster gestartet wurde. In Abbildung 4.1 ist ein Bild der GUI zu sehen.

4.3.2 Labelmode

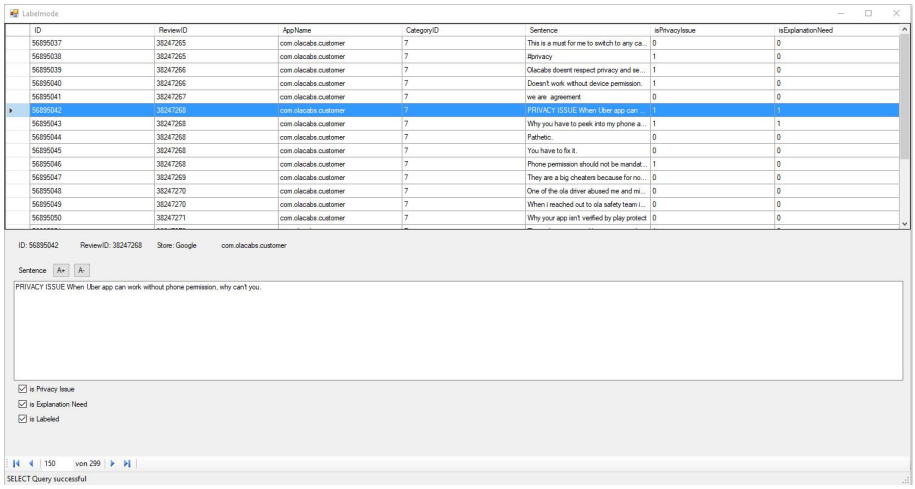

Abbildung 4.2: Labelmode der GUI

Im Labelmode kann man Bewertungssätze in der Tabelle anklicken und dann mithilfe von Checkboxen auswählen, ob es sich bei diesem Satz um ein Privacy Issue oder Erklärungsbedarf handelt oder nicht (R10). Dazu werden zusätzliche Informationen angezeigt, wie ID, ReviewID, App oder Store. Darüber hinaus lässt sich die Schriftgröße des Textfeldes einstellen. Sobald man auf eine Checkbox klickt, werden die zugehörigen Einträge in der Datenbank aktualisiert. Es ist nicht möglich, Erklärungsbedarf ohne Privacy Issues auszuwählen. Der Labelmode ist in Abbildung 4.2 dargestellt.

Weitere Bilder zur GUI sind im Anhang (s. Abschnitt A.2).

4.4 Heuristiken

In diesem Unterkapitel wird beschrieben, wie die Identifizierung von Privacy Issues und Erklärungsbedarf umgesetzt wurde. Hierfür sind eine NLP Preprocessing Pipeline, Word2Vec model Training, Keyword-Selektion, die Identifizierung an sich und Optimierungen nötig.

4.4.1 Datenvorverarbeitung

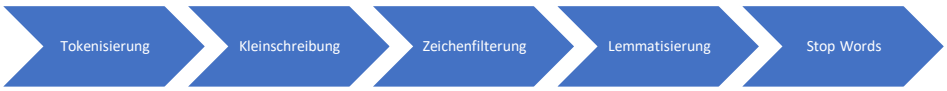

Abbildung 4.3: NLP Datenvorverarbeitung

Die Datenvorverarbeitung wurde in Python mithilfe der Natural Language Toolkit[6] (NLTK) Bibliothek implementiert. Dabei werden alle Schritte aus dem Natural Language Processing Grundlagen Bereich (s. Abschnitt 2.3) durchgeführt. Im ersten Schritt kann man einen Satz tokensieren mit der NLTK tokenize-Funktion. Die einzelnen Token können anschließend mit der lowercase-Funktion aus Python zu Kleinschreibung umgewandelt werden und mithilfe einer Regular Expression kann man alle Zeichen herausfiltern, die nicht für die Verarbeitung genutzt werden können. Da nur Buchstaben relevant sind, kann man mit der simplen Regular Expression `[^a-z]` alle Satz- und Sonderzeichen entfernen. So bestehen die Tokens jetzt nur noch aus kleingeschriebenen Buchstaben. Der WordNetLemmatizer von NLTK kann nun verwendet werden, um die einzelnen Token zu lemmatisieren. Anschließend wird jedes Token aus einem Satz gelöscht, das auf der Liste der Stop Words von NLTK zu finden ist. Diese komplette Liste enthält 179 Wörter der englischen Sprache und ist im Anhang in Tabelle A.1 zu finden.

Da diese Verarbeitungsschritte bei Millionen von Bewertungssätzen lange dauern, habe ich die fertigen Tokens in eine Datei schreiben lassen, da das Lesen von einer Datei deutlich schneller funktioniert und diese Tokens immer wieder gebraucht werden. Die Schritte sind in Abbildung 4.3 zu sehen.

[6]NLTK: https://www.nltk.org - letzter Zugriff am 05.08.2022

4.4.2 Word2Vec

Der Word2Vec Algorithmus wurde mit der Gensim[7] Bibliothek implementiert. Das model wurde mit dem CBOW Algorithmus trainiert. Als Eingabe dienten hier die Review-Sätze aus der Datenbank, die vorher von der NLP Preprocessing Pipeline vorverarbeitet wurden. Jedoch besteht die Eingabe einmal aus allen Review-Sätzen als unigrams und einmal als bigrams. Da sich durch diese große Trainingsdatenmenge die Trainingsdauer stark erhöhte, musste ich die Eingabe auf die Hälfte aller Bewertungssätze beschränken. Des Weiteren wurde eingestellt, dasx ein Wort mindestens zehnmal in der Eingabe vorhanden sein muss, um für die Ausgabe in Betracht gezogen zu werden.

Außerdem ist es möglich, dass trainierte model mit der save-Funktion abzuspeichern, um es später weiter zu trainieren oder um es direkt zu benutzen, ohne nochmal den Trainingsprozess durchlaufen zu müssen. Um das gespeicherte model abzurufen, muss wiederum die load-Funktion verwendet werden.

4.4.3 Keywords

Die Auswahl der Keywords basiert teilweise auf der Ausgabe des trainierten Word2Vec models und teilweise auf eigenen Überlegungen. Zuerst wurden viele Bewertungen gelesen, um ein Gefühl für diese zu bekommen. Insbesondere die bereits gelabelten Daten aus der Arbeit von Baruth [2] boten eine gute Möglichkeit, die Unterschiede zwischen Privacy Issues und normalen Bewertungen zu erkennen. Auf diese Art und Weise konnte eine vorläufige Liste mit vielen Wörtern wie unter anderem „privacy“, „permission“, „confidential“, „gdpr“, „ssn“ oder „trust“ erstellt werden.

Nun kann man das Word2Vec model einsetzen, um sich zu jedem dieser Wörter andere ähnliche Wörter ausgeben zu lassen. Eine beispielhafte Ausgabe ist in Abbildung 4.4 dargestellt. Diese kann man dann wiederum in die Keyword-Liste übernehmen, wenn sie möglicherweise dabei helfen, Privacy Issues zu erkennen. Den gleichen Vorgang kann man für bigrams wiederholen.

Nachdem diese Grundauswahl an Keywords getroffen wurde, wurde die Liste nach und nach basierend auf der Performance der Heuristik auf der Testdatenmenge optimiert (s. Abschnitt 4.4.5). Die Keywords werden alle nur kleingeschrieben, da die Token nach der Datenvorverarbeitung nur aus Kleinbuchstaben bestehen. Weiterhin befinden sich darin auch Keywords, die die falsch geschriebene Version anderer Keywords sind, um diese auch erkennen zu können.

[7] Gensim Word2Vec: https://radimrehurek.com/gensim/models/word2vec.html - letzter Zugriff am 05.08.2022

```
[('security', 0.8547893166542053),
 ('safety', 0.8494041562080383),
('protection', 0.7953622937202454),
 ('private', 0.7643076181411743),
   ('tos', 0.7605305314064026),
 ('policy', 0.7541942596435547),
('encryption', 0.7439422011375427),
 ('integrity', 0.7379205226898193),
('confidentiality', 0.7364035844802856),
   ('rule', 0.7356370687484741)]
```

Abbildung 4.4: Ausgabe des Word2Vec models zu dem Wort „privacy“

Die finale Auswahl der Keywords für Privacy Issues ist in Tabelle 4.1 zu sehen.

Für Erklärungsbedarf basieren die Keywords auf dem 5W1H-Ansatz. Darüber hinaus wird nach zusätzlichen Wörtern gesucht, die die Erkennung von Erklärungsbedarf verbessern könnten. Das weitere Vorgehen entspricht der Auswahl der Keywords für Privacy Issues.

Die finale Auswahl der Keywords für Erklärungsbedarf ist in Tabelle 4.2 zu sehen.

4.4.4 Identifizierung

Die Identifizierung von Privacy Issues funktioniert mithilfe einer Erkennung von Keywords als unigrams und bigrams. Man iteriert über einen verarbeiteten Bewertungssatz und schaut dabei, ob ein Wort in einer Liste aus unigrams auftaucht. Wenn ja, dann wird der Satz als Privacy Issue markiert. Wenn nicht wird der Satz in das bigram-Format umgewandelt und nach bigrams gesucht. Wenn hier eine Übereinstimmung mit einer Liste aus bigrams gefunden wird, kann man den Satz als Privacy Issue markieren. Wenn beide Ansätze keine Privacy Issues finden, so wird der Satz als ohne Privacy Issue markiert.

Die Identifizierung von Erklärungsbedarf funktioniert auch über die Erkennung von Keywords, allerdings nur mit unigrams. Ist ein Wort aus der Keyword-Liste im verarbeiteten Satz enthalten, so wird dieser als Erklärungsbedarf eingestuft, ansonsten nicht.

4.4.5 Optimierungen

Die Heuristik zur Identifizierung von Privacy Issues in App Reviews konnte noch weiter optimiert werden. Zuerst wurde die Liste an unigrams verbessert. Dies wurde getan, indem immer die Performance des Ansatzes mit und ohne ein bestimmtes keyword evaluiert worden ist. Anschließend konnte

ads	advertising	anonymity
breach	breeche	confidential
confidentiality	consent	disclosure
divulge	dsgvo	gdpr
hipaa	hippa	infringement
permission	pii	pricacy
privacy	privecy	privicy
privacysecurity	provacy	provecy
securityprivacy	safety	security
ssn	surveillance	tos
access_cell	access_info	access_personal
access_phone	collect_data	collect_information
collect_usage	complete_access	complete_invasion
data_leak	full_access	full_permission
give_access	identifiable_information	info_phone
information_sell	information_stolen	need_access
password_breach	password_breached	permission_phone
personal_data	personal_info	personal_information
phone_info	private_info	private_information
sell_info	sell_information	sensitive_information
steal_information	term_service	terms_service
total_access	trust_issue	user_agreement

Tabelle 4.1: Liste aller Privacy Issue Keywords

how	why	what	explain	explanation

Tabelle 4.2: Liste aller Erklärungsbedarf Keywords

auch die Liste an bigrams auf die selbe Art und Weise verbessert werden. Beispielsweise sind dies Statistiken aus dem Optimierungsprozess vor und nach dem Hinzufügen des bigrams „full_access“:

```
tp: 824, fn: 463, tn: 4094, fp: 484,
Precision: 0.63, Recall: 0.64, Accuracy: 0.839, F1: 0.635
------------------------------------------------------------------
tp: 879, fn: 408, tn: 4094, fp: 484,
Precision: 0.645, Recall: 0.683, Accuracy: 0.848, F1: 0.663
```

Des Weiteren ist es sehr hilfreich, sich genau anzuschauen, welche Bewertungssätze richtig erkannt werden und welche nicht. Am meisten Auskunft geben die false positive und false negative Review-Sätze. Darüber hinaus kann man auch verschiedene ungelabelte Daten verarbeiten lassen und überprüfen, ob die Heuristik zum gleichen Ergebnis kommt wie man selbst.

Bei der Optimierung der Keywords für Erklärungsbedarf wurde bemerkt, dass „who“, „where“ und „when“ nicht gut auf Erklärungsbedarf schließen lassen. Des Weiteren wurden einige Wörter wie zum Beispiel „reason“ oder „ask“ ausprobiert, aber alle hatten eine Verschlechterung der Performance zur Folge. Wie bei den Privacy Issues wurde auch hier auf die Ergebnisse auf der Testdatenmenge geschaut, aber es ließen sich keine weiteren Wörter ermitteln, die vermehrt auf Erklärungsbedarf hinwiesen.

Beim Labeln der Daten fiel allerdings auch auf, dass die meisten Sätze mit Erklärungsbedarf Fragen sind. Deswegen filtere ich Fragezeichen in der NLP Verarbeitung nun nicht mehr heraus und erhalte so eine neue Testdatenmenge. Nun wird ein Bewertungssatz auch als Erklärungsbedarf markiert, wenn es eine Frage ist. Dadurch kann der Recall deutlich von 69,3 % auf 96 % gesteigert werden, aber die Accuracy sinkt auch von 92,4 % auf 90,1 %.

Die besten Resultate werden hingegen erzielt, wenn man die Keywords komplett streicht und stattdessen die Einstufung von Erklärungsbedarf nur auf das Suchen nach einer Frage beschränkt. Wie in Kapitel 5 zu sehen ist, können so Precision, Accuracy als auch F1-Score auf Kosten des Recalls verbessert werden.

Kapitel 5

Evaluation

In diesem Kapitel werden die Ergebnisse der Arbeit vorgestellt, miteinander verglichen und interpretiert. Darüber hinaus wird auf Limitationen der Arbeit eingegangen und auf Herausforderungen, die während der verschiedenen Phasen bewältigt werden mussten.

5.1 Ergebnisse

Ansatz	Precision	Recall	Accuracy	F1-Score
Privacy Issues (nur unigrams)	0.613	0.586	0.828	0.599
Privacy Issues (unigrams + bigrams)	0.645	0.708	0.85	0.675
Erklärungsbedarf (keywords)	0.206	0.693	0.924	0.318
Erklärungsbedarf (keywords + Fragen)	0.199	0.96	0.901	0.329
Erklärungsbedarf (nur Fragen)	0.297	0.74	0.949	0.424

Tabelle 5.1: Performance der einzelnen Ansätze

Die Performance der Ansätze wurde evaluiert, indem ich die Ausgabe der Modelle mit den fast 6.000 manuell gelabelten Review-Sätzen aus der Ground Truth (s. Abschnitt 3.4) verglichen habe.

Aus Tabelle 5.1 geht hervor, dass mein Ansatz bei der Identifizierung von Privacy Issues ohne bigrams auf eine Accuracy von 82,8 % und einen F1-Score von 59,9 % kommt. Durch das Hinzufügen von bigrams konnten diese Werte verbessert werden, auf 85 % beziehungsweise 67,5 %.

Die Performance der Ansätze zum Finden von Erklärungsbedarf ist schlechter. Nur mit Keywords wird hier eine Accuracy von 92,4 % erreicht, allerdings bei einem F1-Score von nur 31,8 %. Werden Fragen bei der Suche mit berücksichtigt, dann sinkt die Accuracy auf 90,1 %, aber der F1-Score steigt auf 32,9 %. Am besten sind die Ergebnisse bei dem Ansatz, der nur auf Erkennung von Fragen setzt und die Keywords ignoriert. Hier wird

eine Accuracy von 94,9 % und ein deutlich verbesserter F1-Score von 42,4 % erreicht.

5.2 Interpretation

Die Ergebnisse zum Finden der Privacy Issues in App Reviews sind ausreichend gut. Mit dieser Heuristik kann man relativ gut abschätzen, ob es sich bei einer Review um ein Privacy Issue handelt oder nicht. So könnte man in der Praxis in App Stores einen Filter einbauen, der speziell Bewertungen hervorhebt, die sich mit dem Thema Privatsphäre auseinandersetzen. Allerdings lässt sich der Ansatz mit unigrams und bigrams noch verbessern. Zum einen gehe ich davon aus, dass man bessere Ergebnisse erzielen würde, wenn man zusätzlich oder ausschließlich höhere n-grams benutzt, z.B. trigrams. Dadurch könnten noch mehr typische Aussagen erkannt werden, bei denen man sich sicher sein kann, dass diese wirklich ein Privacy Issue darstellen. Aktuell werden Wörter wie „privacy“ erkannt, obwohl dieses nicht zwingend auf ein Issue hindeuten muss. Da es viele Bewertungen gibt, die die Privatsphäre einer App loben, ist es wichtig, dass entweder der positive oder negative Kontext eines Satzes erfasst wird. Der Satz

„good confidential messaging app“

würde in dem Ansatz fälschlicherweise als Privacy Issue markiert werden. Daher ist mein zweiter Vorschlag, einzelne Wörter auf ihre Konnotation zu überprüfen. Dazu könnte man das trainierte Word2Vec model einsetzen und prüfen, ob das aktuelle Wort eine positive oder negative Konnotation oder Ähnlichkeit zu anderen negativ oder positiv konnotierten Wörtern hat. Des Weiteren könnte man die Performance noch verbessern, indem man die Keywords mithilfe von mehr gelabelten Trainingsdaten weiter optimiert. Dennoch gehe ich davon aus, dass diese Heuristik schnell an ihre Grenzen stößt. Deep-Learning- oder Klassifizierungsalgorithmen bieten eine bessere Performance, beispielsweise ein Convolutional Neural Network, wie Baruth [2] bereits gezeigt hat.

Die Performance der Erklärungsbedarf-Heuristiken ist akzeptabel. Im Vergleich mit der Identifizierung von Privacy Issues ist hier ein Keyword-basierter Ansatz ineffektiver. Viele der Wörter, die in einem Satz mit Erklärungsbedarf auftauchen, kommen ebenso in Sätzen ohne Erklärungsbedarf vor. Selbst die Wörter aus dem 5W1H-Prinzip weisen nicht zwingend auf Erklärungsbedarf hin.

Bei vielen Sätzen ist es selbst für Menschen schwierig einzuschätzen, ob es sich bei dem Satz um Erklärungsbedarf handelt oder nicht. Die Benutzer verwenden oft Sarkasmus oder sind einfach erstaunt über die Privatsphäreprobleme der App, wodurch unklar wird, ob der Benutzer überhaupt eine Erklärung braucht oder nicht. Die Sätze

„Haven't they heard of privacy?“

und

„This app harvests all your personal data, how else do you think they pay their bills?“

sind dafür passende Beispiele, die von Menschen und Heuristiken fälschlicherweise als Erklärungsbedarf eingeschätzt werden könnten.

Auch hier könnte man die Keywords mithilfe von mehr gelabelten Trainingsdaten weiter optimieren, oder höhere n-grams verwenden, um die Performance zu verbessern. Dennoch bleibt der Keyword-basierte Ansatz für Erklärungsbedarf in Privacy Issues schwierig umzusetzen.

5.3 Limitationen

Eine größere Menge an gelabelten Daten wäre hilfreich, um bessere Heuristiken entwickeln zu können. Aktuell sind lediglich 150 Beispiele für Erklärungsbedarf in Privacy Issues gelabelt. Besonders praktisch wäre es, mehr positive Beispiele studieren zu können. Mit der hier entwickelten GUI inklusive Labelmode sollte das Labeln von neuen Review-Sätzen nun auch einfach möglich sein. Allerdings sollte man vorher mit der Hilfe einer Heuristik nach Privacy Issues oder Erklärungsbedarf suchen und die so gefundenen Reviews labeln, da die willkürliche Suche nach Privacy Issues und besonders Erklärungsbedarf sehr zeitintensiv sein kann.

Des Weiteren ist hier noch anzumerken, dass Privacy Issues zwar als solche markiert werden, allerdings nur auf Aussagen der Benutzer beruhen und nicht auf Echtheit überprüft werden. Dies und der Fakt, dass nur englische Bewertungen berücksichtigt und andere Sprachen vernachlässigt werden, wird in der Arbeit von Baruth [2] diskutiert. Im Rahmen dieser Arbeit sollten nur englische Bewertungen analysiert werden, aber es ist möglich, die Heuristiken so auszubauen, dass sie Privacy Issues und Erklärungsbedarf auch in Bewertungen anderer Sprachen erkennen können. Die Verifizierung der Echtheit der Privacy Issues ist nicht Teil dieser Bachelorarbeit gewesen.

Außerdem gab es beim Training des Word2Vec models hohe Wartezeiten. Dies machte es schwierig, die Parameter für den Algorithmus immer weiter zu optimieren. Dennoch hat das Word2Vec model für Privacy Issues viele ähnliche Wörter ausgegeben, die zur Identifizierung verwendet werden konnten. Bei Erklärungsbedarf konnte es jedoch nicht erfolgreich angewandt werden, da keine Wörter gefunden wurden, die auf diesem Testdatensatz eine spürbare Performancebesserung herbeiführten.

Kapitel 6

Verwandte Arbeiten

In Kapitel 6 werden weitere wissenschaftliche Arbeiten vorgestellt, die sich mit App Store Reviews, Privatsphäre oder Erklärungsbedarf beschäftigen. Anschließend findet eine Abgrenzung meiner Arbeit statt.

Casillo et al. [6] stellen eine Web Application names PReDUS vor, die Datenschutzanforderungen in User Stories erkennen kann. Dazu wird eine User Story eingegeben, auf der mithilfe von NLP eine syntaktische und eine semantische Analyse durchgeführt wird. Das Ergebnis wird an zwei Convolutional Neural Networks weitergegeben, die anschließend vorhersagen, ob und wie die User Story Datenschutzanforderungen enthält.

In [24] analysieren Vasa et al. 8,7 Millionen Bewertungen aus 17.330 Apps aus dem Apple App Store, um ein grundlegendes Verständnis für App Reviews aufzubauen. Unter anderem untersuchen die Autoren die Durschnittslänge einer Bewertung, ob der score die Länge der Bewertung beeinflusst sowie die Frage, ob die Kategorie der App die Länge der Bewertung beeinflusst.

Brunotte et al. [3] entwickelten ein Tool, das visuelle Erklärungen zu Datenschutzbestimmungen ausgibt, um Transparenz und Verständnis der Richtlinien zu erhöhen. Dafür werden drei Schritte benötigt, das Checken auf Vorhandensein einer Datenschutzbestimmung, das Analysieren der Datenschutzbestimmung sowie das Bereitstellen von visuellen Erklärungen.

Phong et al. [19] stellen eine Anwendung zur Auswertung von App Reviews vor. Diese benutzt einen auf Keywords basierenden Ansatz, um beispielsweise Keywords aus Reviews zu extrahieren oder Reviews, die gegebene Keywords beinhalten, zu finden. So ist es möglich, Trends in der Verwendung von Wörtern zu erkennen und zu visualisieren. Dadurch wären Entwickler in der Lage, schnell Probleme aus den Bewertungen herauszuarbeiten und zu lösen.

Hatamian et al. [9] haben ein Programm entwickelt, um für den Datenschutz relevante Aussagen aus Google Play Store App Reviews heraus-

zufiltern und zu kategorisieren. Zur Umsetzung wurden NLP, maschinelles Lernen und Stimmungsanalyse-Techniken eingesetzt. Unter anderem wird untersucht, ob es einen Zusammenhang zwischen App Reviews und der Realität der Datenverwendung einer App gibt.

In [4] untersuchen Brunotte et al. die Frage, ob und wie Erklärungen eingesetzt werden können, um das Bewusstsein für digitale Privatsphäre zu stärken. In einer Studie wurden Teilnehmende gefragt, ob Erklärungen das Vertrauen in ein System stärken könnten, wie groß ihr Interesse an Privatsphäre-Erklärungen ist, ob sie auf erforderliche App Berechtigungen achten sowie wie oft sie sich die Datenschutzbestimmungen von Websites durchlesen.

Kuhnke [14] und Jacobsen [12] identifizieren Erklärungsbedarf in App Reviews. Dafür setzt Kuhnke verschiedene Klassifizierer ein. Das beste Ergebnis konnte Kuhnke mit einem Convolutional Neural Network erreichen. Jacobsen verwendet einen Rule-based Matching Ansatz der Python Bibliothek spaCy[1]. Darüber hinaus hat Jacobsen ein Tool zur Analyse von Nutzer-Feedback entwickelt.

Baruth [2] identifiziert Privacy Issues in App Reviews aus dem Apple App Store und dem Google Play Store mithilfe unterschiedlicher Klassifizierungs- und Deep Learning Algorithmen. Des Weiteren wurden Daten aus den App Stores heruntergeladen, die Methoden zur Identifizierung gegenübergestellt und eine Knowledge Base eingerichtet.

Es existieren also schon einige wissenschaftliche Arbeiten, die sich mit Themen wie App Store Reviews, Privatsphäre und Erklärungsbedarf auseinandersetzten. In dieser Arbeit wird zusätzlich zur Identifizierung von Privacy Issues [2] und Erklärungsbedarf [12, 14] eine Datenbank aufgebaut, die App Reviews, einzelne Sätze, Kategorien, Unterkategorien und Apps beinhaltet. Weiterhin wurde eine GUI entwickelt, die nützlich für das App Review Management ist. In dieser kann man sich verschiedene Daten anzeigen lassen, sortieren, importieren, hinzufügen, löschen oder labeln. Die GUI kann auch mit anderen Datenbanken verbunden werden. Außerdem gibt es eine Schnittstelle zur Ausführung von Python Programmen, die genutzt werden kann, um Heuristiken zur Erkennung von Privacy Issues und Erklärungsbedarf auszuführen. Die entwickelten Heuristiken arbeiten in dieser Arbeit nur mit App Reviews und basieren auf Keywords. Es findet keine grundlegende Analyse von App Reviews, Privatsphäre oder Erklärungsbedarf an sich statt.

[1] spaCy: https://spacy.io/ - letzter Zugriff am 24.08.22

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Ziel dieser Arbeit Arbeit war es, App Reviews speichern und verwalten zu können und Privacy Issues sowie Erklärungsbedarf in diesen identifizieren zu können. Dazu wurde im ersten Arbeitspaket eine Datenbank eingerichtet, in der Reviews, Apps, Review-Sätze, Kategorien und Unterkategorien gespeichert werden. Anschließend wurden mehr als 38 Millionen Reviews aus mehr als 4.000 Apps in die Datenbank importiert.

Im zweiten Arbeitspaket wurde eine GUI entwickelt, mit der es unter anderem möglich ist, Daten geordnet anzuzeigen, hinzuzufügen, zu sortieren, zu verändern, zu löschen oder zu labeln. Von der GUI aus können weitere Aktionen gestartet werden, wie das Starten der Heuristiken aus dem dritten Arbeitspaket oder das Importieren von Daten aus dem ersten Arbeitspaket.

In dem letzten Arbeitspaket wurden Heuristiken zur Identifizierung von Privacy Issues und Erklärungsbedarf entwickelt und optimiert. Diese Heuristiken basieren auf der Erkennung von Keywords, die zu einem großen Teil einem trainierten Word2Vec model entstammen. Bevor die Heuristiken auf die App Reviews angewandt wurden, wurde eine Datenvorverarbeitung mithilfe von NLP-Schritten durchgeführt.

Für die Identifizierung von Privacy Issues konnte eine Accuracy von 85 % und ein F1-Score von 67,5 % erreicht werden. Für die Identifizierung von Erklärungsbedarf in Privacy Issues konnte eine Accuracy von 94,9 % und ein F1-Score von 42,4 % erreicht werden. Diese Ansätze können hinreichend gut verwendet werden, um Privacy Issues und Erklärungsbedarf zu erkennen. Mit diesen Heuristiken könnte man in der Praxis in App Stores Filter implementieren, die Bewertungen anzeigen, die sich wahrscheinlich mit Privatsphäre und Erklärungsbedarf auseinandersetzen, um auf diese Art und Weise Nutzern und Entwicklern Informationen zu diesen Themen anzuzeigen.

7.2 Ausblick

Durch meine Herausforderungen im ersten Arbeitspaket (s. Abschnitt 4.2.3) kam es zu Zeitverzögerungen, wodurch es schwierig wurde, den Zeitplan einzuhalten. Mit mehr Zeit wäre es möglich gewesen, im dritten Arbeitspaket bessere Heuristiken zur Identifizierung von Privacy Issues und Erklärungsbedarf entwickeln zu können. Baruth [2] und Kuhnke [14] konnten im Kontext ihrer Masterarbeiten in diesen Bereichen erfolgreichere Ansätze entwickeln, die zusammen mit der in dieser Arbeit entwickelten GUI verwendet werden könnten. Natürlich könnte man auch meine Heuristiken verbessern, aber ich denke, es ist sinnvoller, komplexere Deep Learning oder Klassifizierungen einzusetzen oder direkt die Ansätze von Baruth und Kuhnke zu verwenden und eventuell weiter auszubauen.

Aktuell können nur englische Bewertungen ausgewertet werden, was viele Reviews direkt ausschließt. Eine Erweiterung auf weitere Sprachen sollte deshalb in Betracht gezogen werden. Des Weiteren wäre es interessant, einen Zusammenhang zwischen markierten Privacy Issues und den tatsächlichen Datenschutzverletzungen einer App herstellen zu können. Einen möglichen Ansatz stellen Hatamian et al. [9] vor, in dem der Datenverkehr einer App überwacht wird. Momentan kann die Echtheit von Privacy Issues in Bewertungen nicht überprüft werden.

Es gibt viele Möglichkeiten, die entwickelte GUI weiter zu verbessern. Zum Beispiel könnte die Abfragemöglichkeit hinzugefügt werden, dass ausgewählt werden kann, zu welcher App man Reviews im Labelmode labeln möchte. Dies wurde bereits in der Arbeit von Jacobsen [12] umgesetzt. Ebenso könnte die Auswahl hinzugefügt werden, von welcher App Reviews analysiert werden sollen. So könnte man gezielter bestimmte Apps untersuchen und Ergebnisse miteinander vergleichen.

Eine weitere Idee, die sich mit dieser Arbeit verbinden ließe, ist aus der Arbeit von Phong et al. [19]. In dieser Arbeit werden unter anderem Keywords aus App Reviews extrahiert und anschließend Trends in der Verwendung bestimmter Wörter erkannt. Mit dieser Implementierung könnte man schnell feststellen, ob Nutzer sich beispielsweise nach einem App-Update über Probleme mit der Datenschutzbestimmung beschweren. So könnte man Nutzern und Entwicklern Probleme sowie Qualitätsaspekte einer App aufzeigen. Des Weiteren könnte man eine bessere Anzeige der Ergebnisse der Review Analyse in der GUI entwickeln, indem man die einzelnen Sätze mit den flags in Tabellenform darstellt. Zusätzlich dazu könnte der Teil eines Satzes farblich markiert oder anders hervorgehoben werden, der dafür verantwortlich ist, dass ein Satz als Privacy Issue oder Erklärungsbedarf markiert worden ist [12].

Außerdem könnte die GUI verwendet werden, um App Reviews auch auf andere Software-Qualitätsaspekte hin zu untersuchen [2]. Auch könnte die Menge an gelabelten Daten größer sein, vor allem für Erklärungsbedarf in

Privacy Issues. Mit der GUI können leicht mehr Reviews gelabelt werden, um die Entwicklung besserer Heuristiken zu fördern. Darüber hinaus könnten gefundene Privacy Issues und gefundener Erklärungsbedarf kategorisiert werden [2]. Dazu müsste die Datenbank um zusätzliche Tabellen und die Identifizierung um eine automatische Erkennung der Kategorien erweitert werden.

Anhang A

Anhang

A.1 Mockups

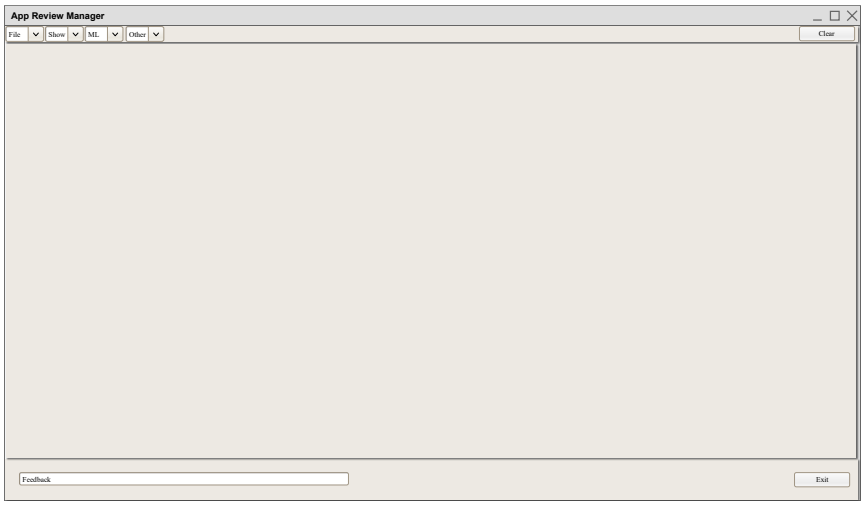

Abbildung A.1: Mockup der Frontpage

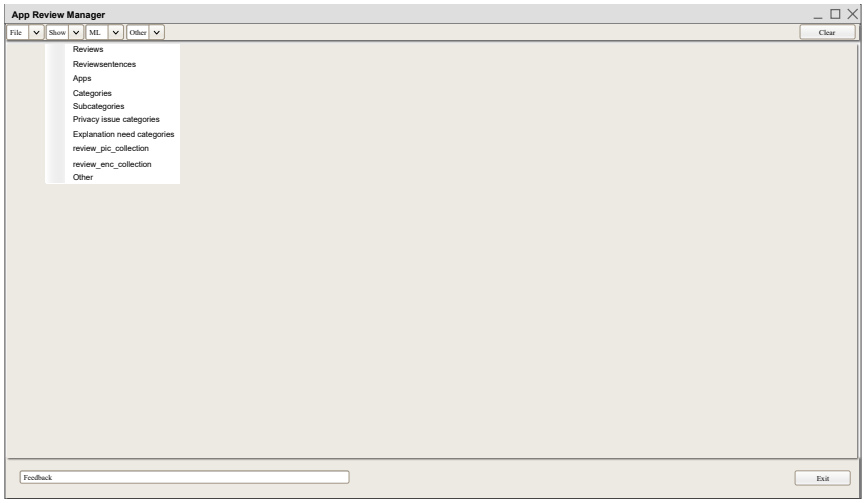

Abbildung A.2: Mockup der Frontpage, mit Klick auf „Show“

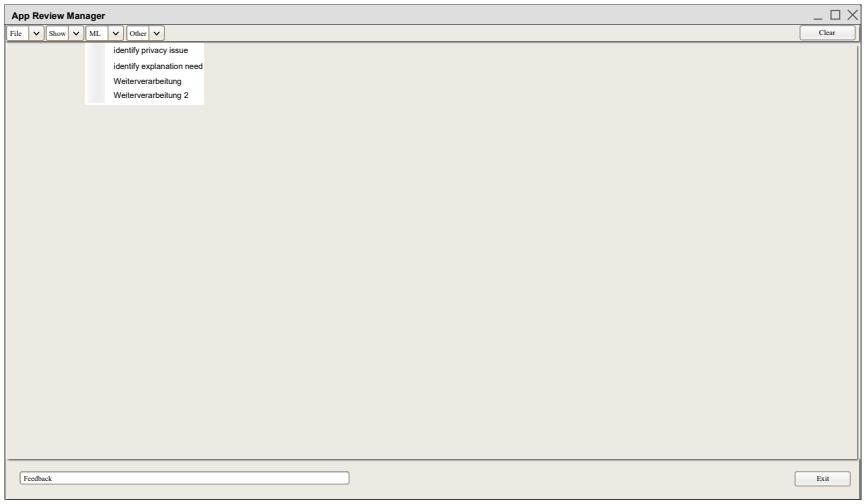

Abbildung A.3: Mockup der Frontpage, mit Klick auf „ML“

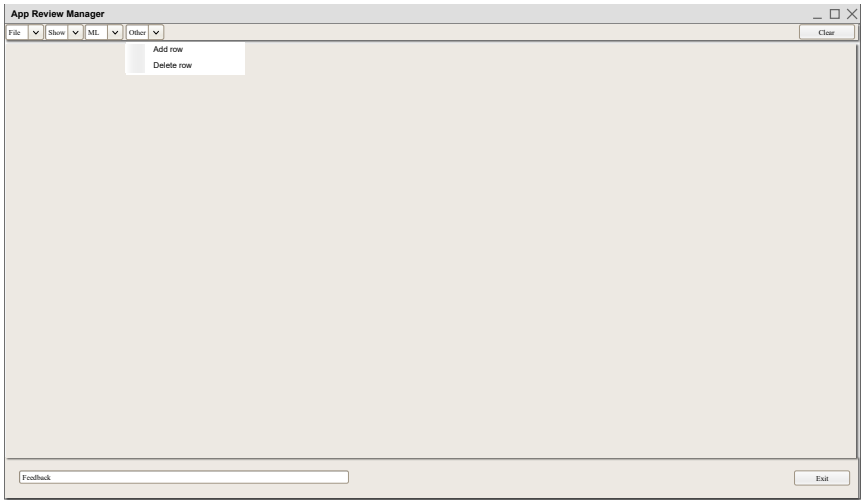

Abbildung A.4: Mockup der Frontpage, mit Klick auf „Other“

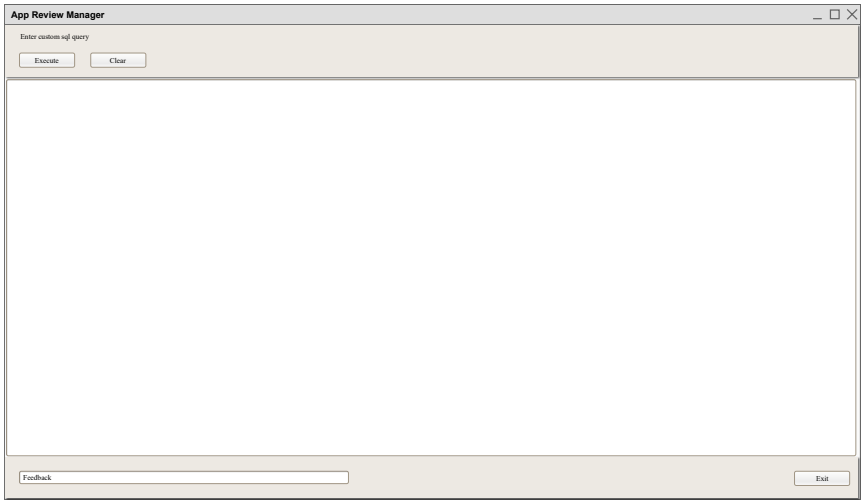

Abbildung A.5: Mockup des SQL-Fensters

A.2 GUI-Screenshots

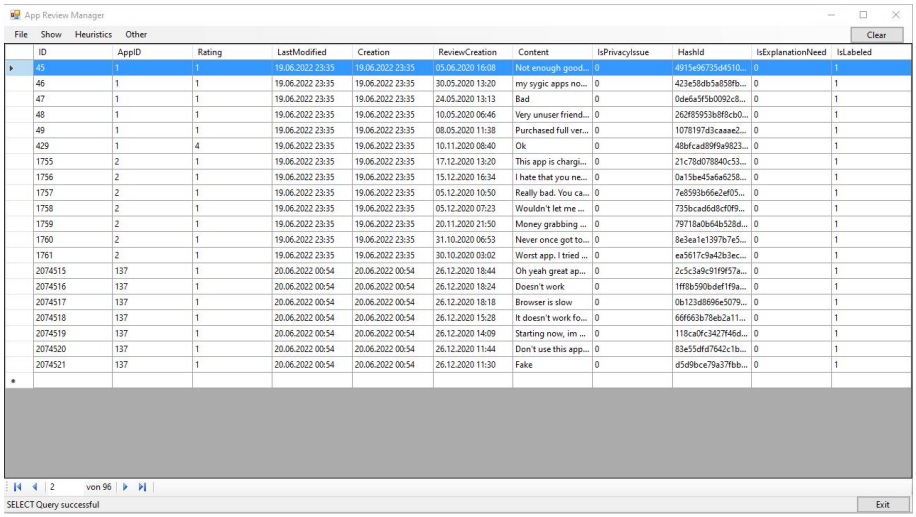

Abbildung A.6: Hauptseite der GUI, die gelabelte Reviews anzeigt

Abbildung A.7: SQL-Fenster der GUI

A.3 Stop Words

i	me	my	myself	we	our
ours	ourselves	you	you're	you've	you'll
you'd	your	yours	yourself	yourselves	he
him	his	himself	she	she's	her
hers	herself	it	it's	its	itself
they	them	their	theirs	themselves	what
which	who	whom	this	that	that'll
these	those	am	is	are	was
were	be	been	being	have	has
had	having	do	does	did	doing
a	an	the	and	but	if
or	because	as	until	while	of
at	by	for	with	about	against
between	into	through	during	before	after
above	below	to	from	up	down
in	out	on	off	over	under
again	further	then	once	here	there
when	where	why	how	all	any
both	each	few	more	most	other
some	such	no	nor	not	only
own	same	so	than	too	very
s	t	can	will	just	don
don't	should	should've	now d	ll	m
o	re	ve	y	ain	aren
aren't	couldn	couldn't	didn	didn't	doesn
doesn't	hadn	hadn't	hasn	hasn't	haven
haven't	isn	isn't	ma	mightn	mightn't
mustn	mustn't	needn	needn't	shan	shan't
shouldn	shouldn't	wasn	wasn't	weren	weren't
won	won't	wouldn	wouldn't		

Tabelle A.1: Liste der englischen Stop Words von NLTK

Literaturverzeichnis

[1] I. Andone, K. Błaszkiewicz, M. Eibes, B. Trendafilov, C. Montag, and A. Markowetz. How age and gender affect smartphone usage. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, page 9–12, New York, NY, USA, 2016. Association for Computing Machinery.

[2] R. Baruth. Identifizierung und Analyse von Privacy Issues in App Store Reviews. Master's thesis, Leibniz Universität Hannover, 2021.

[3] W. Brunotte, L. Chazette, L. Kohler, J. Klunder, and K. Schneider. What About My Privacy? Helping Users Understand Online Privacy Policies. In *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering*, ICSSP'22, page 56–65, New York, NY, USA, 2022. Association for Computing Machinery.

[4] W. Brunotte, L. Chazette, and K. Korte. Can Explanations Support Privacy Awareness? A Research Roadmap. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 176–180, 2021.

[5] J. Bucanek. *Model-View-Controller Pattern*, pages 353–402. Apress, Berkeley, CA, 2009.

[6] F. Casillo, V. Deufemia, and C. Gravino. Predus: A privacy requirements detector from user stories. In *REFSQ Workshops*, volume 3122 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

[7] K. W. Church. Word2vec. *Nat. Lang. Eng.*, 23(1):155–162, 2017.

[8] S. Han, K. Lee, D. Lee, and G. G. Lee. Counseling dialog system with 5w1h extraction. In *Proceedings of the SIGDIAL 2013 Conference, The 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 22-24 August 2013, SUPELEC, Metz, France*, pages 349–353. The Association for Computer Linguistics, 2013.

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

[10] J. Hirschberg and C. D. Manning. Advances in natural language processing. *Science*, 349(6245):261–266, 2015.

[11] M. Hossin and M. N. Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.

[12] B. L. Jacobsen. Entwicklung eines tools zur feedback-visualisierung. Bachelor's thesis, Leibniz Universität Hannover, 2021.

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

[14] M. Kuhnke. Identifizierung von erklärungsbedarf via user-feedback-analyse. Master's thesis, Leibniz Universität Hannover, 2021.

[15] R. Kumari and S. K. Srivastava. Machine learning: A review on binary classification. *International Journal of Computer Applications*, 160(7), 2017.

[16] S. Lebovitz, N. Levina, and H. Lifshitz-Assaf. Is AI ground truth really true? the dangers of training and evaluating AI tools based on experts' know-what. *MIS Q.*, 45(3), 2021.

[17] B. Liu. Text sentiment analysis based on CBOW model and deep learning in big data environment. *J. Ambient Intell. Humaniz. Comput.*, 11(2):451–458, 2020.

[18] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta. Comparison of JSON and XML data interchange formats: A case study. In D. Che, editor, *Proceedings of the ISCA 22nd International Conference on Computer Applications in Industry and Engineering, CAINE 2009, November 4-6, 2009, Hilton San Francisco Fisherman's Wharf, San Francisco, California, USA*, pages 157–162. ISCA, 2009.

[19] M. V. Phong, T. T. Nguyen, H. V. Pham, and T. T. Nguyen. Mining user opinions in mobile app reviews: A keyword-based approach (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 749–759, 2015.

[20] R. Ramakrishnan and J. Gehrke. *Database management systems (3. ed.)*. McGraw-Hill, 2003.

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

[22] J. M. Rivero, G. Rossi, J. Grigera, J. Burella, E. R. Luna, and S. Gordillo. From mockups to user interface models: an extensible model driven approach. In *International Conference on Web Engineering*, pages 13–24. Springer, 2010.

[23] L. Teixeira, V. Saavedra, C. Ferreira, J. Simões, and B. Sousa Santos. Requirements engineering using mockups and prototyping tools: Developing a healthcare web-application. In S. Yamamoto, editor, *Human Interface and the Management of Information. Information and Knowledge Design and Evaluation*, pages 652–663, Cham, 2014. Springer International Publishing.

[24] R. Vasa, L. Hoon, K. Mouzakis, and A. Noguchi. A preliminary analysis of mobile app user reviews. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, page 241–244, New York, NY, USA, 2012. Association for Computing Machinery.