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

Analyse und Visualisierung der Stimmung innerhalb von Open-Source-Projekten durch Entwicklung einer Firefox-Erweiterung

Masterarbeit

im Studiengang Technische Informatik

von

Elham Salajegheh Tezerji

Prüfer: Prof. Dr. rer. nat. Kurt Schneider
Zweitprüferin: Dr. rer. nat. Jil Ann-Christin Klünder
Betreuer: M.Sc. Martin Obaidi

Hannover, 16.05.2022

Kurzzusammenfassung

Die moderne Softwareentwicklung wird immer vielschichtiger und ist mit komplexen Anforderungen an den Informationsaustausch auf Softwareentwicklung-Plattformen konfrontiert. In diesem Zusammenhang wird zukünftig die Stimmungsanalyse von Open-Source-Projekten auf Softwareentwicklung-Plattformen eine elementare Rolle spielen. Da erwiesen ist, dass zufriedenere Entwickler:innen produktiver sind und sich damit auch die Erfolgswahrscheinlichkeit eines Softwareprojekts erhöht. Vor diesem Hintergrund kann eine automatisierte Analyse schriftlicher Daten bzw. Kommentare ein wichtiger Baustein sein, um die Stimmung in Open-Source-Projekten wie auf z.B. GitHub zu identifizieren. Dies erfordert eine zentrale Sammlung von Daten aus Open-Source-Projekten. Es existieren bereits Tools zur Stimmungsanalyse, die mit Daten von Plattformen wie GitHub trainiert und bewertet wurden. Ziel dieser Arbeit war es, ein Konzept zu entwickeln, das die automatisierte Analyse von Stimmung und deren Visualisierung durch eine Erweiterung für Firefox ermöglicht. Dem vorgeschaltet ist die automatische URL-Erkennung der Softwareentwicklung-Plattformen GitHub. Die Implementierung soll Entwickler:innen dabei helfen, mittels automatisiertem Crawlen potentielle Daten zur Stimmungsanalyse auf GitHub zu identifizieren, um auf dieser Grundlage bezüglich der Teilnahme an einem Open-Source-Projekte eine Entscheidung zu treffen. Die Konzeptentwicklung umfasst die Aspekte der Datensammlung, der Sentimentanalyse und der Datenvisualisierung. Zur praktischen Evaluierung des Konzepts wurde eine Befragung von GitHub nutzenden Probanden aus dem universitären Umfeld durchgeführt. Das Plugin als Firefox-Erweiterung zeigt im Wesentlichen, dass die Stimmungsanalyse als unterstützendes System betrachtet wird, um wiederum durch Polarität von geschriebenen Kommentaren zwischen Entwickler:innen positiv zu wirken. Perspektivisch ist es notwendig, einen Crawl der Daten auf GitHub durchzuführen, um einerseits die Datensammlung in der Praxis zu erfassen und andererseits mehr Daten zu sammeln, um die Genauigkeit der Stimmungsanalysen zu optimieren.

Abstract

Modern software development is becoming more and more complex and is confronted with complex requirements for the exchange of information on software development platforms. In this context, the sentiment analysis of open source projects on software development platforms will play an elementary role in the future. Since it is proven that more satisfied developers are more productive and thus also increase the probability of success of a software project. Against this background, automated analysis of written data or comments can be an important building block for identifying the mood in open source projects such as on GitHub. This requires a central collection of data from open-source projects. Sentiment analysis tools already exist that have been trained and evaluated with data from platforms such as GitHub. The goal of this work was to develop a concept that enables automated analysis of sentiment and its visualization through an extension for Firefox. This is preceded by the automatic URL recognition of the software development platform GitHub. The implementation is intended to help developers identify potential data for sentiment analysis on GitHub by means of automated crawling, in order to make a decision regarding participation in an open-source project on this basis. The concept development includes the aspects of data collection, sentiment analysis, and data visualization. For the practical evaluation of the concept, a survey of GitHub-using subjects from the university environment was conducted. The plugin as a Firefox extension essentially shows that sentiment analysis is seen as a supportive system to in turn positively impact through the polarity of written comments between developers. In perspective, it is necessary to perform a crawl of the data on GitHub to both capture the data collection in practice and collect more data to optimize the accuracy of sentiment analysis.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 1
- 1.2 Ziel der Arbeit . . . 2
- 1.3 Struktur der Arbeit . . . 3

2 Grundlagen **5**
- 2.1 Datenbank . . . 5
- 2.2 Repository-Mining . . . 5
- 2.3 Sentiment-Analyse . . . 7
  - 2.3.1 Natural Language Processing . . . 8
- 2.4 Verwendete Tools und Bibliotheken . . . 9
  - 2.4.1 Python . . . 9
  - 2.4.2 Crawler . . . 9
  - 2.4.3 Python-Flask . . . 10
  - 2.4.4 MariaDB . . . 11
  - 2.4.5 Senti4SD . . . 11
  - 2.4.6 Austauschformat . . . 13

3 Verwandte Arbeiten **15**

4 Konzeptentwicklung **19**
- 4.1 Ziel des Konzeptes . . . 19
- 4.2 Entwicklungsschritte . . . 20
- 4.3 Datenerfassung . . . 22
- 4.4 Datenanalyse . . . 23
  - 4.4.1 Struktur der zu analysierenden Datensätze . . . 24
  - 4.4.2 Klassifikation einer Sentimenanalyse . . . 24
- 4.5 Datenvisualisierung mittel Webinterface . . . 25
  - 4.5.1 Kozept-Plugin . . . 25
  - 4.5.2 Konzept-Webseite . . . 26

5 Implementierung **27**
- 5.1 Implementierung der Aufbau . . . 27
- 5.2 Datenquellen . . . 29

5.2.1 Repositories 29
5.2.2 Issues, Pull-Requests und Kommentare 29
5.3 Datenextraktion 30
5.3.1 Repository-Mining 31
5.4 Datenbank-Anwendung 31
5.4.1 Datei-Tabellen 31
5.4.2 Laden der Repositories 38
5.4.3 Laden der Kommentare 39
5.5 Stimmungsanalyse 42
5.6 Visualisierung 46
5.6.1 Flask-Webinterface 46
5.6.2 Plugin 47
5.6.3 Webseite 48

6 Evaluation und Ergebnisse 53
6.1 Zielsetzung 53
6.2 Evaluationsumgebung 53
6.3 Durchführung 55
6.3.1 Technische Umsetzung 55
6.4 Testaufbau 56
6.5 Ergebnisse der Studie 56
6.6 Beantwortung der Forschungsfragen 57

7 Diskussion 61
7.1 Zusammenfassung der Ergebnisse 61
7.2 Limitationen 61
7.3 Empfehlungen 62

8 Zusammenfassung und Ausblick 65
8.1 Zusammenfassung 65
8.2 Ausblick 66

A Unterlagen zur Studie 67
A.1 Einführung 67
A.2 Fragebogen 67
A.2.1 Phase 1 67
A.2.2 Phase2 68
A.3 Einladung 70

Abbildungsverzeichnis 73

Literaturverzeichnis 75

Kapitel 1

Einleitung

Nachfolgend wird zunächst auf die dieser Arbeit zugrunde liegenden Motivation eingegangen. Diese steht damit im Zusammenhang, dass die moderne Softwareentwicklung mit immer mehr und immer komplexeren Anforderungen konfrontiert wird und Funktionen erfüllt zu haben, die seitens der Benutzer:innen gefordert bzw. gewünscht werden [42]. Das sich daraus ergebende Ziel dieser Arbeit wird im Anschluss erläutert. Abschließend wird ausgeführt, wie diese Arbeit aufgebaut ist.

1.1 Motivation

Der kontinuierliche Austausch von Informationen auf Softwareentwicklung-Pattformen wie GitHub[1] ist essentiell für einen positiven Entwicklungsprozess und damit für den Projekterfolg und die Produktivität von Entwickler:innen insbesondere dann, wenn es sich um getrennt arbeitende Entwickler-Teams handelt [18]. Allerdings sind Open-Source-Projekte wie auf z.B. GitHub öffentlich, d.h. es gibt nicht zwingend Teamleiter[2], die auf eine erfolgreiche Projekt-Realisierung achten. Dabei ist erwiesen, dass Entwickler:innen zufriedener sind, wenn sie Software-Produkte erfolgreich realisieren [1]. Daher wäre es für Entwickler interessant zu wissen, wie beispielsweise die Stimmung in einem Projekt ist, um sich auf dieser Basis entscheiden zu können, ob sie sich an der Entwicklung beteiligen wollen oder nicht. Deswegen spielt im Bereich der Softwareentwicklung die Stimmungsanalyse (engl.: Sentiment Analysis (SA)) eine wesentliche Rolle und stellt ein aktuelles Forschungsgebiet im Bereich Text-Mining dar [31]. Die Stimmung kann auf der Basis von Texten in Bezug auf die Polarität und Emotionen gemessen werden. Die Polarität bzw. emotionalen

[1] https://github.com/

[2] In dieser Arbeit wird aus Gründen der besseren Lesbarkeit nicht durchgängig gegendert. Im Falle der maskulinen Form impliziert diese immer die feminine Form.

Aspekte werden dabei positiv, negativ und neutral kategorisiert. [3] Eine grundlegende Aufgabe der Stimmungsanalyse besteht darin, diese Polarität eines gegebenen Textes auf der Ebene des Dokuments oder des Satzes zu klassifizieren - unabhängig davon, ob die in einem Dokument oder einem Satz ausgedrückte Stimmung positiv, negativ oder neutral ist [29]. Es existieren bereits Tools, die die Stimmung messen können, wie Senti4SD oder RoBerta, die auf der Grundlage von Daten der Plattformen GitHub oder Stack Overflow trainiert und evaluiert wurden.

Wenn ein Entwickler ein Open-Source-Projekt betrachtet oder daran mitwirken möchte, weiß dieser aber noch nicht, ob sich die Mitarbeit lohnt. Die Stimmungsanalyse von Open-Source-Projekten soll dabei helfen, dem Entwickler einen weitergehenden Einblick in ein Projekt zu geben. Es wäre in diesem Zusammenhang von großem Vorteil, wenn ein Tool konzipiert wird, um die Stimmung der Kommentare bei GitHub automatisch zu analysieren.

1.2 Ziel der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung einer Software-Lösung auf Basis eines neu entwickelten Konzeptes zur Erkennung von Stimmung in Open-Source-Projekten. Dabei stehen im Fokus das automatisierte Crawlen, die Datenaufbereitung, die sich daran anschließende automatisierte Stimmungsanalyse mittels eines Plugins sowie eine Webseite zur Visualisierung der Stimmung. Eine Software-Lösung zur Vorhersage und Charakterisierung von Stimmung innerhalb von Open-Source-Projekten kann eine Firefox-Erweiterung sein. Aufbauend auf der Evaluation des vorgestellen Konzeptes werden im Rahmen dieser Arbeit diefolgenden Forschungsfragen (engl.: Research Questions (RQs)) beantwortet:

RQ1

Gibt es bei den Nutzern einen Bedarf an einer Stimmungsanalye für GitHub Projekten?

RQ2

Unterstützt diese Firefox-Plugin-Erweiterung die automatische Stimmungsanalyse von GitHub-Open-Source-Projekten?

RQ3

Unterstützt die Webseite die Nutzer, besondere Funktionen und Filter nach die Stimmung von GitHub Projekten zu identifizieren?

[3] https://wiki.edu.vn/wiki20/2020/12/31/stimmungsanalyse-wikipedia/

1.3 Struktur der Arbeit

Zunächst werden die für das Verständnis der Konzeptentwicklung und -implementierung notwendigen Grundlagen in Kapitel 2 beschrieben. In Kapitel 3 wird auf verwandte Forschungen eingegangen und erläutert, wie sie sich von der vorliegenden Arbeit unterscheiden. Außerdem wird die Einordnung und Abgrenzung dieser Arbeit erläutert. Im dann folgenden Kapitel 4 wird die Entwicklung des Konzepts zur automatisierten Stimmungsanalyse von Open-Source-Projekten auf GitHub mittels einer Firefox-Erweiterung und eines Webinterfaces vorgestellt. In Kapitel 5 wird die Implementierung des Konzepts beschrieben. Dazu wird die Umsetzung des Sentiment-Analyse-Verfahrens anhand der einzelnen, definierten Arbeitsschritte erläutert. In Kapitel 6 werden die im Rahmen einer Studie ermittelten Ergebnisse des Verfahrens evaluiert. Das Kapitel 7 beinhaltet die Diskussion, die zunächst Limitationen erläutert werden. Abgeschlossen wird dieses Kapitel durch eine Empfehlung für den Nutzen und zukünftige Weiterentwicklungen sein. Schließlich wird in Kapitel 8 eine Zusammenfassung präsentiert und zusätzlich einen Ausblick gegeben, wie das Firefox-Plugin in Zukunft erweitert werden könnte.

Kapitel 2

Grundlagen

Es werden in diesem Kapitel die notwendigen Grundlagen zum besseren Verständnis dieser Arbeit ausgeführt. Dazu wird zunächst in Abschnitt 2.1 auf die Funktion einer Datenbank eingegangen. Dann wird das Repository-Mining in Abschnitt 2.2 erläutert. In Abschnitt 2.3 wird die Sentiment-Analyse erklärt. Anschließend werden in Abschnitt 2.4 Tools behandelt, die für bestimmte Teilaufgaben dieser Arbeit verwendet werden.

2.1 Datenbank

Der wachsende Bedarf an Daten und Informationen macht das Datenmanagement im täglichen Leben unverzichtbar, z. B. persönliche Informationen in sozialen Medien, medizinische Daten oder Geschäftsdaten betreffend [15]. Eine Datenbank ist eine Sammlung gespeicherter und spezifizierter Daten und wird definiert als ein System mit einer bestimmten Struktur und Form mit minimaler Redundanz [23]. Sie wird von einem zentralen Kontrollsystem verwaltet und kann von einem oder mehreren Personen gleichzeitig genutzt werden.

Diese Datensammlung, die miteinander verbunden und auf einem Hardware-Server gespeichert ist, sowie eine Software zur Bearbeitung der Daten sind in Tabellen mit einer regelmäßigen Struktur gruppiert. Das bedeutet, dass alle Daten an einem bestimmten Ort gespeichert sind, der bei Bedarf leicht gefunden werden kann. Aus diesem Grund wird der Zugang zu Informationen mit Hilfe einer Datenbank wesentlich erleichtert [40].

2.2 Repository-Mining

GitHub ist die größte Hosting-Webseite, die Quellcodes für die kollaborative Zusammenarbeit bereitstellt und auf dem Versionskontrollsystem Git

basiert. Sie enthält eine Reihe von Funktionen, die die Teamarbeit und die laufende Diskussion über den Verlauf eines Software-Projektes fördern. GitHub verwendet ein Fork&Pull-Kollaborationsmodell, bei dem Entwickler ihre eigenen Kopien eines Repositorys erstellen und Anfragen einreichen können, ob der Projektbetreuer ihre Änderungen in den Hauptzweig des Projekts einfügt [48]. Github APIs (oder Github ReST APIs) sind die APIs, die Sie für die Interaktion mit GitHub verwenden können. Sie ermöglichen es Ihnen, Repositories, Branches, Issues, Pull Requests und vieles mehr zu erstellen und zu verwalten. Um öffentlich verfügbare Informationen (wie öffentliche Repositories, Benutzerprofile usw.) abzurufen, können Sie die API aufrufen[32].

Ein Software-Repository, kurz Repo genannt, ist ein Speicherort für Softwarepakete. Oft wird auch ein Inhaltsverzeichnis zusammen mit Metadaten gespeichert. Ein Software-Repository wird in der Regel von Source-Control- oder Repository-Managern verwaltet [33]. Die Kommentare, die jedes Repository in seinen Issues hat, können den Entwicklern eine Orientierung in Bezug auf das Software-Projekt bzw. die Entscheidung geben, ob sie an dem Projekt teilnehmen oder nicht. Außerdem gibt es auch viele Beispiel-Bibliotheken in Kommentaren auf GitHub, was bei der Suche nach einer geeigneten Bibliothek für ein Projekt vorteilhaft ist. Unter anderem könnten hierbei die Bewertungen und Meinungen von anderen Personen berücksichtigt werden. Zum Beispiel könnte eine Bibliothek, über die Positives geschrieben wird, besser sein als eine, über die viel Negatives geschrieben wird [9].

Repository-Mining ist ein Prozess, mit dem Daten aus einem Versionskontrollsystem wie GitHub extrahiert werden können. Dazu werden verschiedene Bibliotheken und Tools verwendet, die in Abschnitt [2.4] erläutert werden. Dabei bewegt sich dieses Thema im Bereich des Data-Mining. Zu den extrahierten Daten gehören z.B. die Commit-Kommentare oder Issues, die innerhalb der Versionsverwaltung gemeldet werden können, und die dazugehörigen Kommentare. Zusammenfassend kann Repository-Mining als Data-Mining beschränkt auf Versionsverwaltungssysteme beschrieben werden. Jedes Repository kann optional das Issue-Tracking-System von GitHub nutzen, um Fehler und andere Probleme zu melden und diese zu diskutieren. GitHub enthält auch integrierte soziale Funktionen. Beispielsweise können sich Benutzer für Aktualisierungen anmelden, indem sie Projekte beobachten und anderen Benutzern folgen, wodurch ein ständiger Strom von Daten über Personen und Projekte entsteht, der von Interesse für die Stimmungsanalyse ist.

2.3 Sentiment-Analyse

Bei der Stimmungsanalyse (engl.: Sentiment-Analysis) geht es um die Identifizierung positiver und negativer Meinungen und Bewertungen. D.h. die Sentiment-Analyse ist ein Oberbegriff für verschiedene Analysemethoden aus dem Bereich des Text-Minings. Dabei werden einzelne Sätze oder ganze Dokumente auf ihr Sentiment analysiert. Die Klassifizierung bei der Sentiment-Analyse erfolgt in der Regel in die drei Polaritäten positiv, neutral und negativ [25]. Die meisten Arbeiten beschränken sich auf die beiden Klassen positive und negative Polarität [37] [19]. In manchen Arbeiten wird auch der Aspekt der Bewertung behandelt, d.h. es wird hinterfragt, wie sich das Vorhandensein neutraler Instanzen auf die Leistung von Merkmalen bei der Unterscheidung zwischen positiver und negativer Polarität auswirkt, um ambivalente Texte besser einordnen zu können [46].

Die Stimmungsanalyse ist auch eine computergestützte Untersuchung der Meinungen oder auch Aussagen (engl.: opinions), Einstellungen und Emotionen von Menschen [28].„Opinions“ sind subjektive Äußerungen, die Gefühle über ausdrücken (z. B. „I like this tool“). Der Grund für das zunehmende Interesse an Opinion-Mining (s. Abschnitt 2.2) ist, dass davon ausgegangen wird, dass Menschen es vorziehen, Ratschläge von anderen einzuholen, um diese in Projekte einzubinden [31].

Beim Opinion-Mining werden Meinungen von Personen über eine Entität extrahiert und analysiert, während die Sentiment-Analysis (SA), Unter SA wird die Identifizierung der Gefühle von Menschen zu einem Thema und dessen Aspekte verstanden [36]. Die wird in einem Text ausgedrückte Stimmung identifiziert und analysiert. Das Ziel der SA ist es folglich, Meinungen und die darin ausgedrückten Stimmungen zu identifizieren und dann ihre Polarität zu klassifizieren (s. Abb. 2.1). Die Sentiment-Analyse ist ein Verfahren zur Verarbeitung natürlicher Sprache und zur Analyse der Einstellung eines Sprechers oder eines Verfassers eines Textes gegenüber Produkten, Dienstleistungen, Organisationen, Personen, Themen oder Ereignissen [36].

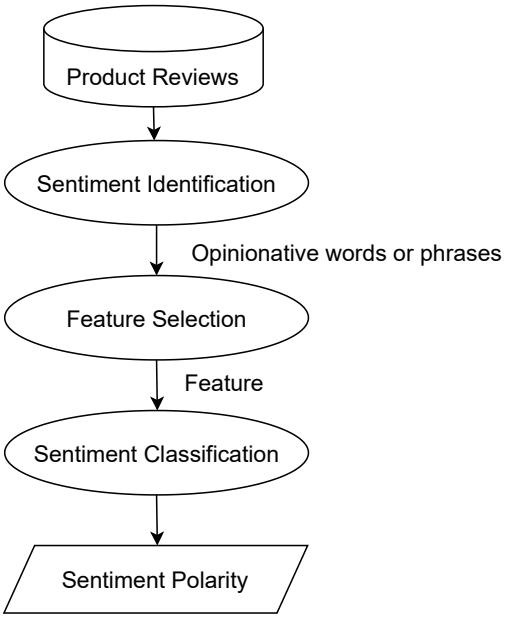


Abbildung 2.1: Sentiment-Analyseverfahren für Produktbewertung [31]

Um Meinungsäußerungen oder Stimmungswörter zu generieren und zu untersuchen, wird in dieser Arbeit das Natural Language Processing (NLP) eingesetzt, auf das im folgenden Unterkapitel näher eingegangen wird. Die Identifizierung subjektiver Einstellungen aus großen sozialen Datenmengen ist ein Fokus im Bereich von Data-Mining und NLP.

2.3.1 Natural Language Processing

Die natürliche Sprachverarbeitung (engl.: Natural Language Processing (NLP)) dient als Grundlage für Sentiment-Analysis-Tools. NLP ist eine Methode bzw. ein Bereich der künstlichen Intelligenz, der Maschinen die Fähigkeit verleiht, menschliche Sprachen zu lesen, zu verstehen und deren Bedeutung abzuleiten. Per Definition ist Natural Language Processing [26]:

> A theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications.

Übers.: Eine theoretisch motivierte Reihe von Computertechniken zur Analyse und Darstellung natürlich vorkommender Texte auf einer oder mehreren Ebenen der linguistischen Analyse mit dem Ziel, eine

menschenähnliche Sprachverarbeitung für eine Reihe von Aufgaben oder Anwendungen zu erreichen.

In dieser Definition ist die Aussage „ Reihe von Computertechniken“, die die verschiedenen Methoden zur Verarbeitung natürlicher Sprache beschreiben. Unter anderem gibt es einfache regelbasierte Methoden, wie die von Hutto und Gilbert [13] entwickelten. Diese Methoden können positive und negative Polaritäten aus Texten ableiten, um natürliche Sprache zu verarbeiten.Mit Hilfe der Verarbeitung natürlicher Sprache sind in der Lage, die in einem Text enthaltenen Emotionen zu verstehen, z. B. die positive oder negative Sichtweise des zentralen Themas, die kritische Sichtweise oder die bejahende Sichtweise. Mit„natürlich vorkommender Texte“sind Texte gemeint, die nicht künstlich für die Analyse erzeugt wurden. Dies ist wichtig, weil neuronale Netze nicht mit künstlich erzeugten Datensätzen trainiert werden, sondern idealerweise mit Datensätzen, die bereits aus einem Bereich des Software-Engineering stammen.

2.4 Verwendete Tools und Bibliotheken

2.4.1 Python

Python ist eine High-Level-Programmiersprache, die auf die Lesbarkeit des Codes abzielt. Die Zeilen für die Webcode-Entwicklung sind kürzer als bei anderen Sprachen. Python ist eine hochentwickelte Programmiersprache, die ein objektorientiertes Programmieren im Web-Bereich sowie eine sehr umfangreiche Standardbibliothek umfasst, die den Webentwicklungscode einfach und kurz gestaltet. Die Bibliotheken von Python enthalten von der Python-Gemeinschaft bereitgestellte Funktionen, die einfach heruntergeladen und je nach Entwicklungsbedarf verwendet werden können. Ursprünglich wurde Python für Webserver entwickelt, um den auf dem Server eingehenden Datenverkehr zu verarbeiten[4] [12].

Die Verfügbarkeit einer zahlreiche Funktionen umfassenden API (Application Programming Interface) ist auf die Verwendung von Python zurückzuführen.

2.4.2 Crawler

Die meisten Internet-Seiten unterhalten im Hintergrund einen Crawler. Es ist jedoch nicht festgelegt, auf welche Weise sie verlinkt werden müssen. Außerdem brauchen Benutzer bzw. Entwickler oft viel Zeit für die Suche nach einer Datei. Aus diesem Grund wurde ein automatisches Crawling-Tool benötigt. Benutzer, die an einem solchen Tool interessiert sind, können daher bei diesem Schritt aufhören und es verwenden. Ein Web-Crawler

wird für die Web-Bestellung im World Wide Web (www) eingesetzt. Er spielt eine wichtige Rolle innerhalb einer Suchmaschine, z.B. zum Auffinden neuer Seiten für die Indizierung oder für das regelmäßige Neu-Scannen und Aktualisieren des Indexes mit neuen Informationen. Ein fokussierter Crawler kann als Schlüsselkomponente einer vertikalen Suchmaschine betrachtet werden[17]. Er versucht, alle oder einige explizite Hyperlinks und HTML-Inhalte von verschiedenen Websites zu sammeln, die für ein bestimmtes Thema relevant sind, und filtert die irrelevanten Seiten heraus. So kann fokussiertes Crawling verwendet werden, um Daten für einen einzelnen Benutzer oder eine Gemeinschaft zu generieren, die an einem bestimmten Thema interessiert ist, oder um automatisch Webverzeichnisse wie GitHub zu erstellen. Der Web-Crawler ist in dieser Arbeit in Python aufgebaut.

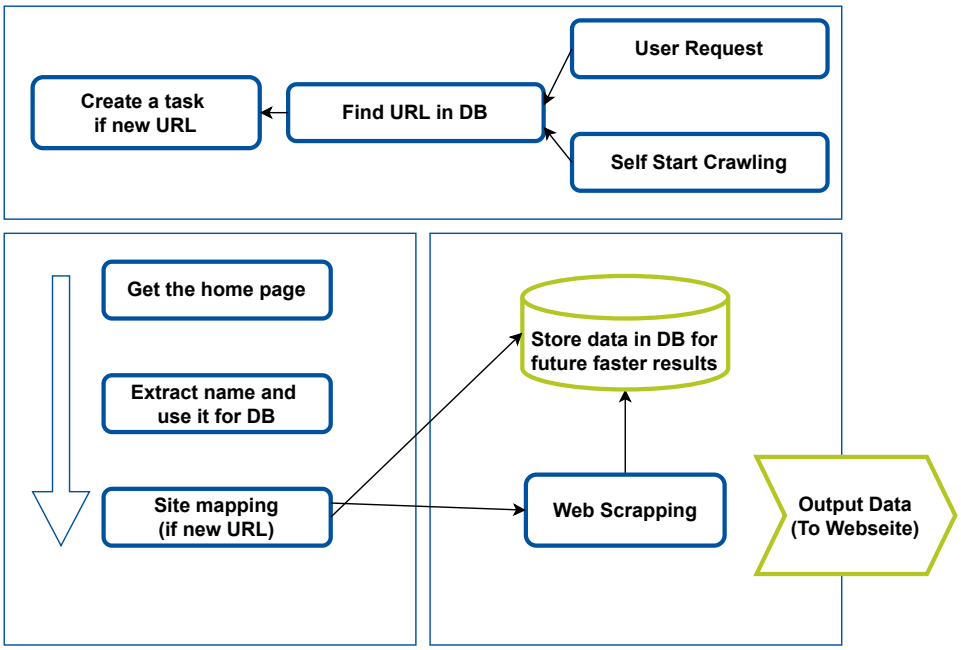


Abbildung 2.2: Übersicht des Web-Crawlers[17]

2.4.3 Python-Flask

In dieser Arbeit wird Python-Flask verwendet. Flask ist ein Mikro-Framework von Python, das die grundlegende Funktionalität eines Web-Frameworks bietet und das Hinzufügen weiterer Plugins ermöglicht, so dass die Funktionalität und der Funktionsumfang auf eine neue Ebene erweitert werden können [16]. Flask wird als Mikro-Framework von Python

bezeichnet, weil es die zentralen Funktionen einfach aber in Bezug auf die Entwicklung erweiterbar macht [30]. Es kann auch verwendet werden, um bei der Entwicklung von Webanwendungen Zeit zu sparen. Die Struktur von Flask ist in die zwei Bereiche statische Dateien und Template-Dateien unterteilt. Die Vorlagendatei enthält alle Jinja-Vorlagen einschließlich der HTML-Seiten, während die statische Datei den gesamten statischen Code enthält, der für eine Website benötigt wird, z. B. den CSS-Code, den JavaScript-Code oder die Bilddateien[16].

2.4.4 MariaDB

Datenbanken werden mit verschiedenen Sprachen erstellt, wobei MariaDB ein sogenanntes relationales Datenbankmanagementsystem (RDBMS) und eine der am häufigsten verwendeten relationalen Open-Source-Datenbanken ist. Die MariaDB-Datenbank wurde von den ursprünglichen MySQL-Entwicklern erstellt und soll Open-Source bleiben [5].

Die neuen Funktionen unterscheiden sich jedoch mit anderen Datenbanken-Sprachen und umfassen neue Speicher-Engines wie Aria, ColumnStore und MyRocks [39]. Eine Möglichkeit, dies zu tun, ist die Bereitstellung von INDEX- und TABLE-Statistiken durch eine Funktion, die sinnvolle Benutzerstatistiken genannt wird und mehrere neue Informationsschema-Tabellen sowie mehrere neue FLUSH- und SHOW-Befehle enthält. Diese Tabellen und Befehle können verwendet werden, um die Serveraktivität besser zu verstehen und die Ursachen für die Belastung der Datenbank zu ermitteln. Eine weitere, neue Funktion in MariaDB ist die Tabelleneliminierung[5].

2.4.5 Senti4SD

Senti4SD ist eine Erweiterung von SentiStrength. Neben dem lexikonbasierten Ansatz werden auch eine schlagwortbasierte und eine semantische Analyse verwendet. Ähnlich wie bei SentiStrengthSE soll die Kommunikation der SE-Domäne berücksichtigt werden. Senti4SD stellt somit eine „Sentiment-Polaritätserkennung für die Softwareentwicklung“ dar[8]. Die Open-Source-Anwendung läuft unter Java 8 oder R [1].

In den letzten Jahren wurden mehrere Tools entwickelt, um Stimmungen in Software-Artefakten zu erkennen, z.B. SentistrengthSE , Senti4SD oder SentiCR. Studien zufolge wird das Tool in dem Bereich des Software-Engineering ausgewählt, der eine höhere Genauigkeit aufweist [28]. Zufolge wurden vier Benchmarking-Studien durchgeführt, um die Leistung von Stimmungsanalyse-Tools zu bewerten [28].

[1] https://github.com/collab-uniba/Senti4SD

Tabelle 2.1: Performanz der Tools angewandt auf den GitHub-Datensatz[34]

Polarity Class	Senti4SD			SentiCR			SentiStrength-SE		
	P	R	F1	P	R	F1	P	R	F1
Negative	.92	.90	91	.90	.63	.74	.79	.77	.78
Neutral	.90	.93	92	.76	.94	.84	.78	.86	.82
Positive	.95	.91	93	.89	.85	.87	.86	.76	.81
Micro-avg.	.92	.92	92	.82	.82	.82	.80	.80	.80
Macro-avg.	.92	.92	92	.82	.82	.82	.80	.80	.80

In einer Studie wird die Leistung der drei Tools (Senti4SD, SentiCR und SentiStrength-SE) in der plattformübergreifenden Umgebung sowohl nach Polaritätsklassen als auch insgesamt angegeben. Lin et al. [27] vergleichen fünf Sentiment-Analyse-Tools, nämlich SentiStrength, NLTK, Stanford CoreNLP, SentiStrength-SE und Stanford CoreNLP anhand von folgenden drei Datensätzen: Bewertungen von mobilen Apps, Stack-Overflow-Beiträge und Jira-Kommentare. In Bezug auf die Anzahl der korrekten Vorhersagen schneiden SentiStrength-SE, Stanford CoreNLP und SentiStrength für einen der Datensätze am besten ab. Islam et al.[22] vergleichen drei Tools, und zwar SentiStrength-SE, Senti4SD und EmoTxt, mit folgenden drei Datensätzen: Jira-Problemkommentare, Stack- Overflow-Beiträge und Code-Review-Kommentare. Der Vergleich zeigt, dass SentiStrength-SE den höchsten makrogemittelten F1-Score für Jira-Problemkommentare und Code-Review-Kommentare erzielte. Gleichzeitig erzielte Senti4SD die beste Leistung für den Stack- Overflow-Post-Datensatz. Novielli et al.[35] vergleichen vier Tools, und zwar Senti4SD, SentiStrength-SE, SentiCR und SentiStrength, mit den vier Datensätzen Stack-Overflow-Posts, Jira-Problemkommentare, Code-Review-Kommentare und Stack-Overflow-Posts mit Java-Bibliotheken. Sie fanden heraus, dass Senti4SD den höchsten makrogemittelten F1-Score für den Stack-Overflow-Datensatz erzielte, während SentiCR den höchsten für die anderen drei Datensätze erzielte. Beim Vergleich mit anderen Tools hat Senti4SD häufig gut abgeschnitten[49].
In dieser Tabelle 2.1 geht es um die Auswahl eines der vorgestellten Tools zur Sentiment-Analyse im Bereich des Software-Engineerings. Zu diesem Zweck wird ein Vergleich zwischen den Tools SentiStrength-SE , Senti4SD und SentiCR durchgeführt, um zu entscheiden, welches Tool am besten für den Einsatz in einem Sentiment-Analyseverfahren in einem Open-Source-Projekt, nämlich mit Datensätzen auf GitHub, geeignet ist.

Wie in Tabelle 2.1 zu erkennen ist, ist die Performanz (F1-Score) von Senti4SD im Vergleich zu allen vorgestellten Tools am höchsten. Hieraus lässt sich folgern, dass für eine möglichst akkurate Bestimmung von Sentimenten in Daten von GitHub Senti4SD verwendet werden kann. Außerdem kann durch Senti4SD eine höhere Genauigkeit erreicht werden und es geht in dieser Arbeit um die Kommentare und Issues innerhalb von Open-Source-Projekten, diese zentral auf GitHub zu finden sind, basiert die Leistung der Tools auf den oben erwähnten GitHub-Datensätzen [34]. Die Leistung von Senti4SD ist im Vergleich zu allen vorgestellten Tools am höchsten. Außerdem wurde Senti4SD speziell für den Zweck entwickelt, Sentiment-Analysen in einer überwachten Umgebung auf Kommunikationskanälen von Entwicklern durchzuführen [8]. Die Autoren behaupten, dass ihr Klassifikator die Fehlklassifizierungen von neutralen und positiven Beiträgen in ihrem Datensatz im Vergleich zu SentiStrength reduziert. Senti4SD verwendet einen umfangreichen Merkmalsraum, der aus Worteinbettungen, n-Grammen, Lexika und schlagwortbasierten Merkmale besteht [50].

2.4.6 Austauschformat

JSON ist ein Datenaustauschformat, das für JavaScript Object Notation mit der Erweiterung .json steht[7]. JSON ist als leichtgewichtiger Datenformattyp bekannt und wird wegen seiner guten Lesbarkeit und seiner Verschachtelungsmöglichkeiten bevorzugt. Es wird häufig in Verbindung mit APIs und der Datenkonfiguration verwendet, bei denen die Dateigröße aufgrund der Leichtigkeit des Formats eine wichtige Rolle spielt. Einer der Hauptgründe, warum JSON so erfolgreich ist, kommt daher, dass es für viele Programmierspachen, Entwicklunsplattformen etc. existierende Parser gibt, die JSON-Dokumente direkt in Objekte umwandeln können. Und weil es so weit verbreitet ist.

Da JSON von der weit verbreiteten Programmiersprache JavaScript abgeleitet ist, lässt es sich außerdem leicht in die Front-End- und Back-End-Entwicklung integrieren[2]. Im Folgenden sind einige der Datentypen (Elemente) aufgeführt, die von JSON unterstützt werden.

[2]Format von JSON in dieser Arbeit wird in Plugin verwendet

```
{
  "repo": "Test Repo",
  "date_time": 20.04.2022-12:12,
  "full_scanned": true,
  "comments": [
    {
      "name": "Autor 1",
      "date_time": 14:01.1004-18:13,
      "text": "Ich habe den Fehler behoben und neu gepusht!"
    }
  ]
}
```

CSV ist ein Datenspeicherformat, das für Comma Separated Values mit der Erweiterung .csv steht[14]. In CSV-Dateien werden Datenwerte (einfache Texte) in einem durch Kommata getrennten Listenformat gespeichert. CSV-Dateien sind in der Regel kleiner und können in Texteditoren geöffnet werden, d.h. sie können verwendet werden, um einfache Daten in Tabellenkalkulationen und Tabellen zu untersuchen. CSV-Dateien können in das JSON-Format konvertiert werden, allerdings können komplexe JSON-Dateien Lese- und Schreibfehler verursachen. CSV-Dateien stellen jedoch einige Hindernisse dar, insbesondere wenn es sich um Dateien mit einer großen Anzahl von Dateneinträgen handelt.

Nach vollständiger Verarbeitung werden die enthaltenen Ergebnisse einer CSV-Datei in einer Datenbank gespeichert. Die Datenbank enthält alle Ergebnisse, ihre Klasse, die wichtigsten zugehörigen Metadaten sowie den Namen und den Pfad jeder zugehörigen Open-Source-Projekten . Darüber hinaus bietet die Schnittstelle zur Datenbank Funktionen, über die bestimmte Schlüsseldaten extrahiert werden können. Dazu gehören die gespeicherten Ergebnisse, die Anzahl der Ergebnisse einer bestimmten Klasse und die Zählung[20].

Kapitel 3

Verwandte Arbeiten

Die Analyse von Emotionen ist in der Softwaretechnik kein neues Konzept. Von daher wird in diesem Kapitel ein Überblick über verwandte Forschungsarbeiten gegeben. Die Darstellung konzentriert sich dabei auf ausgewählte wissenschaftliche Publikationen, die einen besonderen inhaltlichen Bezug zu den im Rahmen der vorliegenden Arbeit fokussierten Fragestellungen aufweisen.

Stimmungsanalyse in text-basierter Kommunikation

SentiStrength wurde ursprünglich Großbritannien für die englischsprachige Welt entwickelt. Aufbauend auf der ursprünglichen Arbeit von Thelwall et al. aus dem Jahr 2010 wurden viele zusätzliche Erkenntnisse in das Tool und das Lexikon eingearbeitet[43]. Für Forschungszwecke kann diese Anwendung sowohl im Internet als auch als Java-Applikation [1]genutzt werden. Inzwischen haben sich in verschiedenen Ländern Organisationen und Gruppen gefunden, die dieses Tool nutzen, um die Sentimentanalyse von Texten mit anderen Sprachen mit angepassten Lexika zu unterstützen. Die Anwendung SentiStrength ist ein lexikonbasierter Klassifikator. Sie interpretiert Sätze und ganze Texte in Bezug auf ihre Stimmung.

SentiCR wurde von Ahmed et al.[2] speziell für Code-Review-Kommentare entwickelt. Basierend auf den Merkmalen von Code-Review-Kommentaren verfügt SentiCR über eine Reihe von Datenvorverarbeitungsschritten, einschließlich der Entfernung von URLs und Codeschnipseln. SentiCR beinhaltet einen zweistufigen Negationsvorverarbeitungsansatz. Zunächst erstellen Ahmed et al. eine Chunk-Grammatik (d. h. eine Reihe von Regeln, die angeben, wie Sätze gechunked werden sollten) für NLTK RegexpChunkParser, um Negationsphrasen zu identifizieren. Zweitens werden alle Verben, Adjektive und Adverbien in einer vom Chunker

[1]2http://sentistrength.wlv.ac.uk/

identifizierten Negationsphrase modifiziert, indem ihnen das Präfix not vorangestellt wird[2]. Nach der Erstellung von Merkmalsvektoren mit TF-IDF werden acht überwachte Klassifikatoren bewertet. Zur Validierung der einzelnen Algorithmen wird eine 10-fache Kreuzvalidierung durchgeführt. GBT (Gradient Boosting Tree)[10] hat unter allen acht verwendeten Algorithmen die höchste Präzision, Erkennung und Genauigkeit. Daher ist der überwachte Klassifikator in SentiCR standardmäßig GBT. Das ursprüngliche SentiCR ist darauf trainiert, einen Code-Review-Kommentar entweder als negativ oder nicht-negativ zu klassifizieren.

Extraktion von Repo

Werder und Brinkkemper haben [45] ein System zur Extraktion von Emotionen aus den Kommentaren von Softwareentwicklern in Software-Repositories entwickelt, und zwar MEME - Toward for Emotions Extraction from GitHub. Ziel von MEME ist, die Emotionsextraktion zu demonstrieren und zu automatisieren. Unter Verwendung von GHtorrent und GitHub als Datenquellen präsentierten sie eine Implementierung dieser Methode. Die Ergebnisse der sich an die Implementierung anschließenden Evaluierung des Tools zeigten, dass MEME im Vergleich zu dem R-Paket Syuzhet eine bessere Leistung der Emotionsanalyse aufweist.

Ein kompletter Prozess zur Analyse von Sentiments in Open-Source-Projekten ist Gegenstand der Arbeit von Powelske [38]. Dabei geht es um die Analyse von Stimmungen im Zusammenhang mit textbasierter Kommunikation innerhalb eines Softwareprojekts. Es wurde die Sentiment-Analysis mittels Senti4SD in einem zusammenhängenden Prozess in dem Bereich des Repository-Mining und bei der Analyse größerer Datensätze eingesetzt. Allerdings wurde kein Tool konzipiert und damit auch nicht evaluiert.

Stimmungsanalyse von Nutzerbewertungen

In einer Untersuchung im Bereich Stimmungsanalyse wurde TOUR, ein automatisiertes Tool zur Themen- und Stimmungsanalyse von Nutzerbewertungen, vorgestellt [47]. TOUR ist in der Lage, aufkommende App-Probleme über App-Versionen hinweg zu erkennen und zusammenzufassen, die Meinung der Nutzer zu App-Funktionen zu ermitteln und die wichtigsten Nutzerbewertungen zu priorisieren, um Entwicklern bei der Überprüfung zu helfen. Die dynamische Natur von App-Bewertungen bedeutet, dass sich Probleme und die Stimmung in den Bewertungen mit den App-Versionen ändern. App-Bewertungen liefern Nutzermeinungen und aufkommende Probleme (z. B. neue Bugs) zu App-Versionen. Zahlreiche Studien analysieren die Stimmung in Bezug

auf App-Funktionen, um Nutzermeinungen zusammenzufassen. Das Tool TOUR identifiziert und fasst App-Probleme zwischen verschiedenen Versionen einer App zusammen, misst die Stimmung der Nutzer in Bezug auf App-Funktionen und priorisiert die wichtigsten Bewertungen, so dass die Entwickler diese leichter überprüfen können. Die wichtigsten TOUR-Techniken umfassen einen Online-Problem-Modellierungsansatz und eine Strategie zur Emotionsvorhersage.

Plugin

Venigalla und Chimalakonda [44]entwickelten das Google Chrome-Plugin StackEmo als eine praxisnahe Anwendung, das Kommentare auf Stack-Overflow mit Emojis anreichert, die auf der Stimmung des jeweiligen Kommentars basieren. Die Datengrundlage in ihrer Arbeit war die Q-A-Seite von Stack-Overflow. Um zu ermitteln, wie StackEmo von den Benutzern wahrgenommen wird, ließen sie StackEmo mittels einer Umfrage unter 30 Universitätsstudierenden bewerten. Sie kamen zu dem Schluss, dass 83 Prozent der Teilnehmer bereit waren, das Plugin an andere Personen weiter zu empfehlen. Die Ergebnisse der Umfrage lieferten außerdem Erkenntnisse zur Verbesserung von StackEmo.

Köhler [24] hat eine automatisierte Analyse von Datenschutzerklärungen als Plugin entwickelt. Dabei wurde mit Hilfe von Informationen aus Original-DSEs ein System entworfen und implementiert, das eine DSE für den Nutzer visuell aufbereitet und zusammenfasst. Das Plugin verfügt über einen Mechanismus zur automatischen Erkennung von DSEs auf einer Webseite. Darüber hinaus wurde die Erkennung einer DSE und die Anzeige der Analyseergebnisse als Browser-Extension implementiert, die in fast allen gängigen Browsern verwendet werden kann.

Abgrenzung der Arbeit

Für die Optimierung von Stimmungsanalysen wurden bereits Verfahren erstellt, um Daten aus Open-Source-Projekten zu gewinnen. Die vorliegende Arbeit skizziert ein Konzept, um Stimmung in einem open-Source-Projekt zu erfassen und so Nutzern so zu visualisieren, damit diese einen weitergehenden Einblick in ein Projekt erlagen können, um sich beispielsweise an der Entwicklung zu beteiligen. Eine automatisierte Stimmungsanalyse von Github Projekten, die Gegenstand dieser Arbeit ist, wurde aber noch nicht durchgeführt. Die genannten Arbeiten zeigen, dass die bestehenden Verfahren an bestimmte Branchen angepasst wurden. Z.B das StackEmo, das Emojis für Kommentare auf Stack-Overflow unterstützt und darüber hinaus auch anpassbar für Branchen wie die Software-Entwicklung ist, stellte sich bei den Recherchen für diese Masterarbeit als eines der

zuverlässigsten und meist verbreitetsten Verfahren heraus.Aufbauend auf diesen Vorarbeiten wird graphicher visualisierung als Plugin und als Webseite realisiert.

Kapitel 4

Konzeptentwicklung

In diesem Kapitel wird die Entwicklung des Konzepts zur automatisierten Stimmungsanalyse von Open-Source-Projekten auf GitHub mittels einer Firefox-Erweiterung und eines Webinterfaces vorgestellt. Für diese Stimmungsanalyse wird Senti4SD gewählt. Vor diesem Hintergrund werden zunächst die Anforderungen an das Konzept ausgearbeitet. Anschließend wird die Datenerfassung erläutert. Des Weiteren wird die Datenanalyse ausgeführt. Damit ein konkretes Konzept für die Sentimentanalyse innerhalb eines GitHub-Projekts ausgearbeitet werden kann, werden die Methoden und deren ausgewählten Funktionen thematisiert. Abschließend wird das für die Szenarien-Visualisierung erforderliche Webinterface (Plugin und Webseite) vorgestellt.

Folgende Fragen gilt es im Rahmen dieser Arbeit zu beantworten:

- Auf Welcher Basis an Daten, können in Open-Source-Projekten Rückschlüsse auf die Stimmung gewonnen werden?
- Mit welchen (vorhandenen) Verfahren kann die Stimmung detektiert werden?
- Wie kann die Stimmungslage visualisiert werden?

4.1 Ziel des Konzeptes

In dieser Arbeit soll ein Plugin konzeptuell entwickelt werden, um Stimmung in Open-Source-Projekten auf GitHub zu erkennen, zu analysieren und zu visualisieren. Neben der Vorschlau im Plugin wird eine Webseite entworfen , um Details anzuzeigen. D.h. für das Webinterface gibt es die zwei Kernelemente Plugin und Webseite. Um den Werte der positiven und negativen sowie der neutralen Sentimenten darzustellen, ist eine Sentimentanalyse erforderlich, die, wie in Kapitel 2.4.5 erwähnt, durch Senti4SD durchgeführt wird. Neben der Stimungsanalyse und dem Webinterface ist

der Comment Loader für das Herunterladen der Kommentare und Issues zuständig. Alle drei sind mit der Datenbank verbunden. Analog zu den Bausteinen des Systems werden die Aspekte der Datenbank, der -erfassung, der -analyse sowie der Datensammlung und -visualisierung im nächsten Abschnitt skizziert.

4.2 Entwicklungsschritte

Zunächst werden die Issues und Kommentare während eines Software-Entwicklungsprozesses auf GitHub den entsprechenden Repositories entnommen und ihre Daten bzw. Kommunikationsdaten verarbeitet. Deswegen werden im Ablaufdiagramm die Informationen verpackt, die über einige Einzelheiten systemseitig definierter Felder verfügen. Es werden folgende Schritte durchgeführt (s. Abb4.1):

- Datenerfassung: Mittels Comment-Loader werden Kommentare und Issues von Projekten auf der Plattform GitHub kopiert und in einer eigenen Datenbank abgespeichert.
- Datenanalyse: Mittels Stimmungsanalyse werden die Daten aus der Datenbank gelesen und auf Stimmung hin untersucht. Die gewonnenen Ergebnisse werden in einer weiteren Tabelle in der Datenbank abgespeichert
- Visualisierung: Um die Stimmung in einem Projekt darzustellen, werden zwei Oberflächen dem Nutzer zur Verfügung gestellt. Eine schnelle „Mini-Ansicht" wird im Form eines Plugins direkt auf der Internetseite von GitHub zum aktuell betrachtetem Projekt bereitgestellt. Eine ausführliche Analyse des Projektes kann auf einer eigenen Webseite angezeigt werden, die auf die Stimmungs-Datenbank drauf zurückgreift.

Alle in dieser Arbeit verwendeten Attribute und Funktionen werden im nächsten Kapitel tabellarisch aufgeführt und ausführlich erläutert.

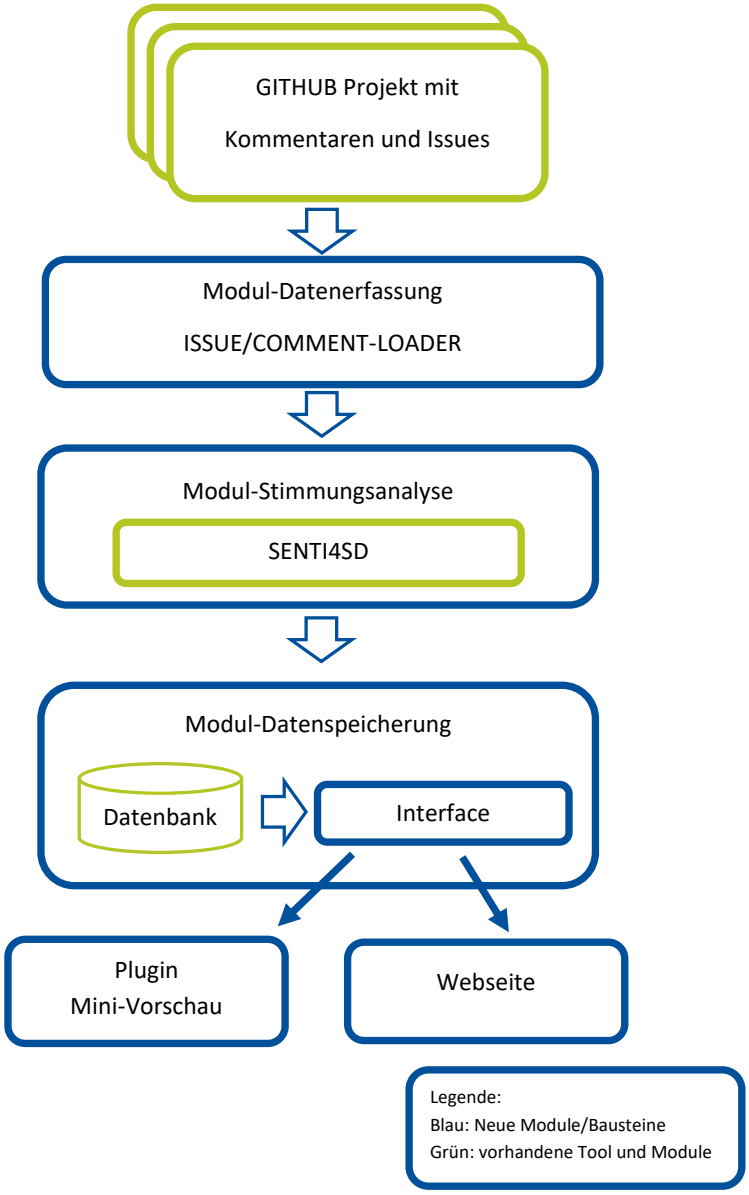


Abbildung 4.1: Struktur des Konzepts

4.3 Datenerfassung

Um jederzeit auf die Daten zugreifen und sie nutzen zu können, bietet es sich an, die Daten extra zu sammeln und in den nächsten Schritten den Text daraus zu extrahieren und ihn schließlich zu analysieren, um die dem Benutzer anzuzeigen. Dazu soll die Datenaufbereitung zur Verfügung gestellt werden.

Zentral für das Konzeptes ist das Abrufen der Daten von Plattform, die Open-Source-Projekte hostet. Im Rahmen dieser Arbeit wird dabei GitHub betrachtet. Erhobene Datensätze beinhalten Kommentare und Issues von Repository auf GitHub, dessen Aussagen positive, negative und neutrale Sentimente enthalten. Jedes Repository von GitHub hat in der Regel mehrere Issues und jedes Issue beinhaltet vielen Kommentaren, die mittels Senti4SD analysiert werden. In Abbildung 4.2 ist die exemplarische Darstellung eines Repository-Issues wie die Blätter eines Baumes zu sehen.

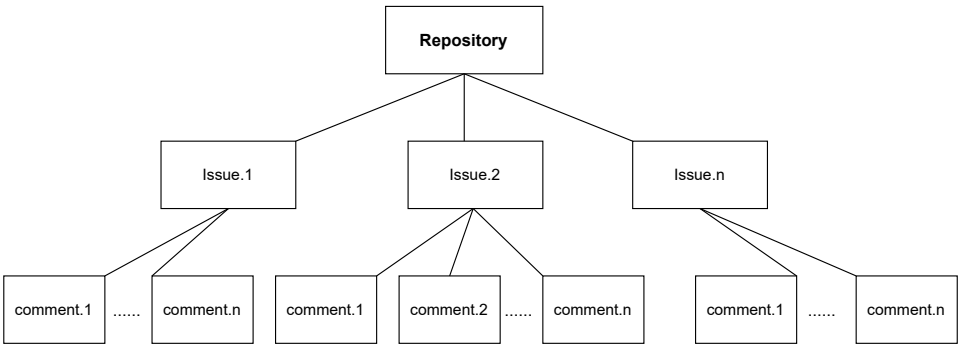


Abbildung 4.2: Vereinfachte Struktur eines Repository mit Issues

Abrufen von Daten aus GitHub

Im ersten Schritt müssen zunächst Repositories beziehungsweise die Projekte auf GitHub, die Issues und Kommentare enthalten können, erfasst werden. Um die Issues und Kommentare im zweiten Schritt zu kopieren oder speicher wird die GitHub-API verwendet. Die Suche nach Repository erfolgt über die von GitHub bereitgestellte Such-API (engl.: Search-API), die auch die Suche in Repositories und Commits ermöglicht. Da der gesamte Code auf GitHub in Repositories bzw. in Commits organisiert ist, werden zunächst diese durchsucht. Der Kommentar eines Commits wird nach dem jeweiligen Schlüsselwort durchsucht.

Zusätzlich zu den oben genannten Möglichkeiten gibt es ein Tool

namens **Seart** [1], das eine Liste von GitHub-Repositories in Form von Comma-Separated-Values (CSV)-File bereitstellen kann. Weiterhin werden einige Schlüsselkandidaten[20][2] mit Datenbank-Tabellen verwendet, die im nächsten Kapitel detailliert beschrieben werden.

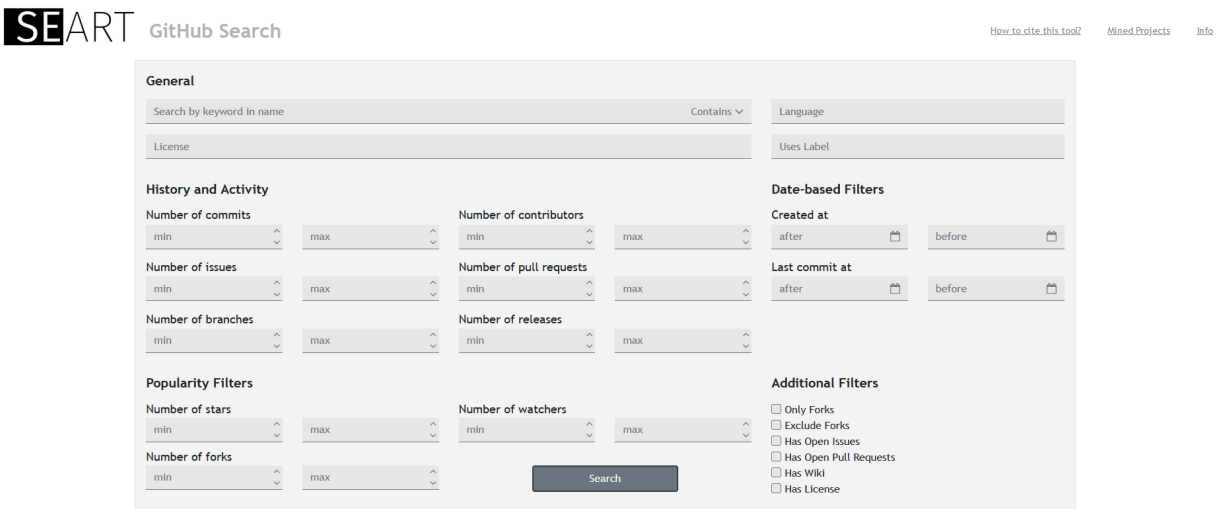


Abbildung 4.3: Seart Tool

Filter

Davon ausgehend, dass ein Entwickler, der bzw. die auf GitHub nach einem Repository sucht, manchmal lediglich die Kommentare oder Issues einer bestimmten Zeitspanne sehen will, wurde die Filter-Funktion Zeit implementiert, um die Kommentare oder Issues eines Repositorys in einem bestimmten Zeitraum zu sehen (Filter nach Zeit). Darüber hinaus wurde ein Filter zur Identifizierung von offenen oder geschlossenen Issues verwendet. Die Funktionsweise und der Zweck der Filter werden ausführlich in Kapitel 5 erläutert.

4.4 Datenanalyse

In diesem Abschnitt wird erläutert, wie eine Sentiment- bzw. Stimmungsanalyse durchgeführt wird, um Sätze und Dokumente auf positive, neutrale oder negative Stimmung hin zu untersuchen. Zu Beginn werden grundlegende Funktionen der Datenanalyse und die verwendeten Datensätze erläutert. Anschließend wird der konzeptionelle Aufbau des eigentlichen Klassifikationsalgorithmus beschrieben.

[1] https://seart-ghs.si.usi.ch/
[2] Spaltenkombinationen, die als Schlüssel möglich wären.

4.4.1 Struktur der zu analysierenden Datensätze

Die zur Stimmungsanalyse verwendeten Datensätze basieren auf den Kommentaren und Issues aus Open-Source-Projekten. Zum Analysieren dieser Daten werden bestimmte Datenformate benötigt, um die Daten analysieren zu können. Tabellenbasierte Datenbanken können unterschiedliche Formate untersützen, aber man arbeitet in der Regel doch nur mit einem. Für die Datenbank, die in dieser Arbeit verwendet wurde, sind es die Datenformate JSON und CSV eingegangen wird.

4.4.2 Klassifikation einer Sentimenanalyse

Um die Stimmung innerhalb eines Open-Source-Projektes beurteilen zu können, müssen zuerst die notwendigen Daten mittels GitHub-API angefordert werden. Diese Aufgabe übernimmt das Repository-Mining-Modul. Der Schritt der Vorverarbeitung zielt darauf ab, die Rohdaten, die als CSV-Datei sind, zu formatieren und bedeutungslose Bewertungen für die nachfolgende Analyse zu entfernen. Dann wird die Stimmungspolaritäten der Daten identifiziert und in einer Datenbank zur Analyse gesammelt.

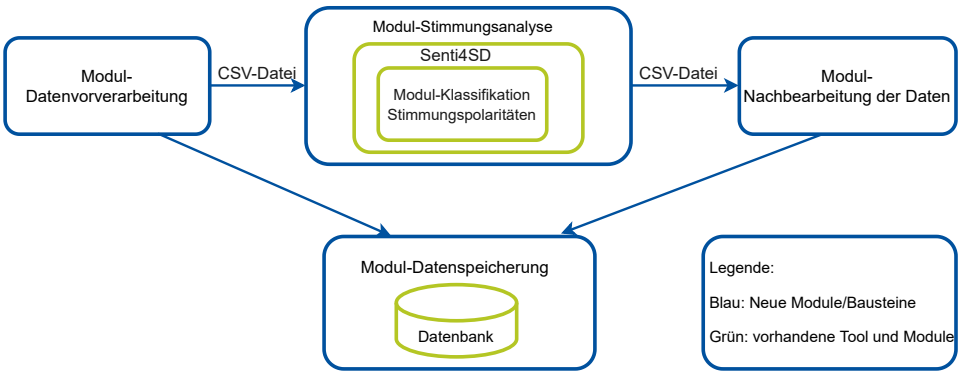


Abbildung 4.4: Klassifikation der Sentimenanalyse

Diese einzelnen Module führen Teilaufgaben oder einzelne Arbeitsschritte innerhalb des Prozesses aus. Die Module sollen in sich konsistent und vollständig sein, so dass eine Abhängigkeit zwischen den Modulen besteht. Es wird also eine maximale Kohärenz der Module angestrebt.

Sentiment-Bewertung

Die Bewertung der Stimmung innerhalb eines Open-Source-Projekts wird in dieser Arbeit durch die textbasierte Issues und Kommentare in GitHib bestimmt. Ein Text klassifiziert auf der Grundlage seiner Stimmungspolarität als positiv, negativ oder neutral. Eine Metrik ist ein Wert. Z.B. die Anzahl an positiven Wörtern gegen über negativen. Aus dem lässt sich

Positiv	Neutral	Negativ
+1	0	-1

Tabelle 4.1: Numerischen Werten zu Polaritätsklassen

Stimmung ableiten. Ein Verfahren nutzt eine oder mehrere Metriken, um Rückschlüsse auf Stimmung zu ermöglichen. Eine Metrik, die sich für eine Beurteilung eignet, ist der Emotionalitäts-Bewertung. Besonders hilfreich für die Bewertung wäre diese Metrik, die die Gesamtheit der Stimmungen in der Issues und Kommentare auf eine Kennzahl abbildet. Emotionalitäts-Bewertung wurde von Horstmann [21] benannt(s.Tabelle 4.1).

4.5 Datenvisualisierung mittel Webinterface

Für die Darstellung von Stimmung in Open-Source-Projekten kann es sinnvoll sein über einfache grafische Benutzeroberflächen im Internet (Web-Interfaces) die Stimmung in einem Projekt zu visualisieren. Mittels Plugin kann eine kompackte Darstellung zur verfügung stehen. Für das Ziel, mit Hilfe des Plugins Stimmungen zu analysieren, wurde eine grafische Benutzeroberfläche entworfen, die sowohl eine hohe Benutzerfreundlichkeit aufweist als auch eine gute Lesbarkeit bietet.

Bei Webinterface handelt es sich um eine Schnittstelle zu einem System, auf das über das Hypertext-Transfer-Protocol (HTTP) zugegriffen werden kann. Dies ist eine grafische Benutzeroberfläche (engl.: Graphical User Interface (GUI)), über die ein Benutzer mit Hilfe eines Webbrowsers mit dem System interagieren kann, oder ein Webdienst, über den das System Daten oder Funktionen für andere Systeme bereitstellt. Während der Fokus bisher auf der Struktur der Webinterfaces lag, sind im Hinblick auf das Design der Webinterfaces auch Aspekte der Interaktion, d.h. das Plugin betreffend, und des Inhalts, d.h. die Webseite betreffend, zu untersuchen.

4.5.1 Kozept-Plugin

Der Bereich Webinterface und Plugin befasst sich mit dem internen Aufbau und den Strukturen des Plugins. Das Plugin ist im Wesentlichen in drei Teile gegliedert. Der erste Teil des Plugins ist die Datenbanklogik, bei der es um alle Datenbankzugriffe geht und die die Bereitstellung von Daten über eine REST-API ermöglicht. Der zweite Teil enthält die gesamte Logik rund um Berechnungen und die Speicherung von Vorhersagen. Beides wird auf der Serverseite ausgeführt. Der letzte Teil umfasst die gesamte Darstellung in der Weboberfläche und die Aufbereitung der Daten, um sie

in einem Diagramm vergleichbar darstellen zu können. Dieser Teil findet auf der Client-Seite statt. Während die ersten beiden Teile komplett in Python geschrieben sind, verwendet die Präsentationsschicht neben HTML und CSS auch JavaScript als Programmiersprache. Es kann mit solchem Aussehen die genannte Informationen anzeigen (s. Abb 4.5).

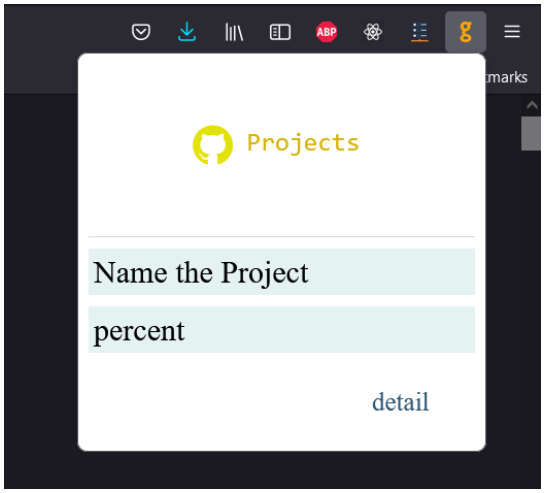


Abbildung 4.5: Ansicht des Konzepts des Plugins

4.5.2 Konzept-Webseite

Das Entstehen neuer Kommunikationsformen und erweiterter Design- und Interaktionsmöglichkeiten haben dazu beigetragen, dass sich das Konzept der Webseite zu einer dynamischen und interaktiven Webschnittstelle entwickelt hat[3].

Eine Webseite kann eine grafische Benutzeroberfläche bzw. GUI sein, bei der der Benutzer mit dem System interagiert. Mittels Webbrowser können von einem Systemen verschiedene Funktionen und Daten zur Verfügung gestellt werden. Die Benutzer sollten dabei nicht durch zu viele Informationen und die daraus resultierende kognitive Belastung überfordert werden. Eine Web-Benutzer-Schnittstelle oder eine Webseite ermöglicht dem Benutzer die Interaktion mit Inhalten oder einer Software, die auf einem entfernten Server über einen Webbrowser läuft. Aus diesem Grund folgt die zu entwerfende Benutzeroberfläche dem Prinzip„so wenig wie möglich, so viel wie nötig“. Beispielsweise erfolgt in dieser Arbeit die Visualisierung der Funktionen in Form eines Charts, das auf einer Webseite gezeigt wird.

[3]Vgl. Rada (2006), vgl. Jørgensen& Myers (2008).

Kapitel 5

Implementierung

Dieses Kapitel widmet sich der Implementierung des im vorangegangenen Kapitel 4 vorgestellten Konzepts. Zunächst wird der implementierte Aufbau beschrieben, die die Grundlage für das Verfahren zur Sentimentanalyse und für die Gestaltung der Webseite und des Plugins bildet. Dann werden die Datenquellen und anschließend die Daten-Extraktionen, die als Ausgangspunkt für die Analyse dienen sowie als Input für das Verfahren erwartet werden, erläutert. Ziel der Daten-Extraktionen ist es herauszufinden, welche Daten in Datenbanken gespeichert sind, um ein geeignetes Schema für die Vorverarbeitung der Tabellen zu finden. Danach wird das auf Basis der Untersuchungen des vorangegangenen Kapitels entwickelte, gemeinsame Datenbankschema vorgestellt. Anschließend wird mit Hilfe dieser Informationen und der Untersuchungen die Sentimentanalyse der Daten implementiert. Abschließend wird erörtert, wie die Anwendung für die Visualisierung umgesetzt wurde, welche Technologien eingesetzt wurden und wie die Implementierung erfolgte.

5.1 Implementierung der Aufbau

Der implementierte Afbau bildet die Grundlage für das in Kapitel 4 vorgestellte Konzept (s. Abb. 4.1) zur Sentimentanalyse sowie für die Gestaltung der Webseite und des Plugins. Die vereinfachte Darstellung soll die in Abschnitt 4.4.1 aufgeführten Details umsetzen.

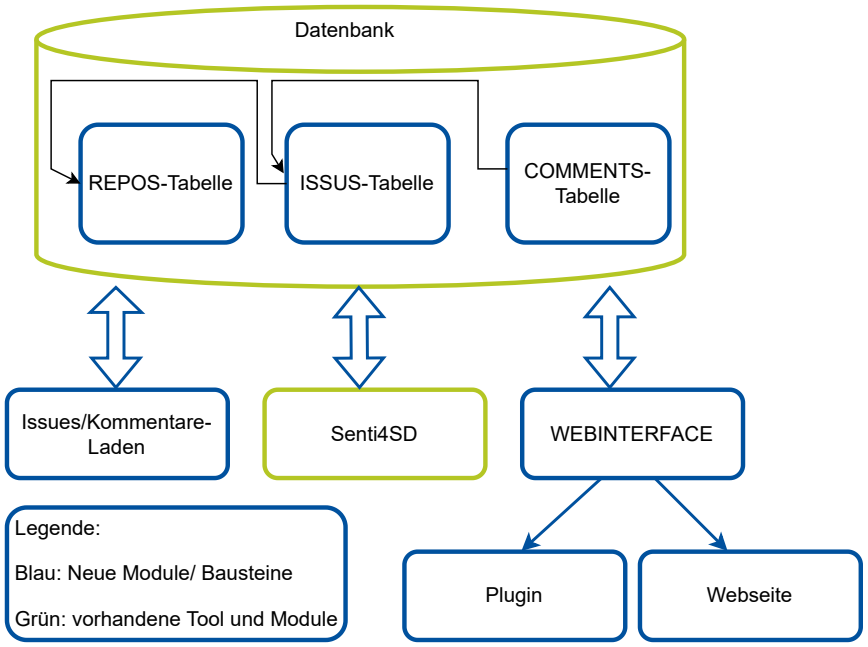


Abbildung 5.1: Übersicht der vereinfachten Darstellung

Wie in Abb. 5.1 zu sehen ist, sind die einzelnen Schritte in Teilmodule unterteilt. Jedes Teilmodul bzw. die Blätter des Baums enthält bzw. enthalten den größten Teil der Logik der Anwendung. Entsprechend der Baumstruktur ist jeder Knoten des Baums ein Python-Skript und eine Python-Bibliothek, das von dem jeweiligen Knoten oder Skript über ihn aufgerufen wird. Das Blatt ist das Webinterface, das die Entwickler sehen können. Für die Gestaltung des Webinterface und für die Durchführung der Stimmungsanalyse sind zunächst die Daten von der Datenquelle zu extrahieren (engl.: data mining).

Für die Implementierung der laufenden Arbeit besteht dieses Bereich aus den unten aufgeführten und erläuterten sechs Komponenten, die unabhängig voneinander starten und bearbeitet werden können und somit eine bessere Verwaltung des Programms ermöglichen.

- **Datenbank**: Um eine Datenbank zu strukturieren, müssen zunächst die Daten extrahiert und dann in Tabellen gespeichert werden.
- **Laden der Repositories**: Um über die benötigten Daten zu verfügen, müssen zunächst die Repositories wie eine Liste zur Verfügung stehen und heruntergeladen werden.
- **Laden der Kommentare**: Um Stimmungen analysieren zu können, sind die Texte, die sich in den Kommentaren und Issues befinden, erforderlich.

- **Stimmungsanalyse**: Die gesammelten Daten müssen analysiert und die Ergebnisse der Analyse angezeigt werden.
- **Plugin**: Ein Plugin soll einen Überblick über die Analyse und Daten geben.
- **Webseite/Webservice**: Eine Webseite wurde entwickelt. Damit sie voll funktionsfähig ist, muss der im ersten Schritt entwickelte Webserver responsiv sein. Darüber hinaus müssen alle Details verfügbar sein.

Nachfolgend wird auf alle oben kurz erläuterten Komponenten des Aufbaus im Detail eingegangen.

5.2 Datenquellen

Die Datenquellen sind die Grundlage der im Rahmen dieser Arbeit entwickelten Firefox-Erweiterung zur Bewertung der Stimmung innerhalb von Open-Source-Projekten auf GitHub. Zunächst ist zu prüfen, wie GitHub strukturiert ist und welche Informationen bzw. Daten von GitHub für sowohl die Stimmungsanalyse als auch für die Firefox-Erweiterung verwendbar sind. In nächsten Abschnitt wird Datenextraktion mit Hilfe dieser Informationen und der Untersuchungen der Daten implementiert. Diese Daten werden in einer Datenbank gespeichert und zur Analyse der Stimmung sowie für die Komponenten der Website und des Plugins verwendet.

5.2.1 Repositories

Die primäre Datenquelle sind Repositories auf GitHub, wobei diese wiederum aus Issues, Pull-Requests und Kommentaren bestehen. Ein Repository enthält mehrere Projektpakete und die dazugehörigen Metadaten. Die Repositories werden von GitHub entnommen und in einer CSV-Datei gespeichert. Zunächst muss geprüft werden, welche Möglichkeiten die GitHub-API bietet. Je nachdem, was in dieser API zu finden ist, wird jeder Teil von GitHub, der einen Kommentar hat, für diese Arbeit verwendet, also auch Issues, Pull-Requests und Kommentare. Dementsprechend müssen zunächst die notwendigen Daten extrahiert und im nächsten Schritt in einer Datenbank gespeichert werden. Danach werden diese Daten zur Analyse der Stimmung genutzt, deren Visualisierung sich daran anschließt.

5.2.2 Issues, Pull-Requests und Kommentare

Die Daten bzw. Issues, Pull-Requests und Kommentare werden in der Funktion Load-Comments in der Datenbank verwendet. In dieser Arbeit sind für die zu entwickelnde Methode Issue-Tracker bzw. **Issues** ein wichtiger Bestandteil, in denen Entwickler „Tickets“ erstellen können,

die anschließend von anderen Entwicklern kommentiert werden können. Pull-Requests werden auch bei der Arbeit mit Issues berücksichtigt, da sie sich in Bezug auf den Kommentar wie ein Issue verhalten, der einen Titel und einen Body enthält, der wiederum einen Kommentar enthalten kann. Aus diesem Grund ist in dieser Arbeit immer dann, wenn es um Issue geht, auch ein Pull-Request gemeint.

Einem oben erwähnten Ticket können Kommentare hinzugefügt werden, was eine direkte Kommunikation und Diskussion zwischen den Entwicklern ermöglicht. Wenn die Entwickler und Teilnehmer in diesen Tickets miteinander kommunizieren, entspricht das einem Forum, in dem der ursprüngliche Text zu einer Software-Ausgabe geändert oder ergänzt werden kann. D.h. die Teilnehmer oder Entwickler des Projekts stehen hier in direktem kommunikativen Kontakt. Diese Kommunikation zwischen den Teilnehmern stellt eine wertvolle Informationsquelle für die Beurteilung der Stimmung in einem Projekt dar. Deswegen sind die Kommentare zu den einzelnen genannten Issues auch für dieses Verfahren von Bedeutung.

Die Kommentare zu **Pull-Request** sind eine weitere Datenquelle, die als Ausgangspunkt für die Analyse dient. Andere wichtige Datenquellen sind neben den Issues die **Kommentare**, die als Ausgangspunkt für die Analyse dienen. Sie sind aus dem Grund nützlich, weil die Entwickler darin direkt über ihre Arbeit schreiben und die anderen Entwickler sich somit an der Entwicklung eines Software-Projektes beteiligen können. Das bedeutet, dass die Art und Weise, wie sie über die von ihnen vorgenommenen Änderungen am Quellcode berichten, einen direkten Hinweis auf die Stimmung der Kommentare geben könnte. Wenn alle Kommentare zu einem Projekt berücksichtigt werden, bieten diese eine gute Grundlage, um die Stimmung innerhalb eines Projekt-Teams zu beurteilen. Die Issues und Kommentare von jedem Repository sind für den Teil **Load-Comments**(Laden der Kommentare) erforderlich.

5.3 Datenextraktion

Die Datenextraktion von GitHub erfolgte über das Repository-Mining (vgl. Abschnitt 2.2). Dabei wurden die Daten aus den beschriebenen Datenquellen extrahiert und dann in eine verwertbare Form umgewandelt, damit sie als Input für die Durchführung der Sentimentanalyse geeignet sind. Die Datenextraktion umfasst die Erstellung einer Liste der Repositories. Des Weiteren umfasst sie die Datenbereinigung (mit Application Seart), die in nächsten Abschnitt beschrieben wird.

5.3.1 Repository-Mining

Für das Repository-Mining wird eine Methode oder ein System verwendet, um die benötigen Daten zu extrahieren. Dann ist es notwendig, dass jedes Repository aus GitHub entnommen und in einer Datei darstellt wird. Neben der Extraktion der einzelnen Abschnitte, die einen Kommentar enthalten, sollen auch andere Informationen extrahiert werden, wie die Zeit, die für den Zeitfilter benötigt wird, und andere Informationen, die in den Spalten der Tabelle im Einzelnen aufgeführt werden (s. Abschnitt 5.4.1).

Application Seart

Um im Rahmen dieser Arbeit das Extrahieren der Repositories durchzuführen, wurde das Tool Seart verwendet. Seart erstellt eine Liste von Repositories, die als CSV-Dateien angezeigt werden. Seart ist so aufgebaut, dass es nach jedem Repository sucht und dann dieses der jeweiligen CSV-Datei hinzufügt. Auf diese Weise ergibt sich eine Liste mit allen Repositories. Dieser Liste kann ein Repository immer dann automatisch hinzugefügt werden, sobald eines von einem Entwickler erstellt worden ist.

5.4 Datenbank-Anwendung

Um die Architektur (s. Abb. 5.1) zu entwerfen, muss am Anfang eine geeignete Datenbank erstellt werden und dann müssen die weiteren Komponenten, die mit der Datenbank verbunden sind, konzipiert werden. Die weiteren Komponenten sind das Laden des Repositories, das Laden der Kommentare, die Stimmungsanalyse sowie die Visualisierung der Webseite und des Plugins. Nach der Extrahierung der Daten von GitHub ist zu deren Speicherung und weiterer Untersuchung eine Datenbank erforderlich.

5.4.1 Datei-Tabellen

Auf Grundlage der Daten, die GitHub selbst entnommen werden können. D. h. der Daten, die von der API von GitHub stammen, erfolgt das Erstellen von Tabellen, in denen die Daten gespeichert bzw. zwischengespeichert werden können, um sie bei Bedarf zu ändern oder an die Erfordernisse anzupassen. Es werden drei Tabellen mit den Bezeichnungen **REPOS, ISSUES** und **COMMENTS** erstellt. Die genannten Tabellen werden getrennt angelegt, um die Abfrage (engl.: query) zu erleichtern. Jede Zeile in der Tabelle steht für eine Spalte. Um die Tabellen benutzerfreundlicher zu gestalten, werden die zwei Merkmale Fremdschlüssel (engl.: foreign key) und Primärschlüssel (engl.: primary key) verwendet.

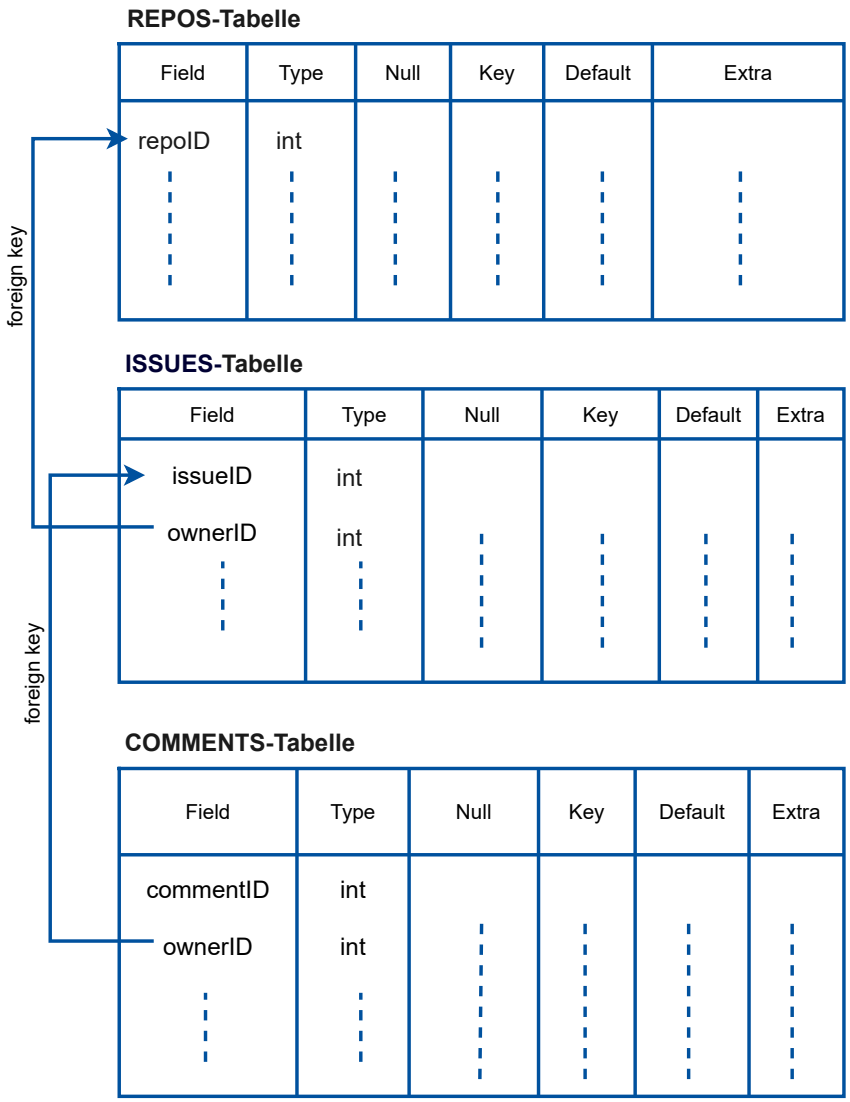


Abbildung 5.2: Ansicht der Beziehung der Tabellen

Daten in REPOS-Tabelle

Nach dem Repository-Mining ist die Datei verfügbar, die die Namen der Repositories in der Form wie „owner/reponame“ anzeigt. Dann wird die Informationen aufgenommen, die in der CSV-Datei sind und die nicht benötigt werden. Es wird der Name der Repositories benötigt, die in einer Tabelle in der Datenbank gespeichert sind. Diese Tabelle heißt REPOS, die einen Zeilennamen hat, in dem die Namen der Repositories stehen. Hier kann eine weitere Spalte mit den Kommentaren hinzugefügt werden.

Dies ermöglicht es, die Repositories mit mehr Kommentaren aus der Warteschlange zu trennen und sie früher als die anderen zu bearbeiten. Auf diese Weise hat der Entwickler eine höhere Chance, die richtigen Kommentare zu finden, wenn er nach ihnen sucht.

Die erste Spalte ist die **repoID**. Für jede hinzugefügte Zeile erhöht sich die id-Nummer, die auch der primäre Schlüssel dieser Tabelle ist, d.h. jede Zeile wird damit identifiziert. Die nächste Spalte ist der **Name**, der eindeutig ist. Dies ist der Name des Repositorys in der Form „owner/reponame“. Die dritte Spalte ist die **Commits**, die die Anzahl der Commits auflistet. Diese Spalte wird für die Sortierung der Commits verwendet. Der Grund für die Eindeutigkeit des Namens liegt darin, dass es eine weitere CSV-Datei gibt, die der vorherigen hinzugefügt werden muss. Wenn die CSV-Datei diesen Namen trägt, soll sie nicht erneut hinzugefügt sondern aktualisiert werden.

Die REPOS-Tabelle enthält noch drei verschiedene Daten. Die erste Spalte gibt der Daten an: wann ein Repository, das im vorherigen Schritt gesucht, gefunden und hergestellt wurde. D.h. Wann sein Besitzer es auf GitHub erstellt hat (**createAt**). Dieser Zeitpunkt ist der entsprechenden CSV-Datei entnommen. Die nächste Zeitspalte wird aktualisiert, wenn Senti4SD ein Repository analysiert und das Datum der Analyse in **lastAnalysis** einträgt. **lastUpdate** ist die Zeit, die prüft, wann der „Commentloader“ seine Kommentare vom Server erhält. Und wenn er seine Kommentare bekommt und es sonst nichts gibt, wird „**lastUpdate**“ aktualisiert. Das letzte Datum, an dem die Kommentare heruntergeladen wurden, erscheint. D.h. Diese Spalte trägt das aktuelle Datum ein. Die Vorteile dieser Spalten besteht darin: wenn ein Repository gesucht wird, verstanden werden kann, wie viel analysiert wurde. Und wenn ein Kommentar hinterlassen wird, wie lange die Kommentare genommen wurden. Dann ist es nun notwendig, nur die neueren zu nehmen. Andererseits kann das Senti4SD dies auch tun, indem es jedes Mal, wenn es ein Repository erhalten hat, nur die Kommentare und Issues analysiert, die z.B. zum Datum X hinzugefügt wurden. Aus diesem Grund sind diese beiden Verlaufsspalten verwendbarer als die createAt-Spalte (s. Abb. 5.3[1]).

[1]PRI:primary , UNI:unique

Field	Type	Null	Key	Default	Extra
repoID	int(11)	NO	PRI	NULL	auto_increment
name	text	YES	UNI	NULL	
commit	int(11)	YES		NULL	
createAt	datetime	YES		NULL	
lastAnalysis	datetime	YES		NULL	
lastUpdate	datetime	YES		NULL	

Abbildung 5.3: Ansicht der REPOS-Tabelle

Daten in ISSUES-Tabelle

Die nächste Tabelle trägt die Bezeichnung ISSUES (s. Abb. 5.4). Die erste Spalte **issueID** zeigt, welche Daten GitHub entnommen werden. Es wird eine eindeutige ganze Zahl als Primärschlüssel verwendet, weil es dadurch nachvollziehbar ist, ob ein Issue wiederholt wird oder nicht bzw. Ob ein Element nicht zweimal gespeichert wird. Es ist kein Auto-Increment[2], weil es in der vorherigen Tabelle eingestellt wird, dass die neue Zeile, die platziert wurde, automatisch hinzugefügt wird.

Die **ownerID** ist ebenfalls eine ganze Zahl (integer), wird aber als Fremdschlüssel betrachtet, da sie sich auf die repoID der vorherigen Tabelle bezieht. Dieser Fremdschlüssel dient dazu, eine Liste von Ideen zu erstellen und festzustellen, welche dieser issue-ids zu welchem Repository gehört. Wenn es zum Beispiel ein Repository mit einer Anzahl von Issues gibt, ist es möglich, mit Hilfe dieses Schlüssels herauszufinden, welche Issues zu welchem Repository gehören. Ein weiterer Grund, der einen foreign key erfordert, ist, dass, wenn etwas aus der REPOS-Tabelle entfernt wird, es automatisch in allen Tabellen aktualisiert wird. Und die Tatsache, dass mul (mehrere) manchmal in einem Repository gesehen wird, kann es eins oder mehrere Issues haben.

Die nächste Spalte **commentCounter** zeigt die Anzahl der Kommentare

[2]Die IDs in Repo werden automatisch erhöht, d.h. sobald Repo hinzugefügt wird, wird automatisch eine ID zu den IDs hinzugefügt, da hier nicht die Original-ID von GitHub zur Verfügung steht, sondern für Issues und Kommentare die ID von GitHub selbst verwendet wird. So wird sichergestellt, dass sie unique ist. D.h. Auto-Increment wird durchgeführt, um sicherzustellen, dass die ID unique ist. Und wenn erkannt wird, dass sie unique ist und eine ID hat, ist eine Auto-Increment nicht erforderlich.

zu jedem Issue. Die vierte Spalte lautet **number**. Diese Benennung basiert auf GitHub selbst. In dem Link, der erstellt werden soll und von dem die Kommentare heruntergeladen werden, wird diese Nummer benötigt. Die fünfte Spalte der Tabelle heißt **title** und enthält den Titel des Repositories. Die Spalte, die den Textkörper und die Beschreibung enthält, wird **body** genannt. Die Spalte **createAt** entspricht der in der REPOS-Tabelle. Die Spalte **state** zeigt, ob das Issue öpenöder closedïst bzw. dessen Wert Null oder Eins beträgt. Die nächsten beiden Spalten **plusOne** und **minusOne** zeigen an, dass der Nutzer eine Bewertung (engl.: vote) abgeben kann, indem er etwas etwas Positives (like) oder Negatives (dislike) schreibt. Die Spalte **Analyse** zeigt auch den Betrag der Analyse an, der ebenfalls eine Zahl ist.

Zusammenfassend ist festzuhalten, dass ein Repository in der ersten Tabelle gesucht und sein RepoID erkennt werden kann und mit dieser ID seine Issues in der zweiten Tabelle gefunden und entsprechend seiner Texte dessen Issues-Analysen angezeigt werden können.

Field	Type	Null	Key	Default	Extra
issueID	int(11)	NO	PRI	NULL	
ownerID	int(11)	YES	UNI	NULL	
commentCount	int(11)	YES		NULL	
number	int(11)	YES		NULL	
title	text	NO		NULL	
body	text	NO		NULL	
createAt	datetime	YES		NULL	
state	int(11)	YES		NULL	
plusOne	int(11)	YES		NULL	
minusOne	int(11)	YES		NULL	
analyse	int(11)	YES		0	

Abbildung 5.4: Ansicht der ISSUES-Tabelle

Daten in COMMENTS-Tabelle

Die dritte Tabelle COMMENTS (s. Abb. 5.5) hat acht Spalten hat. Die erste Spalte ist die **commentID**, die der Primärschlüssel dieser Tabelle

ist, d. h. die gleiche ID, die GitHub für jeden Kommentar hat. Die nächste Spalte enthält die **ownerID**, die sich genau auf die issueID bezieht, d. h. diese Spalte ist der Fremdschlüssel zur issueID der vorherigen Tabelle (s. Abb. 5.6). Mit diesen Fremdschlüsseln sind alle drei Tabellen miteinander verbunden (s. Abb 5.2). Die Spalte **body** ist ein einfacher Text. Schließlich soll dieser Text mit title und body aus der vorherigen Tabelle verbunden und analysiert werden. Die übrige Spalte sind wie in ISSUES-Tabelle. In vielen Repositories gibt es nämlich viele kommentarlose Ausgaben, z. B. wenn ein Entwickler ein Thema eröffnet oder eine Frage stellt, aber keinen Kommentar hinterlassen hat. Die andere Spalten entsprechen denen der ISSUES-Tabelle. Zu beachten ist, dass die Anzahl der Repositories konstant bleibt, bis diese in die CSV-Datei aufgenommen werden. Jedes Repository wird aus dieser Liste geholt und seine Issues werden geladen. Und wenn die Anzahl der Kommentare nicht null ist, werden auch die Kommentare geladen, ungeachtet dessen, wie viele es sind. Schließlich werden die Kommentare und Issues tiefgehend analysiert.

Field	Type	Null	Key	Default	Extra
commentID	int(11)	NO	PRI	NULL	
ownerID	int(11)	YES	UNI	NULL	
body	text	YES		NULL	
createAt	datetime	YES		NULL	
state	int(11)	YES		NULL	
plusOne	int(11)	YES		NULL	
minusOne	int(11)	YES		NULL	
analyse	int(11)	YES		0	

Abbildung 5.5: Ansicht der COMMENTS-Tabelle

Wie bereits oben erwähnt, sind der Fremdschlüssel und der Primärschlüssel die zwei zentralen Merkmale der Tabellen. Nachfolgend wird dies begründet und es wird die Verwendung der beiden Schlüssel dargestellt:

foreign key: Ein Fremdschlüssel ist ein Satz von Attributen in einer Tabelle, der sich auf den Primärschlüssel einer anderen Tabelle bezieht. Der Fremdschlüssel verbindet diese beiden Tabellen miteinander. Die Tabelle, die den Fremdschlüssel enthält, wird als untergeordnete Tabelle bezeichnet,

und die Tabelle, die den Kandidatenschlüssel enthält, als referenzierte oder übergeordnete Tabelle [41]. In der relationalen Datenbankmodellierung und -implementierung ist ein Kandidatenschlüssel ein Satz von null oder mehr Attributen, deren Werte für jedes Tupel (Zeile) in einer Beziehung garantiert eindeutig sind. Der Wert oder die Kombination von Werten von Kandidatenschlüsselattributen für ein Tupel kann für kein anderes Tupel in dieser Beziehung dupliziert werden. Da der Zweck des Fremdschlüssels darin besteht, eine bestimmte Zeile der referenzierten Tabelle zu identifizieren, ist es im Allgemeinen erforderlich, dass der Fremdschlüssel gleich dem Kandidatenschlüssel in einer Zeile der Primärtabelle ist oder keinen Wert hat[13]. Diese Rolle wird als referenzielle Integritätsbeschränkung zwischen den beiden Tabellen bezeichnet [11]. Da Verletzungen dieser Einschränkungen die Ursache vieler Datenbankprobleme sein können, bieten die meisten Datenbankverwaltungssysteme Mechanismen, die sicherstellen, dass jeder Fremdschlüssel, der nicht null ist, einer Zeile in der referenzierten Tabelle entspricht.

primary key: Im relationalen Modell von Datenbanken ist ein Primärschlüssel eine spezifische Auswahl einer minimalen Menge von Attributen (Spalten), die ein Tupel (Zeile) in einer Beziehung (Tabelle) eindeutig spezifizieren [6]. Inoffiziell sind ein Primärschlüssel die Attribute, die einen Datensatz identifizieren und in einfachen Fällen ein eindeutiges Attribut darstellen, d.h. ein eindeutiger Bezeichner mit Bezug auf Primärschlüsse. Ein formaler Primärschlüssel ist ein ausgewählter Kandidatenschlüssel (ein minimaler Superkey) und jeder andere Kandidatenschlüssel ist ein alternativer Schlüssel.

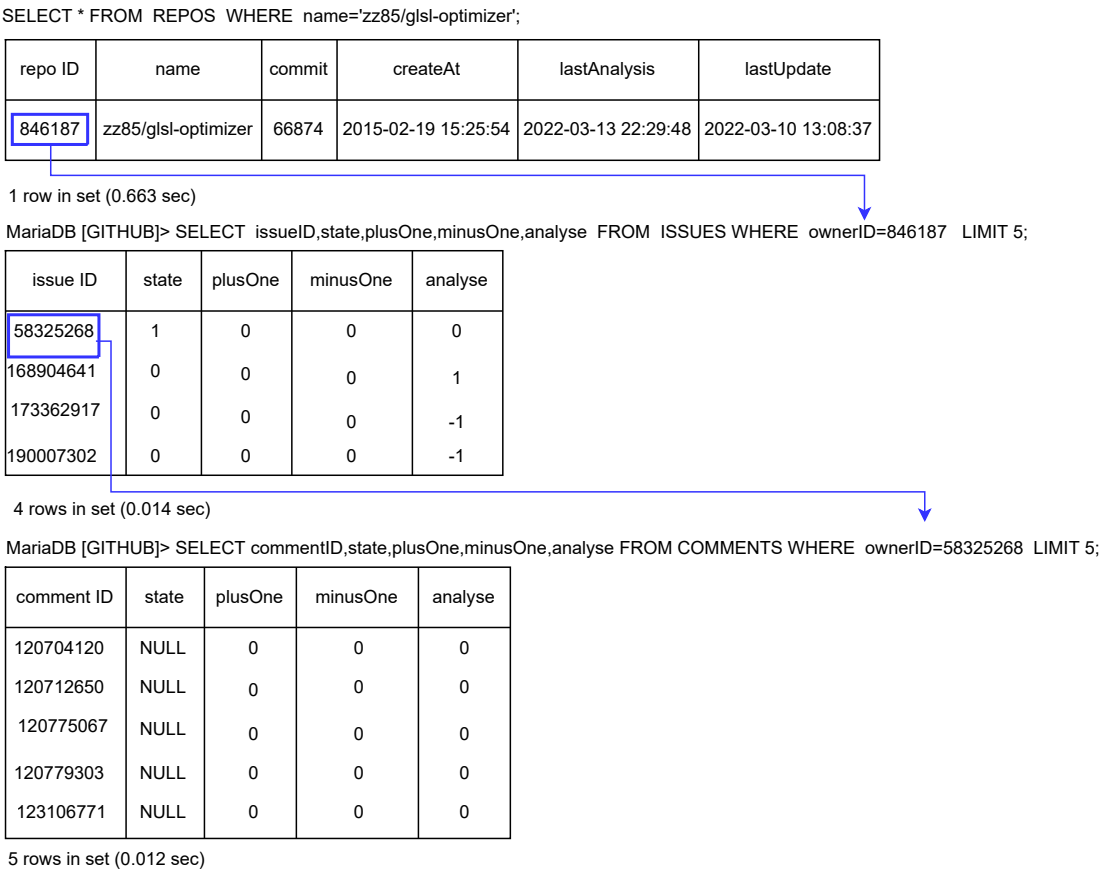


Abbildung 5.6: Verfahren des Fremdschlüssels in Repository zz85/glsl-optimize mit Limit 5

Wie in der Abbildung zu sehen ist, wurde ein Repository mit dem Namen „zz85/glsl-optimize“ aus der Datenbank ausgewählt, und die ersten 5 Issues, die über seine ID erfasst wurden, wurden analysiert und in der Ergebnistabelle angezeigt, aber da es nur 4 Issues hat, sind hier 4 Zeilen zu sehen. In der nächsten Tabelle werden die Kommentare zum ersten Issue mit seiner ID, der Analyse und den Ergebnissen in der Tabelle entsprechend der 5 limit angezeigt.

5.4.2 Laden der Repositories

Das Laden der Repositories ist die erste Komponente der Anwendung der Daten. Abb. 5.7 zeigt die hier dargestellten Schritte dieses Prozesses.

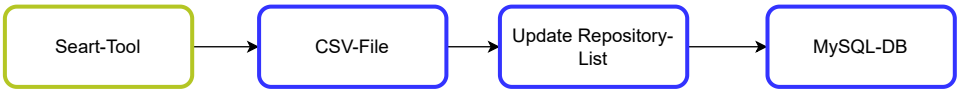


Abbildung 5.7: Laden der Repositories

Im Rahmen dieser Arbeit wurde das Seart-Tool für das Repository-Mining verwendet. Mittels dieses Tool wird eine Liste von Repositories, die in einem Form einer CSV-Datei angezeigt werden, verfügbar sein. Dementsprechend kann der Namen der Repositories nach dem Muster owner/Reponame angeboten werden. Es gibt noch eine Reihe anderer Informationen in dieser CSV-Datei, die für diese Arbeit allerdings nicht relevant sind.

Es werden nur die Namen der Repositories gebraucht. Diese sind in einer separaten Partition in der Datenbank gespeichert, die als REPOS-Tabelle (die Warteschlange des Repositorys) bezeichnet wird und einen Zeilennamen hat, in dem die Namen der Repositories stehen (vgl. Abschnitt 5.4.1). Das Konzept der Warteschlange hier bedeutet, dass das Repository, das mehrere Kommentare hat, früher verarbeitet wird. Auf diese Weise sind die Chancen höher, dass Benutzer später nach einer guten Antwort suchen. Diese Komponente des Programms kann später separat aktualisiert werden oder sogar einen anderen LOADER addieren.

5.4.3 Laden der Kommentare

In diesem Schritt werden die Kommentare, die vorher in der Comments-Tabelle gespeichert wurden, aus der API von GitHub verwendet. Die allgemeine Routine ist, die API zu befragen, die ein Anfragelimit von 5000 Anfragen pro Stunde begrenzt ist (rate limit of 5000 requests per hour. [3]). Das Ablaufdiagramm in Abb. 5.8 veranschaulicht das Verfahren das Laden der Kommentare betreffend.

[3] https://docs.github.com/en/developers/apps/building-github-apps/rate-limits-for-github-apps

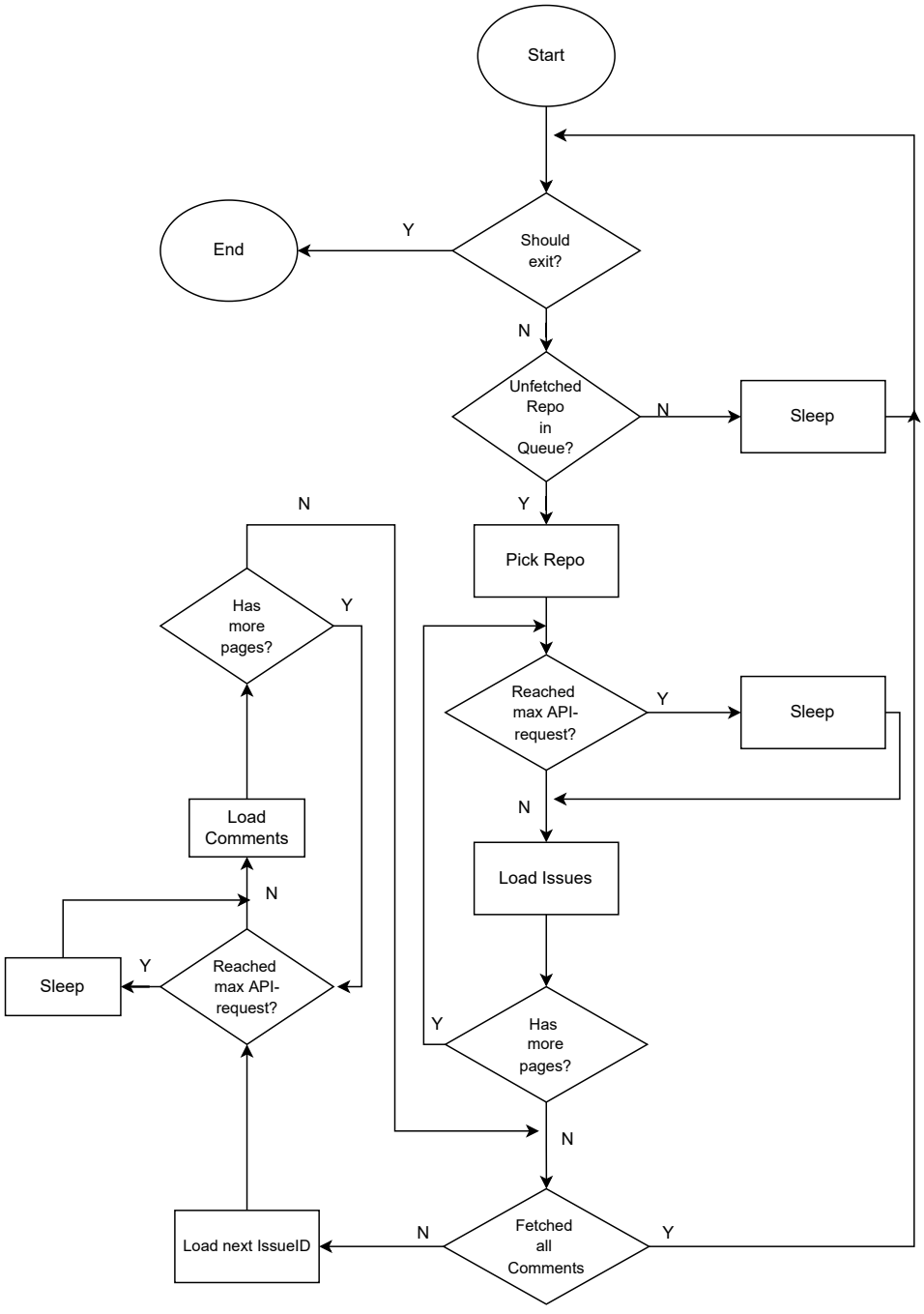


Abbildung 5.8: Verfahren des Ladens der Kommentare

Ein Repository, wird aus der Tabelle entnommen, die im vorherigen Schritt mitHilfe Seart erstellt wurde, und wird in die Lage gebracht, die Issues und Kommentare zu erhalten. Dabei wird entweder ein Maximun

von 5000 Anfragen pro Stunde erreicht oder nicht. Im falle des Erreichen des Maximums wird zurückgemeldet, dass eine Stunde gewarten werden muss. Hier wäre ein Befehl wie „Sleep“[4] notwendig und wenn diese Zeit vergangen ist, könnten die Kommentare wieder hochgeladen werden und es würde wieder geantwortet werden. Sollen zusätzlich zu den Texten der Issues auch die Issue-Kommentare geladen werden, so ist bei einem Projekt wie Tensorflow, das zum Zeitpunkt dieser Arbeit[5] 32.388 geschlossene Issues hatte, die Abfragegrenze von 5000 $\frac{Query}{h}$ schnell erreicht. Nimmt man als untere Grenze die Schätzung, dass nur $\frac{1}{5}$ aller geschlossenen Issues Kommentare haben, so ergibt sich folgendes Problem bei der Anzahl der benötigten Abfragen pro Stunde:

$$\begin{aligned} \lceil 32388 + (\frac{1}{5} * 32388) \rceil &\overset{!}{\leq} 5000 \\ 38867 &\not\leq 5000 \end{aligned} \tag{5.1}$$

Um das Problem der Limitierung zu umgehen, wird von Verfahren des Ladens der Kommentare benutzt. GitHub gibt die Daten so an, dass bei jeder Anfrage maximal 100 Antworten gegeben werden können, und wenn es mehr als 100 sind, wird zur nächsten Seite gewechselt. Aber wenn nicht die Grenze des Anteils erreicht werden, überprüft der Befehl **more comments**, ob zur Seite zwei geht oder nicht. Ob alle mit einer Anfrage (100 Anfragen pro Seite) bekommen werden kann. Hier gibt es eine Schleife, die periodisch wiederholt wird und es muss gewartet werden, bis die maximale Anzahl von Anfragen erreicht ist. Es lädt über die Zeit, aber es erreicht den Punkt, wo es ausläuft. Zum Beispiel gab es 200 Kommentare, was zweimal wiederholt wurde. Dann wurden die 200 Kommentare gespeichert, die erhaltet in zweiter Tabelle bzw. zweiter Seite werden.

„Queue is empty“ ist die gleiche Warteschlange, die im einem Programm erstellt wurde. In Anbetracht der Tatsache, dass es mehr als eine Million Repositories gibt, wird davon ausgegangen, dass diese Anzahl erreicht wurde. Nun können zwei Verfahren ausgeführt werden oder das Programm schläft oder lädt automatisch weitere Repositories dynamisch herunter, so dass es notwendig ist, Es gibt keine neue CSV-Datei. Wenn die Warteschlange nicht leer ist, holt es sich das nächste Element und wiederholt die vorherigen Schritte, sodass die vorherigen Daten nicht verloren gehen, da sie in der Datenbank gespeichert werden.

[4] ein Befehl, den für eine Pause gebraucht wird.
[5] Stand: 05.05.2022

5.5 Stimmungsanalyse

Die Stimmungsanalyse wurde mit dem Programm Senti4SD, das ausführlich in Abschnitt 2.4.5 beschrieben ist, durchgeführt. Dazu wurde GitHub-API aufgerufen und der zu analysierende Kommentar an Repository übergeben. Nach der Berechnung der Stimmung wurde die Daten in die Datenbank eingetragen. Dies wird mit allen Kommentaren aus dem Projekt wiederholt. Im Anschluss wird der aktuelle Zeitstempel mit Datum und Uhrzeit in der Datenbank unter dem Eintrag „lastAnalysis“ hinterlegt. Es gab ein Datum in der REPOS-Tabelle, das als letzter Scan (lastAnalysis) bezeichnet wurde. Seine Aufgabe bestand darin, das Repository daraufhin zu überprüfen, ob sein letzter Scan der älteste ist und seine letzte Aktualisierung (lastUpdate) neuer ist als der letzte Scan, und diesen zu übernehmen. Dies bedeutet, dass eine Reihe von Kommentaren vorhanden ist, die noch nicht analysiert wurden. Der Grund für die Wahl des ältesten Repositorys ist, dass es wahrscheinlich mehr Kommentare als andere Repositories hat. Mit dieser Vorgehensweise wird die Abdeckung der Programme erhöht. Nachdem nun das Repository ausgewählt wurde, wird der Prozess entsprechend der in der Abbildung 5.10 gezeigten und nachfolgend beschriebenen Schritte fortgesetzt.

Zunächst wird eine Liste aller Issues in einem bestimmten Zeitraum erstellt. Der Zeitraum beläuft sich von der letzten Analyse bis heute. Dieser Zeitfilter intern 4 ist erforderlich, weil die Issues, die bereits analysiert wurden, nicht erneut analysiert werden. Dementsprechend gibt es eine umfangreiche Liste von Issues und für jedes Issue, zu dem eine bestimmte Anzahl von Kommentaren gespeichert worden sind, werden diese Kommentare auch heruntergeladen. Dann wird mit dem Schreiben begonnen, indem die erste Zeile des Issues und die nächsten Zeilen der Kommentare zu diesem Issue geschrieben werden. Danach wird zum nächsten Thema übergegangen und es werden auf die gleiche Weise die erste Zeile des Issues und die nächsten Zeilen der Kommentare zu diesem Issue geschrieben. Aufgrund dieses Verfahrens entsteht eine sehr umfangreiche Datei mit Issues und Kommentaren, die im Anschluss an das Programm Senti4SD abgegeben werden muss, was folgende zwei Probleme beinhaltet:

1. Senti4SD hat Probleme mit Zeichen, die im Unicode codiert sind und warnt demzufolge vor ihnen. Um diesem Problem entgegen zu wirken, wurden beim Hinzufügen einer Datei mit Kommentare und Issues, die solche Unicode Zeichen enthalten, diese nicht zur Analyse weitergeleitet.

2. Das zweite Problem basierte auf einer zu großen Anzahl an Sätzen in einer Datei. Das Programm Senti4SD konnte diese nicht analysieren und stürzte ab . Um dieses Problem zu lösen, wurden Kategorien/Teildatein mit

jeweils tausend Kommentaren gebildet und jeweils eine Kategorie wurde in eine Datei eingefügt. Zehntausend Kommentare z.B. ergaben zehn Dateien, und jede dieser Dateien wurde an Senti4SD zur Analyse gesendet und einzeln analysiert, sodass es zu keinem Absturz des Programms kam.

Wie in Kapitel 2.4.5 erwähnt wurde, ist Senti4SD eine mit Java und R-Programmen geschriebene Software, die in GitHub [6] verfügbar ist und auf drei Schritten aufgbeaut ist (s. Abb. 5.10). Der erste Schritt besteht darin, die Eingabedatei zu erstellen. Der zweite Schritt besteht darin, einen Child-Process[7] zu erstellen, in dem der Senti4SD-Programmprozess mit der angeforderten Eingabe ausgeführt wird. Anschließend muss gewartet werden, bis der Child-Process das Programm beendet hat und zum Main-Process[8] zurückkehrt. Auf diese Weise wird sichergestellt, dass die Ausgabedatei korrekt ist. Wenn sie nicht korrekt ist, bedeutet es, dass sie schwer auszuführen war. Im dritten Schritt wird die Ausgabedatei analysiert und in der Datenbank gespeichert. Diese drei Schritte wiederholen sich immer wieder.

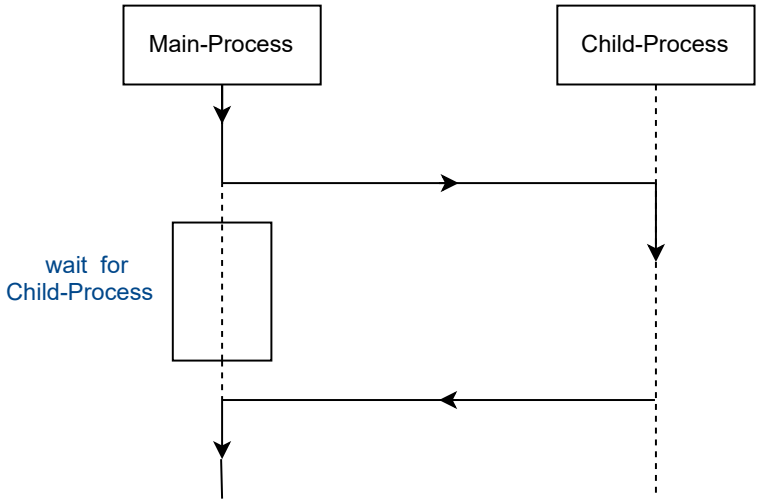


Abbildung 5.9: Übersicht des Child-Processes

Die Komponente, die mit Senti4SD zusammenhängt, nimmt eine Eingabe und eine Ausgabe vor. Die Eingabe ist eine CSV-Datei, die einen Kommentar in jeder zu analysierenden Zeile hat. An dieser Stelle ist wichtig und zu beachten, dass viele Kommentare mehrzeilig sind. D.h. Beim Hinzufügen werden alle neuen Zeilen durch ein Leerzeichen ersetzt, so dass der gesamte Kommentar in einer langen Zeile steht, die analysiert werden soll.

[6] https://github.com/collab-uniba/Senti4SD
[7] Kindprozess
[8] Hauptprozess

Die Ausgabe, die Senti4SD liefert, ist eine CSV-Datei mit mehreren Spalten. In der ersten Spalte steht T plus eine Zahl(z.b T10), dann ein Komma, und dann wird die Stimmung (positiv, negativ, neutral) geschrieben. Diese Spalte ist die Zeilennummer der Eingabedatei. Z.B drückt T10 die Stimmung der zehnten Zeile aus. Da sich jede Zeile bzw. deren Kommentare voneinander unterscheidet. Und einige Antworten früher und einige später eintreffen, sind sie nicht in der richtigen Reihenfolge. Damit können diese mit dem richtigen Kommentar verknüpft werden, wird bei dem Issue der Kommentare in der Eingabedatei eine Liste erstellt. Wenn die Ausgabe wie beschrieben erfolgt, sollte nur die erste Spalte übernommen werden. Dadurch wird die Input-Output-Verbindung gelöscht. Nach Beendigung dieses Vorgangs wird später zum Repository gewechselt, wo dieser Prozess wiederholt wird.

Im Falle von Senti4SD ist zu beachten, dass, wenn das Programm kein Repository zur Analyse findet, es in den Schlaf-Modus (Sleep) geht, während das Programm zwei Stunden lang läuft. Das ist deshalb so geplant, weil es in der Zukunft, wenn alle Repositories von Senti4SD analysiert werden, kein neuer Fall mehr ist, und da dieses Programm in eine Endlosschleife fällt, befindet es sich zwei Stunden lang im Schlaf-Modus.

Eine Besonderheit des Sleep-Modus ist, dass dessen Dauer in 5-Sekunden-Segmente unterteilt ist. D.h. Es schläft für 5 Sekunden, steht dann auf und überprüft die Flagge für die Beendigung des Signals (Signalübersteuerung: engl.signal override). Wenn die Überprüfung beendet ist, muss es für 5 Sekunden schlafen und für weitere 5 Sekunden Aufwach bleiben. Dieser Prozess ist so konzipiert, wenn das Programm in Sleep geht und traf „ctrl+c“ muss auf die geplante Zeit warten.

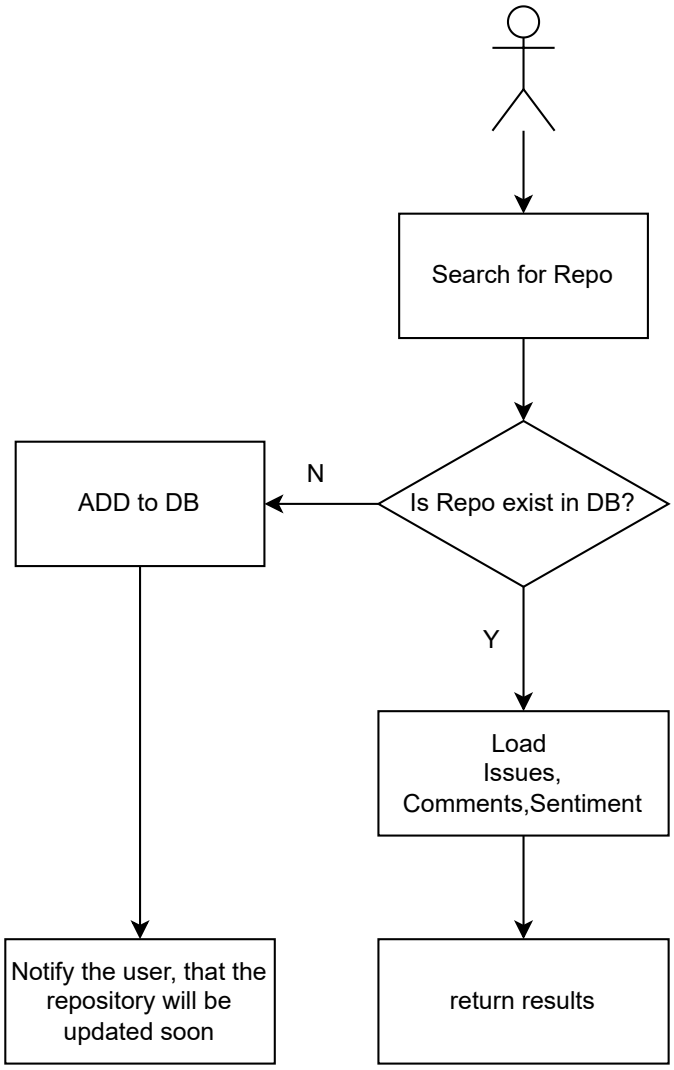


Abbildung 5.10: Übersicht der Datenanalyse

Diese Abbildung fasst den Prozess zusammen, wie der Prozess zum Visualisieren gelangt. Dies ist ein logischer Teil des Servers, der entweder auf die Website oder auf das Plugin reagiert. Der Benutzer sucht nach einem Repository, um zu sehen, ob es existiert. Wenn es nicht in der Datenbank vorhanden ist, wird es hinzugefügt und gespeichert und seine Daten werden heruntergeladen und bald analysiert. Und wenn es nicht existiert, werden die Issues, Kommentare und ihre Stimmungsanalyse geladen und dem Benutzer angezeigt. Auf diese Weise wird die Datenbank regelmäßig aktualisiert.

5.6 Visualisierung

Wie bereits in den vorangegangenen Kapiteln erwähnt, kann es sinnvoll sein die extrahierten und analysierten Daten in zwei Varianten darzustellen. Eine kompakte Ansicht in einem Plugin auf der GitHub Seite und eine ausführlichere Visualisierung auf einer eigenen Webseite. Diese Entwicklung wurde davon abgeleitet, dass die an das Konzept gestellten Anforderungen eine kompakte, als auch eine ausführliche Darstellung der Stimmung vorsehen.

Es wurden bei der Entwicklung diese zwei Teile vorgesehen, da der Aufbau des Systems aus mehreren Komponenten besteht, die aufeinander aufbauen, und dementsprechend sichergestellt wird, dass jede Komponente korrekt funktioniert und in mehr als einer Seite hergestellt wird.

Die Darstellung der Stimmungsanalyse wird mittels verschiedener Diagramme auf der Webseite ermöglicht. Diese stellen die Stimmung, die zuvor in die Kategorien positive, negative, neutral dar. Zusätzlich zu den in Kapitel 4 ausgeführten Diagrammen ist es sinnvoll, die Verteilung der Kategorien der Polaritäten zu visualisieren. Dies ist deshalb sinnvoll, weil beispielsweise ein Übermaß an neutral gekennzeichneten Kommentaren in weiteren Analysen zu einer Verzerrung des Emotionalitäts-Bewertung(Stimmungswert) führen könnte. Zu diesem Zweck werden die verschiedenen Werte der Kategorie aggregiert und als ein Diagramm auf der Webseite angezeigt.

5.6.1 Flask-Webinterface

Für diese Arbeit wurden die beiden Web-Frameworks Flask und Django in Betracht gezogen. Obwohl beide Systeme Vor- und Nachteile haben, wurde Flask ausgewählt. Im Vordergrund stand, wie bereits erläutert, der zu vermeidende Overhead. Eine Django-Anwendung bringt bereits deutlich mehr Funktionalität mit, die für den hier beabsichtigten Zweck nicht notwendig ist und somit einen Overhead darstellt. Darüber hinaus ist eine Flask-Anwendung bereits mit sehr wenig geschriebenem Quellcode lauffähig und bietet durch einen modularen Aufbau und viele verfügbare Bibliotheken die Möglichkeit, in Zukunft ohne große Hindernisse erweitert zu werden.

Flask bietet einen Dienst, der sowohl Plugin als auch Website ist. D.h. Flask wird als Webinterface betrachtet. Deswegen erhält der Webserver eine Anfrage von der Datenbank und empfängt sowohl das Plugin als auch die Website. Diese beiden Teile sind so konzipiert, dass der Benutzer Zugang hat. Hier müssen der Webserver und Apache mit Flask verbunden sein, und

es werden Anfragen aus dem Internet an Apache gesendet, und Apache überträgt sie an Flask. Flask antwortet auf verschiedene Anfragen, die zwei Teile umfassen, und zwar einen Teil, um Seiten anzuzeigen und die Seite zu rendern, und ein API, das das Plugin anfordern und eine Antwort erhalten.

5.6.2 Plugin

Auf der Grundlage des in Abschnitt 4.5 entwickelten Konzepts wird auf Basis der Visualisierung, die von der Zielvorstellung über die Benutzer-GitHub bis zum System reicht, das Plugin vorgenommen, das in das zu entwerfende Konzept einzubetten ist. Die Grundstruktur wurde gegenüber dem entwickelten Konzept beibehalten, um die dort beschriebene Funktionalität zu erhalten.

Mit diesem Hintergrund startet das Plugin eine Ajax-Anfrage bzw. XMLHttpRequest() und die Anfrage wird in JavaScript angekündigt (wie ein Browser mit einem geöffneten Tab und einem eingefügten Link). Es verwendet einen Json und gibt die erforderlichen Informationen in diesen Json ein. Während der letzte Teil komplett in Python-flask geschrieben sind, wird in der Darstellungsebene als Programmiersprache auf JavaScript zusammen mit HTML und CSS zurückgegriffen. Dann kann auf Github eingegeben werden und anschließend kann der Benutzer zuerst manifest.json in **about:debugging** für Firefox [9] eingeben und es dann auf der GitHub-Seite sehen. Für die Umsetzung wurde auf das JavaScript in Verbindung mit JSON(manifest.json) zurückgegriffen. Neben diesen Kategorien lässt sich zusätzlich der Knopf „show more“ ganz unten rechts finden (s. Abb 5.11). Mit einem Klick auf diesen werden der der Kategorie zugehörende Text und Zahlen, die den Benutzer direkt auf die Webseite der besuchten Internetseite führt, angezeigt (s. Abb 5.14).

[9] about:debugging#setup

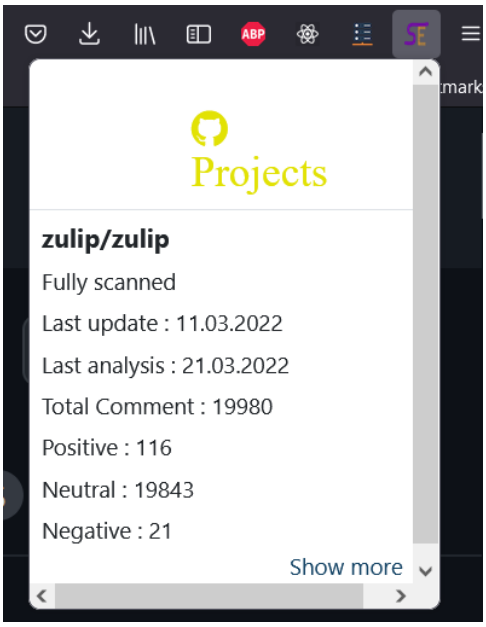


Abbildung 5.11: Ansicht des Plugins

Wenn der Webbrowser auftaucht, nimmt er den Namen der URL und von dieser URL den Namen des Repositorys und seines Besitzers. Dann erfolgt mit XHR die gleiche Anfrage wie oben: Er stellt eine Anfrage an den Server, nimmt die Antwort und zeigt die Antwort. Dieser Vorgang dauert zwischen fünfzehn und zwanzig Sekunden, weil die Datenbank den Namen des Repositorys suchen und die RepoID finden muss. Über diese RepoID erhält sie alle Issues. Alle Issues haben einen Analysewert neben sich (positiv, negativ und neutral), und dann holt sich das Plugin seine Kommentare für jede issueID, und jeder Kommentar bringt auch einen Analysewert (s.Abb. 5.11). D. h., dass die Anzahl der zu stellenden Abfragen groß ist, ist darauf zurückzuführen, dass die Abfrage aus drei Teilen besteht. Die erste Abfrage dient dazu, eine repoID zu erhalten. Die zweite fragt nach der Liste der Issues. Die dritte fordert die einzelnen Kommentare für die Anzahl der Issues an.

5.6.3 Webseite

Die Website ist so gestaltet, dass sie auch unabhängig vom Plugin aufgerufen werden kann[10]. Das Plugin greift mit einem Befehl und einer Schleife auf die Website zu. Auf der Startseite der Webseite kann nach dem Namen des Repository gesucht und daneben auch das offene oder geschlossene Modell der Issues ausgewäht werden. Dann wird mit einem Klick die nächste Seite aufgerufen (s. Abb. 5.12), um weitere Details zu sehen. Danach ist

[10]http://camus.se.uni-hannover.de

ein Linien-Diagramm von der Stimmungsanalyse zu sehen (s. Abb. 5.15). Mittels dieses Diagramms kann die Polarität der Stimmungsanalyse erkannt werden. Zeitreihen sind eine Möglichkeit diese Daten darzustellen, indem die Daten auf einer Zeitachse abgebildet werden.

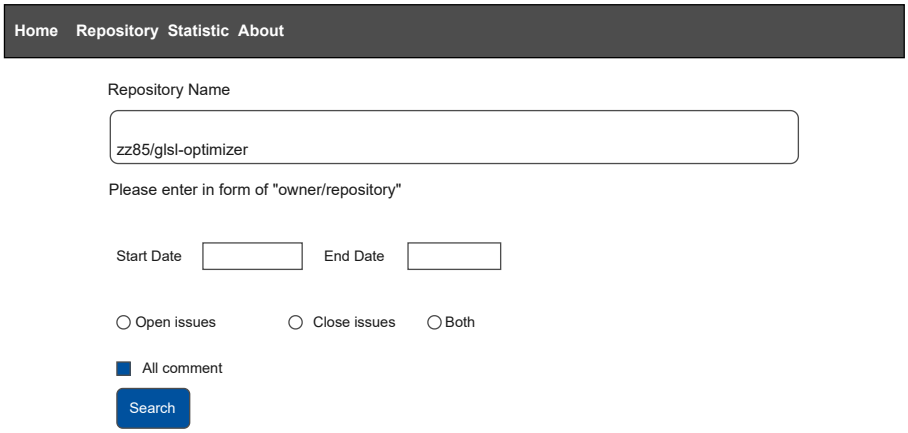


Abbildung 5.12: Ansicht der Startseite von der Webseite

Zusätzlich gibt es die Seite „Database Statistic“, die dem Benutzer allgemeine Informationen über die Anzahl der Repositories und die Gesamtzahl der Issues und Kommentare liefert (s. Abb. 5.13[11]) [12].

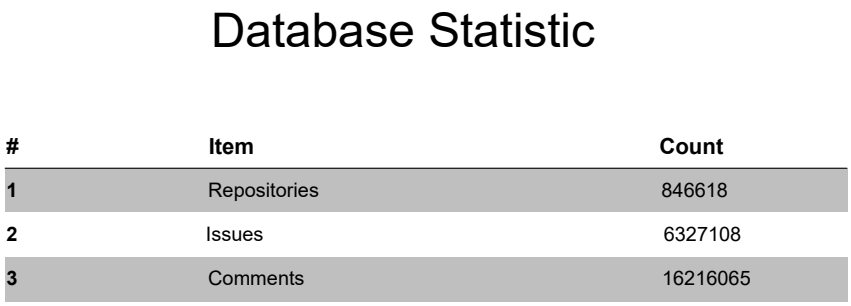


Abbildung 5.13: Ansicht der Database-Statistic-Seite

Verlauf des Stimmungsscores im Laufe der Zeit

Dieser Teil der Seite befasst sich mit der Stimmungsbewertung, die anhand der extrahierten und gespeicherten Daten, die der Startseite

[11] Stand: 16.04.2022
[12] http://camus.se.uni-hannover.de/statistic

entnommen wurden, berechnet wird. Einer der wichtigsten Punkte für die Bewertung der Stimmung innerhalb eines Projekts ist die Entwicklung des Emotionalitätswerts im Laufe der Zeit. Aus diesem Grund wird auch ein Zeitfilter berücksichtigt, so dass die gewünschte Zeit auf der Home-Seite eingegeben werden kann und auf der nächsten Seite entsprechend dem gewählten Filter zu sehen ist. Dieses ermöglicht dem Benutzer, die Analyse zu einem beliebigen Zeitpunkt zu haben (s. Abb. 5.12 und 5.14).

Die erste Eingabe des Screenshots ist der Name des Repositorys. In der Mitte befindet sich die gesamte Zahlen des Emotionalitätswerts und es wird angezeigt, ob der Prozess total gescannt(engl.:fully scanned) wurde. D.h. Wenn das Datum Jahr 2000 angezeigt wird, bedeutet dies, dass dieses Repository in der Datenbankliste enthalten ist, aber noch nicht aktualisiert oder analysiert wurde, für beide Zustände wird „Not fully scanned“ angezeigt. Aber wenn bei dem Repository beide schön aufgeführt wurde, wird „fully scanned“ angezeigt. Schließlich ist letzte Eingabe zum Zeitpunkt ein Diagramm. Es wird automatisch die Stimmung für die Eingaben in den Graphen angeigt(X-Achse Zeit, Y-Achse Stimmung), welches die verschiedenen aktivierten Stimmungsanalysen anzeigt [13].

Funktion „all comments“ und „Start- End Date“ bei den Filters:

Auf der Webseite ist zu sehen, dass mit Hilfe des Zeitfilters zu einem bestimmten Zeitpunkt nur die Ergebnisse von Stimmungsanalyse in diesem Zeitraums angezeigt werden. Und es kann dem Benutzer mit Hilfe Filter „all comments“ auch die Möglichkeit geben, die Ergebnisse der Stimmungsanalyse der Kommentare von entsprechenden Issues, die in diesem Zeitraum in GitHub geschrieben wurden, zu zeigen. Exemplarisch ist dies bei dem Repository zulip/zulip zu sehen (s. Abb. 5.15). Hier ist „Start Date“: 05.09.2015 und „End Date“: 05.12.2016 .

[13] http://camus.se.uni-hannover.de

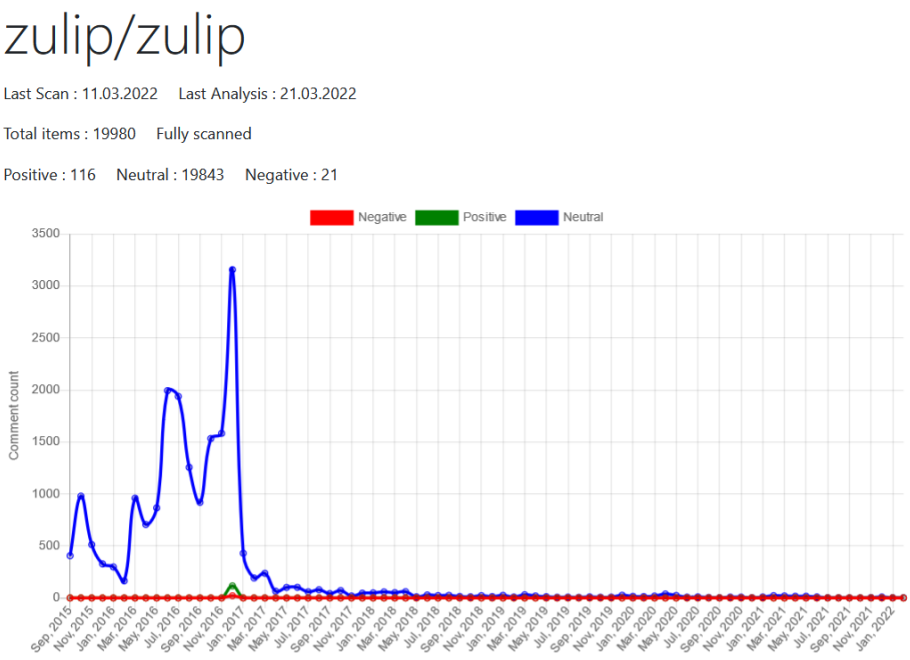


Abbildung 5.14: Linien-Diagramm für das „zulip“ Repository mit zahlreichen Emotionalitätswerten

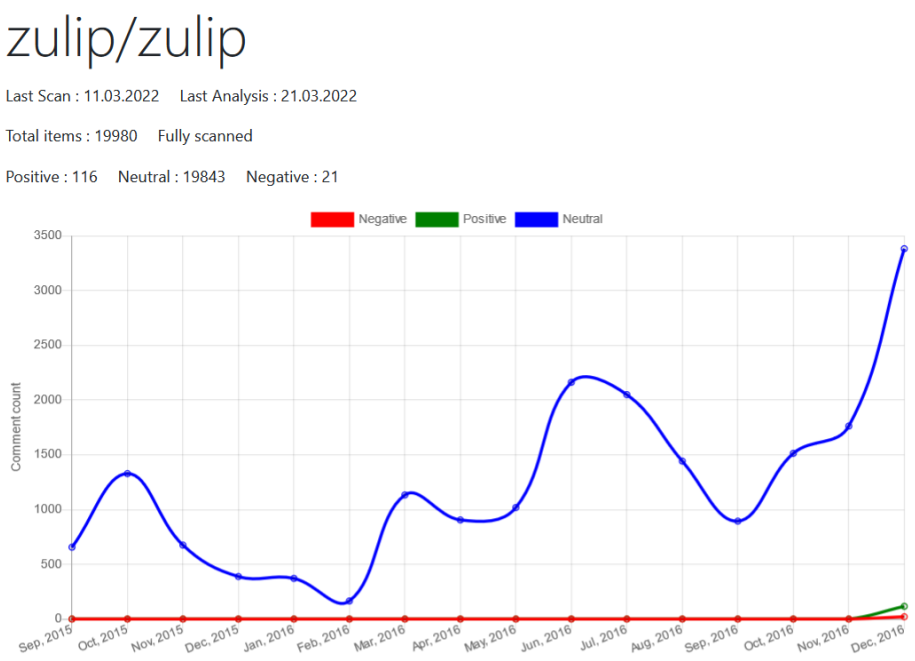


Abbildung 5.15: Linien-Diagramm basierend auf Emotionalitätswert und Zeitfilter

Kapitel 6

Evaluation und Ergebnisse

Um das im vorangegangenen Kapitel vorgestellte und implementierte Plugin und die Website zur Sentimentanalyse möglichst praxisnah zu evaluieren, wurden sie im Rahmen einer Nutzerstudie eingesetzt. Nachfolgend wird zunächst kurz die Zielsetzung der Evaluation erläutert. Im Anschluss daran werden die Evaluationsumgebung und damit im Zusammenhang stehend die Forschungsfragen (engl.: Research Questions (RQs)) beschrieben, die mit Hilfe dieser Studie beantwortet werden sollen. Dann wird die Methodik ausgeführt. Es werden daran anschließend die Ergebnisse der Studie vorgestellt sowie interpretiert.

6.1 Zielsetzung

Das Ziel der Evaluation ist zu ermitteln, ob das erstellte Konzept zur Stimmungsanalyse Entwicklern einen besseren Einblick in Open-Source-Projekte ermöglicht. Dazu wurde eine qualitative Studie durchgeführt, in der das entwickelte Tool bestehend aus Webseite und Plugin von Probanden in der Praxis genutzt wurde.

6.2 Evaluationsumgebung

Zunächst wurde den an der Studie Teilnehmenden das Plugin, die Webseite sowie die durchzuführenden Aufgaben vorgestellt. Anschließend haben sie einen Fragebogen ausgefüllt. Während der Evaluierung wurden einige Notizen über die von den Teilnehmern durchgeführten Schritte und ihre Kommentare gemacht. Um die Nutzerstudie strukturiert durchführen zu können, erfolgte deren Vorbereitung nach dem systematischen Ansatz der Fragen und Antworten in Bezug auf Forschungsfragen. Die RQs sollen dabei zur Erreichung des Ziels der Forschungsfragen beitragen. Um die Arbeit zu bewerten, wurde geprüft, ob alle Komponenten bearbeitet wurden und wie die Ergebnisse lauteten.

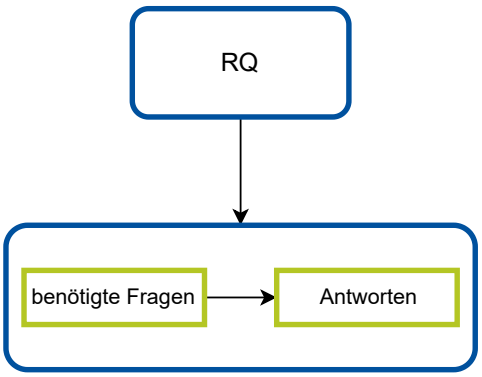


Abbildung 6.1: Übersicht der Forschungsfragen

Abbildung 6.1 zeigt, dass die drei RQs als übergeordnete Fragen den Teilnehmer im Rahmen der Studie gestellt wurden. Die Fragen, die während der Studie gestellt wurden, wurden auch mit ihren Antworten in Betracht gezogen.

RQ1

Gibt es bei den Nutzern einen Bedarf an einer Stimmungsanalye für GitHub-Projekte?

Wenn sich ein Entwickler ein Open-Source-Projekt ansieht, weiß er jedoch noch nicht, ob es sich lohnt, daran mitzuarbeiten. Die Stimmungsanalyse von Open-Source-Projekten soll dazu beitragen, dass der Entwickler weitere Einblicke in das Projekt erhält. Die Sentimentanalyse ermöglicht es, eine hinreichend genaue Stimmungsanalyse von Sätzen zu gewährleisten.

RQ2

Unterstützt diese Firefox-Plugin-Erweiterung die automatische Stimmungsanalyse von GitHub-Open-Source-Projekten?

Wenn Interesse an einem Open-Source Projekt und der Mitarbeite daran besteht, ist ein Plugin mit Sicherheit von Vorteil. Es lassen sich Projekte identifizieren, in denen eine eher positive Stimmung herrscht. Hier würde der Entwickler eher mitarbeiten als in einem Projekt mit schlechter Stimmung. Ein praktikables Plugin zur Erkennung von Stimmungen in Kommentaren kann dabei helfen. Es ist wichtig, dass die GitHub-URL automatisch erkannt wird. Wie bei allen Plugins ist es wichtig, dass sich dieses Plugin nur in der definierten URL öffnet. Befindet sich der Benutzer auf einer anderen Website, öffnet sich das Plugin-Fenster nicht und es wird angezeigt, dass diese Website nicht die gewünschte Website ist.

RQ3

Unterstützt die Webseite die Nutzer, besondere Funktionen und Filter nach der Stimmung von GitHub-Projekten zu identifzieren?

Der Vergleich der Open-Source-Projekte unter Zuhilfenahme des Linien-Diagramms auf der Webseite machte deutlich, dass die Notwendigkeit von Filtern nachvollzogen werden kann. Dies wäre hilfreich, wenn die Entwickler die Stimmung der Kommentare in einem bestimmten Zeitraum wissen wollen. Außerdem sollten mit Hilfe einer Umfrage herausgefunden werden, ob die Webseite, die nur positive oder nur negative Kommentare anzeigt, gut ankommt.

Darüber hinaus wurden den Teilnehmern im Rahmen der Studie einige Fragen zu den RQs gestellt, um deren Einstellung dazu zu erfassen.

6.3 Durchführung

Für diese Studie wurde ein synchroner Remote-Usability-Test (sRUT: synchronous distance usability test) durchgeführt. Das bedeutet, dass die Teilnehmer nicht, wie sonst üblich, in einem Labor sitzen, sondern über das Internet mit Hilfe eines Konferenzsystems an der Studie teilnehmen.

6.3.1 Technische Umsetzung

Die folgenden Programme wurden im Rahmen der Studie verwendet und stellen somit die technischen Voraussetzungen dar, die erfüllt sein mussten, um an der Studie teilnehmen zu können. Die erste Voraussetzung für die Teilnahme an der Studie war eine schnelle und stabile Internetverbindung.

Skype

Skype[1] und BigBlueButton[2] sind Conferencing-Systeme. Sie wurden in dieser Studie verwendet, um die Kommunikation zwischen den Studienteilnehmern und der Studienleiterin zu ermöglichen.

Chrome Remotedesktop Extension

„Chrome Remotedesktop Extension“[3] und „AnyDesk“[4] sind Programme für den Fernzugriff auf Computer. Es sei erwähnt, dass das im Rahmen dieser

[1] https://www.skype.com/de/
[2] https://bigbluebutton.org/
[3] https://remotedesktop.google.com/
[4] https://anydesk.com/de

Arbeit entwickelte Plugin, wie auch Remotedesktop, eines für Chrome ist. Diese Programme wurden verwendet, um den Teilnehmern den Zugang zu dem Computer zu ermöglichen, an dem die Studie durchgeführt wurde.

OBS bzw. „Windowstaste“+ „G“

Um die Teilnehmer während der Durchführung der Studie aufnehmen zu können, wurden die Programme OBS[5] und „Windowstaste“+ „G“ genutzt, die Videoaufnahmen des Bildschirms ermöglichen.

6.4 Testaufbau

Dabei wurden ihnen zuerst das Konzept der Arbeit vorgestellt. Anschließend arbeiteten die Testpersonen selbst mit dem Plugin und der Website. Diese Phase basierte auf dem Think-Aloud-Prinzip, d.h. die Teilnehmer äußerten ihre Gedanken laut und diese wurden aufgezeichnet. Das Format der offenen Fragen sollte den Prozess der möglichst frühzeitigen Einbindung von Feedback in den Entwicklungs- und Verbesserungsprozess der Software unterstützen. Die Dauer pro Teilnehmer betrug je nach Gespräch etwa 20 bis 40 Minuten.

6.5 Ergebnisse der Studie

Im Allgemeinen wurden das Konzept und das Plugin sehr positiv aufgenommen. Die Mehrheit der Teilnehmer äußerte sich positiv darüber, dass die Übersicht über die Stimmungsanalyse von Kommentaren ein wichtiger Baustein sein kann, um die Stimmung in einem Repository gezielter zu erfassen und im Falle einer Verschlechterung präventive Maßnahmen zu ergreifen. Die folgenden Ergebnisse basieren auf den Gesprächen mit den Teilnehmern und der Umfrage. Wie bereits erwähnt, wurden den Teilnehmer in dieser Studie offene und geschlossene Fragen gestellt. Für die geschlossenen Fragen gab es eine einheitliche Skala von „keine Zustimmung“/„nie“ bis „volle Zustimmung“/„immer“. Die offenen Fragen wurden in zwei Phasen eingeteilt. In der ersten Phase wurden die Antworten der Teilnehmer manuell kategorisiert. Anschließend wurden die Ergebnisse zusammengetragen.

Dieser Prozess in der Umfrage wurde in der zweiten Phase wiederholt, um eine genaue Zuordnung der Antworten in Kategorien zu ermöglichen. In der ersten Phase wurden die Antworten zu allgemeinen Fragen gliedert. In der zweiten Phase wurden die Antworten in die Kategorien des entworfenen

[5] https://obsproject.com/de

Plugins sowie der entworfenen Webseite eingeordnet.

Die Studie wurde mit insgesamt 31 Probanden durchgeführt, die Informatik, Elektro- und Informationstechnik und Mechatronik sowie Machinenbau studieren oder in diesem Bereich arbeiten. Im Durchschnitt waren 50% von den Proband:innen tätig. Dabei handelte es sich ausschließlich um Personen, die sich mit GitHub-Projekten beschäftigen und die über grundlegende Englischkenntnisse verfügen, sodass sie dementsprechend in der Lage waren, ein fundiertes Feedback zu dem Gesamtkonzept geben zu können. Die Probanden waren zwischen 24 und 49 Jahre alt. 15 (50%) der Teilnehmer gaben an, Studierende zu sein, 13 (43.33%) von ihnen arbeiten im Bereich Informatik und Elektrotechnik, zwei (6.67%) sind Auszubildende. Zehn (33.33%) Teilnehmer verwenden mehrmals in der Woche GitHub. 18 (60.00%) stimmten der Aussage „Mir ist für das Firefox-Plugin auf GitHub zur Stimmungsanalyse wichtig.“ eher zu. Dies steht im Kontrast zu den Angaben der Teilnehmer, wie oft sie darauf achten, ob eine Internetseite wie GitHub ein Plugin besitzt. Hier gaben zehn (33.33%) der Probanden an, darauf oft zu achten. Weiterhin führten nur zehn (33.33%) Teilnehmer an, ein Stimmungsanalyse-Tool für GitHub gelegentlich zu brauchen. Fünf (16.67%) Probanden gaben an, dies Firefox-Plugin immer zu nutzen und 16 (53.33%) täten dies gelegentlich. Als Grund für die Nutzung des Firefox-Plugins auf GitHub zur Stimmungsanalyse wurde häufig von den Studienteilnehmer:innen angeführt, dass sie vor allem für Programmierzwecke regelmäßig mit GitHub interagieren und daher eine Einschätzung der Stimmung in den Repositories sehr hilfreich sei.

6.6 Beantwortung der Forschungsfragen

In diesem Abschnitt werden die für die Beantwortung von RQ1, RQ2 und RQ3 relevanten Resultate dargestellt. In Bezug auf jede RQ wurde einige Fragen, die von den Entwicklern während einer Umfrage und einer Videokonferenz beantwortet wurden, gestellt. Nach dem Überprüfen des Prozesses wurden in einem kurzen Gespräch Anregungen, Ideen und Verbesserungsvorschläge für das Plugin und die Website ausgetauscht. Dann wurde den Probanden der Evaluationsbogen mittels eines Links zur Verfügung gestellt, der die Probanden durch die Umfrage führte.

Wenn ein Benutzer nicht nur an allen Bestandteilen in diesem Firefox-Plugin interessiert ist, sondern auch die Stimmung des Repositorys herausfinden möchte, kann ihn diese Aufgabe vor eine Herausforderung stellen. Viele Plugins bieten kein Inhaltsverzeichnis für ein Diagramm, das es dem Nutzer ermöglicht, direkt zu dem Punkt zu springen, der ihn interessiert, z.B. „Show more“ für das Zeigen des Linien-Diagramms.

Beantwortung der Forschungsfrage RQ1

Im Rahmen der Frage RQ1 wurden folgende Subfragen während der Umfrage gestellt:

Haben Sie schon einmal darüber nachgedacht, ein Stimmungsanalyse-Tool für GitHub-Projekte, deren Kommunikationsdaten es beinhaltet, zu nutzen?

Nur 32.26% der Studienteilnehmer:innen haben gelegentlich darüber nachgedacht (s. Abb. 6.2). Für sie spielt das Analysieren der Kommunikationsdaten aus der GitHub-API eine Rolle und die Einbindung bereits bestehender Tools wurde angedacht. Die zentrale Funktion der Analyse wurde dem Plugin hinzugefügt, damit der Benutzer sie auf der Plugin-Seite sehen kann. Zu diesen Funktionen wurden die Teilnehmer befragt und das Ergebnis muss berücksichtigt werden.

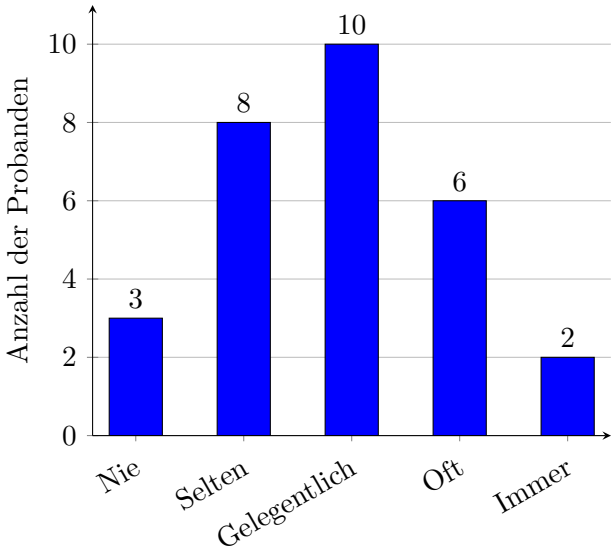


Abbildung 6.2: Nachdenken über ein Stimmungsanalyse-Tool für GitHub

Beantwortung der Forschungsfrage RQ2

Im Rahmen der Frage RQ2 wurden folgende Subfragen während der Umfrage gestellt:

1.Achten Sie generell darauf, ob Plattformen wie GitHub ein Plugin haben?

Da zu diesem Zweck ein Plugin entwickelt wurde, war es wichtig zu wissen, was die Teilnehmer darüber dachten. Insgesamt haben 64.52% bzw. 20 Personen oft und gelegentlich darauf geachtet, d.h. das Plugin ist für sie interessant (s. Abb. 6.3).

2. Ist den Probanden das Firefox-Plugin auf GitHub zur Stimmungsanalyse wichtig?

67.74% bzw. 21 der Studienteilnehmer gaben an, dass es für sie wichtig ist, in GitHub ein Plugin zur Stimmungsanalyse zur Verfügung zu haben. Diese 21 Teilnehmer gaben an, dass sie anhand der Stimmungsdaten Begriffe besser finden können, wenn sie die Gesamtzahl der positiven Kommentare sowie der negativen Kommentare sehen. Ein weiterer genannter Grund war, dass es in der Literaturübersicht für sie zum Beispiel wichtig sei, wie oft ein Repository zitiert wird und das mit einer Übersicht, wie viel positives und negatives Feedback in den Kommentaren von den Issues eines Repositorys enthalten ist. Als dritter Grund wurde angeführt, dass die Websites immer größer und komplizierter werden und es damit immer schwieriger wird, das relevante Material zu finden. Unter diesen Umständen können einige Hilfsanwendungen oder Plugins den Menschen Zeit sparen und ihnen helfen, das zu finden, was sie suchen.

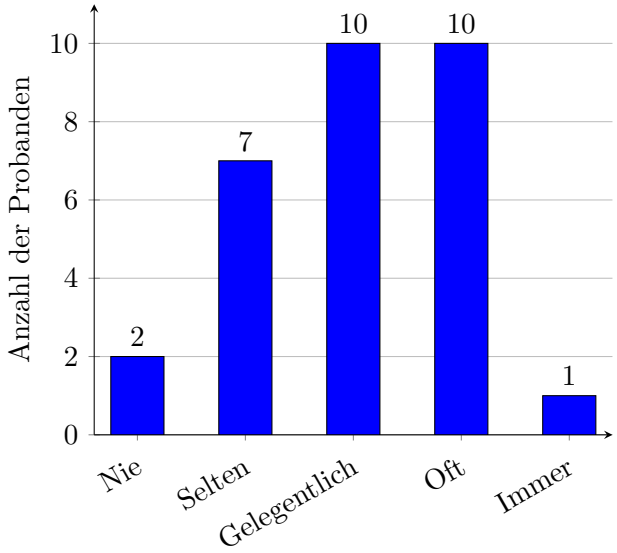


Abbildung 6.3: Achten darauf, ob die Plattformen ein Plugin haben?

Beantwortung der Forschungsfrage RQ3

Um die Forschungsfrage RQ3 untersuchen zu können sind folgende Gesichtspunkte als Teilfragen in den weiteren Überlegungen zu berücksichtigen: Kann der Aufbau der Webseite für gewünschte Funktionen

für den Entwickler hilfreich sein?

Unterstützen das Plugin und die Webseite den Nutzer bei dem Verständnis der Stimmungsanalyse und dem Filtern von einem Repository. Daraus ergibt sich, dass 28 Personen glaubten (s. Abb. 6.4), dass dieser Prozess sehr nützlich war, weil sowohl Entwickler als auch Unternehmen, die ein GitHub-Repository haben, diese Funktion brauchen und es manchmal hilfreich ist, die Emotionen hinter den Kommentaren und Problemen, mit denen ihr Kollegen oder Mitarbeiter konfrontiert sind, zu verfolgen und zu analysieren. Diesbezüglich wurde gefragt, ob ein Zeitfilter notwendig ist. Die meisten Teilnehmer gaben an, dass Filter wie die Zeit betreffend eine interessante Idee sein kann, den ein Entwickler verwenden kann, wenn er die Stimmungsanalyse der Kommentare in einem bestimmten Zeitraum sehen möchte.

Dann wurden sie gefragt, ob die Inhalte der Webseite und des Plugins für sie leicht zu verstehen waren und ob sie es in Zukunft verwenden wollen. Mehr als 75% waren damit zufrieden und die meisten sagten, dass sie es in Zukunft benutzen möchten.

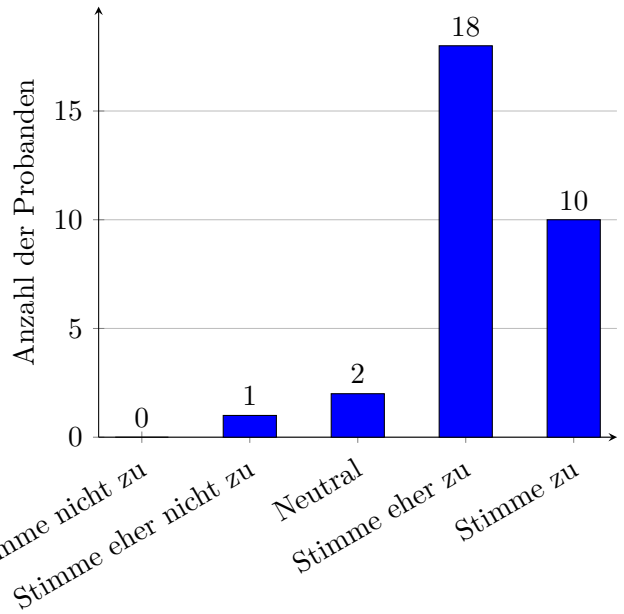


Abbildung 6.4: Meinung der Probanden bzgl. Plugin und Webseite

Kapitel 7

Diskussion

In diesem Kapitel wird zuerst Zusammenfassung der Ergebnisse erwähnt. Anschließend wird die Limitationen erläutert. Abschließend werden Empfehlungen für die Weiterentwicklung des Plugins und der Webseite gegeben.

7.1 Zusammenfassung der Ergebnisse

Aus der Erläuterung der Ergebnisse der Studie (s. Abschnitt 6.5) geht allgemein hervor, dass Sentiment-Analyse auf großes Interesse bei den Befragten stößt. Die gesammelten Ergebnisse deuten darauf hin, dass eine Stimmungsanalyse-Tool in Kommentare nicht nur technisch machbar ist, sondern auch von Praktikern nachgefragt wird. Für viele ist die Optimierung des Plugins in Entwicklung ein wichtiger Aspekt für zukünftige Entwicklungsprozesse. Die zunehmende Komplexität von Projekten stellt neue Herausforderungen an die Entwickler und ihre Zusammenarbeit. Dies ist ein wichtiger Schritt zur Stärkung und Weiterentwicklung des Plugins zur Sentimentanalyse. Das entwickelte Programm fügt sich nahtlos in den Arbeitsablauf der Nutzer ein und wurde von diesen sehr positiv aufgenommen. Viele der Teilnehmer wünschten sich eine Veröffentlichung des Plugins, um es in ihren Browser integrieren zu können.

7.2 Limitationen

Die Ergebnisse dieser Arbeit sind durch interne als auch externe Voraussetzungen in ihrer Aussagekraft begrenzt. Es wird darauf eingegangen, welche Aspekte möglicherweise die Ergebnisse einschränken. Dabei sind drei äußere Rahmenbedingungen zu erwähnen, die die Aussagekraft der Ergebnisse dieser Arbeit begrenzen.
An der Studie nahmen nur 31 Personen teil. Daher sind die Ergebnisse der Studie möglicherweise nicht repräsentativ. Die Aufgaben und Fragen, die den Probanden gestellt wurden, stammen von der Verfasserin dieser Arbeit.

Obwohl vor Beginn der Studie ein Pilotprojekt mit dem Betreuer dieser Arbeit durchgeführt wurde, kann nicht garantiert werden, dass alle Fragen objektiv gestellt wurden.

Außerdem hatten die Teilnehmer nicht alle die gleichen Vorkenntnisse. Das Tool wurde auf englisch evaluiert, aber einige hatten Probleme mit der englischen Sprache. Andere hatten noch nie mit einem Plugin gearbeitet. Eine weitere Einschränkung der durchgeführten Studie ist der begrenzte Rahmen, in dem sie durchgeführt wurde. Nutzer besuchen jeden Tag viele verschiedene Websites, aber im Rahmen des Experiments konnte nur GitHub auf Firefox betrachtet werden.

Aufgrund der COVID-19-Pandemie wurden die Experimente vollständig online durchgeführt, was die Ergebnisse jedoch nicht einschränkte, wie Andreasen et al. [3] in ihrer Arbeit zeigten. Ein Problem, das aus den Sprachaufzeichnungen der Probanden hervorging, ist, dass sie die Fragebögen möglicherweise nicht so beantwortet haben, wie sie sich fühlten. Zum Beispiel gaben einige Teilnehmer im Interview an, dass sie Schwierigkeiten hatten, das Plugin zu verstehen. In ihrer anschließenden Antwort auf den Fragebogen brachten sie dies jedoch nicht zum Ausdruck, da sie durchweg positive Antworten gaben.

7.3 Empfehlungen

Das in dieser Masterarbeit entwickelte Konzept stieß bei den Teilnehmern der Studie auf großes Interesse. In Anbetracht des begrenzten Umfangs, in dem das Plugin und die Website getestet werden konnten, ist es sinnvoll, sie im Rahmen zukünftiger Forschung weiter zu evaluieren. Darauf aufbauend werden nachfolgend Vorschläge, die die Probanden geäußert haben, für weiterführende Untersuchungen gegeben.

Bezüglich des Plugins, das bei der im Rahmen dieser Arbeit durchgeführten Studie verwendet wurde, haben manche Teilnehmern vorgeschlagen, bei der nächsten Arbeiten eine besser gestaltete Version für den Benutzer zu entwerfen. D.h. es wäre von großem Nutzen, wenn der Benutzer mit diesem Plugin beispielsweise alle negativen Kommentaren mit einem Klick aufgelistet bekommt. Es wäre auch sehr hilfreich, der Webseite einen Teil hinzuzufügen, der die Anzahl der Beiträge zu einem Repository oder dem Code anzeigt. Es wäre wünschenswert, wenn gesehen werden könnte, wie oft ein Repository eine Aktualisierung (Commit) erhalten hat. Ein Repository kann in einem Commit eine Menge negativer Kommentare haben, aber im nächsten Commit kann sich diese Polarität ändern. Eine Korrelation zwischen Übertragungen und Kommentaren wäre also

vorteilhaft. Ein Studienteilnehmer hat auch empfohlen, verschiedene Sprachen zu unterstützen, da manche Kommentare in einer anderen Sprache geschrieben werden. Für künftige Veränderungen im Bereich der Quelldaten im Server ist das Plugin vorbereitet, wodurch sich dies durch wenige Zeilenergänzungen bewerkstelligen ließe. Durch diese Maßnahmen kann den Erwartungen der Studienteilnehmer entgegen gekommen werden.

Kapitel 8

Zusammenfassung und Ausblick

In diesem letzten Kapitel wird ein zusammenfassender Überblick gegeben. Außerdem wird ein Ausblick auf mögliche Änderungen und Erweiterungsmöglichkeiten das Plugin und die Webseite betreffend gegeben.

8.1 Zusammenfassung

Die Untersuchung der Stimmung von Texten im Kontext der Softwareentwicklung ist bereits Gegenstand zahlreicher Arbeiten. Ziel dieser Arbeit war es, ein Firefox-Plugin zu entwickeln, mittels dessen die Stimmung in einem Open-Source-Projekt automatisch analysiert und visualisiert werden kann. Ein elementarer Bestandteil war es dabei, das Konzept der Vorhersagen zu erläutern und anzuwenden. Des Weiteren sollte die Darstellung direkt in ein Webinterface integriert werden. Dafür musste sich mit den Strukturen eines Firefox-Plugin beschäftigt werden.

Hierbei sollte nach Möglichkeit auf weitere Software oder externe Programme verzichtet werden, damit das Plugin anschließend ohne Probleme in Firefox installiert werden kann. Deshalb wurde eine Lösung entwickelt, die die Strukturen von JSON nutzt und deren weiterer Code ausschließlich aus einem mit „manifest.json“ kompatiblem JavaScript-Code besteht. Es wurden dabei einfache Vorhersagemethoden selbständig implementiert und für das komplexe Modell auf einige Bibliothek zurückgegriffen. Für diese galt es, die Daten entsprechend vorzubereiten, um eine korrekte Funktion zu gewährleisten. Mit der Implementation eines Repository-Minings und der Analyse von größeren Datenmengen wurde die Sentiment-Analysis in Form von Senti4SD in einem einheitlichen Verfahren eingebunden. Darüber hinaus wurde das entwickelte Verfahren auf Open-Source-Projekten in GitHub angewendet. Anschließend wurde

eine eigene API geschrieben, die zur Kommunikation zwischen Client und Server diente und so für den reibungslosen Austausch der Daten sorgte. Auf Basis der Daten, die der Client nun empfangen konnte, galt es, eine graphische Oberfläche zu erstellen, die dem Nutzer die berechneten Werte von Sentiment-Analysis darstellt. Hierbei wurde sich für ein bei Zeitreihen häufig verwendetes Liniendiagramm entschieden, das verschiedene Polaritäten der gegebenen Kommentare darstellen konnte. Eine anschließende Evaluation des Plugins ergab eine hohe Zufriedenheit bei den Probanden, insbesondere bezüglich der einfachen Bedienung und der direkten Integration des Plugins und der Webseite.

8.2 Ausblick

Nachfolgend aufgeführte Änderungsvorschläge und Erweiterungsmöglichkeiten bezüglich des Plugins und der Webseite basieren sowohl auf den Angaben von Studienteilnehmern als auch auf Erkenntnissen, die im Rahmen des Verfassens dieser Arbeit gewonnen wurden. Es besteht die Möglichkeit, weitere zusätzliche Funktionen wie das Alter eines Projektes zu implementieren. Es wäre von Vorteil, wenn die Kommentare in Echtzeit analysiert werden könnten, damit nicht nur der Verlauf und das, was vorher passiert ist, gesehen werden kann, sondern auch, ob etwas gerade jetzt passiert. Filter für die Programmiersprache (Python, Java, C# etc.) würden beispielsweise nur Android-Studio-Projekte anzeigen. Negative Kommentare direkt auf der Webseite unter dem Diagramm sollten angezeigt werden, um Probleme und Bugs schneller identifizieren zu können. Die Repositories werden nach Auswahl analysiert, d.h. nicht alle, sondern nur die mit relevanten Themen oder Programmiersprachen für die Entwicklung. Schließlich geht es darum, zu ermitteln, warum welche Projekte gut laufen und ob sich Empfehlungen ableiten lassen, an welchem OS-Projekt es sich lohnt, sich zu beteiligen. Für alle Browser (Google Chrome und Edge) kann dieses Plugin unabhängig verwendet werden.

Anhang A

Unterlagen zur Studie

A.1 Einführung

Herzlich Willkommen bei der Studie für meine Masterarbeit „Analyse und Visualisierung der Stimmung innerhalb von Open-Source-Projekten durch Entwicklung einer Firefox-Erweiterung“. Vielen Dank, dass Sie sich die Zeit nehmen um daran mitzuwirken. Bitte denken Sie daran, dass Ihre Stimme und der Bildschirm während der Studienzeit aufgenommen werden. Diese Aufnahmen werden gelöscht, sobald die Studienauswertung abgeschlossen ist. Bei dieser Studie handelt es sich um eine „Think-Aloud“-Studie. Das bedeutet, dass Sie jeden Gedanken, den Sie während der Studie haben, laut aussprechen sollen.

A.2 Fragebogen

A.2.1 Phase 1

In dieser Phase werden allgemeine Fragen zu einem Plugin und Stimmungsanalyse gestellt:

p01: Wie alt sind Sie? (Zahleingabe)

p02: Was machen Sie beruflich? (Mehrfachauswahl)

p03: Wie oft verwenden Sie GitHub? (1: „Ganz selten“ bis 5: „täglich“)

p04: Aus welchem Grund verwenden Sie hauptsächlich GitHub? (Mehrfachauswahl aus: „Beruflich“ und „Privat“)

p05: Mir ist für das Firefox-Plugin auf GitHub zur Stimmungsanalyse wichtig. (1: „Stimme nicht zu“ bis 5: „Stimme zu“)

p06: Bitte erläutern Sie, warum Sie die Antwort auf die letzte Frage gewählt haben. (Langer Freitext)

p07: Achten Sie generell darauf, ob die Plattformen, die Sie besuchen, wie GitHub, ein Plugin haben? (1: „Nie“ bis 5: „Immer“)

p08: Haben Sie schon einmal darüber nachgedacht, ein Stimmungsanalyse-Tool für GitHub zu brauchen? (1: „Nie“ bis 5: „Immer“)

A.2.2 Phase2

Die Fragen über das Plugin und Webseite, die ich entworfen habe:

a01: Ist dieses Plugin kompliziert zu bedienen? (Ja/Nein)

a02: Sind die Informationen aus dem Plugin für Sie leicht zu finden? (1: „Stimme nicht zu“ bis 5: „Stimme zu“)

a03: Die Inhalte der Webseite war für mich leicht zu verstehen. (1: „Stimme nicht zu“ bis 5: „Stimme zu“)

a04: Die Inhalte des Plugins war für mich verständlich.(1:sehr gut, 5:sehr schlecht)

a05: Sind das Plugin und Webseite sinnvoll? (1: „Stimme nicht zu“ bis 5: „Stimme zu“)

a06: Wie ist es nun, Brauchen Sie ein Stimmungsanalyse-Tool für GitHub, nachdem Sie mein Plugin bereits benutzt haben? (Langer freier Text)

a07: Was schlagen Sie für dieses Plugin und diese Webseite vor? (Langer freier Text)

A.3 Einladung

Studie zur Masterarbeit

"Analyse und Visualisierung der Stimmung innerhalb von Open-Source-Projekten

durch Entwicklung einer Firefox-Erweiterung"

Willkommen!
Vielen Dank, dass Sie sich bereit erklärt haben, an der Studie teilzunehmen.

Diese Studie untersucht den Umgang von Endbenutzern mit Stimmungsanalyse.

Corona-Pandemie
Aufgrund der aktuellen Pandemiesituation muss die Studie online über BigBlueButton oder Skype und TeamViewer oder Chrome-remote-desktop durchgeführt werden.

Dauer

Die Durchführung der Studie dauert durchschnittlich 30 Minuten.

Kontakt

est.salajegheh@gmail.com

Voraussetzungen

- Mindestens 18 Jahre alt
- (Grundlegende) Englischkenntnisse
- Stabile Internetverbindung
- Mikrofon und Lautsprecher oder Kopfhörer

Ablauf

Allgemeine Informationen

Bitte denken Sie daran, dass Ihre Stimme und der Bildschirm während der Studienzeit aufgenommen werden. Diese Aufnahmen werden gelöscht, sobald die Studienauswertung abgeschlossen ist.

Bei dieser Studie handelt es sich um eine „Think-Aloud"-Studie. Das bedeutet, dass Sie jeden Gedanken, den Sie während der Studie haben, laut aussprechen sollen. Sehen Sie beispielweise einen Knopf, so könnten Sie „Ich weiß nicht genau, was dieser Knopf macht. Ich werde ihn jetzt einmal betätigen und schauen was dann passiert" sagen.

BigBlueButton

BigBlueButton ist ein Webkonferenzsystem. In dieser Studie wird es genutzt, um die Kommunikation zwischen Ihnen und mir zu ermöglichen (Nur Audio). Diese Kommunikation ist verschlüsselt und findet nur auf den universitätseigenen Servern statt.

Bitte besuchen Sie zu der vereinbarten Uhrzeit folgenden Link:

https://bbb.se.uni-hannover.de/b/mar-qgi-zss-b1z

Es öffnet sich folgende Seite:

Geben Sie dort, in das dafür vorgesehene Feld, Ihren Namen ein und klicken Sie dann auf „Teilnehmen“ (bzw. „Join“).

Sie werden in einen Konferenzraum weitergeleitet, in dem ich bereits auf Sie warte. Dort werden Sie gefragt, wie Sie der Konferenz beitreten möchten:

Wählen Sie bitte „Mit Mikrofon“ aus.

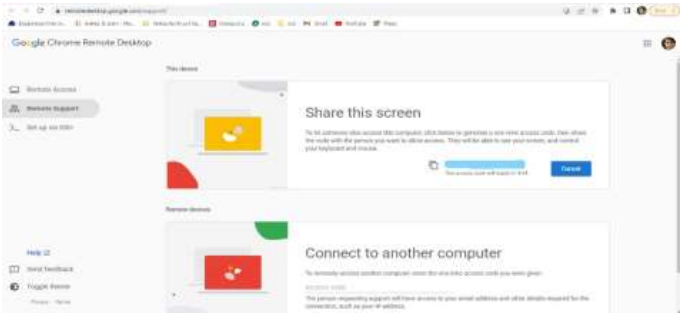
Im Konferenzraum erhalten Sie weitere Informationen über den eigentlichen Ablauf der Studie.
TeamViewer

Bitte laden Sie sich das Programm „Chrome-remote-desktop“ herunter, es muss nicht installiert und kann nach der Studie wieder gelöscht werden.

Auf den folgenden drei Seiten finden Sie eine Anleitung für diesen Vorgang.

Bitte lesen Sie sich einmal komplett durch, bevor Sie mit dem Vorgang beginnen.

1. Besuchen Sie https://remotedesktop.google.com/
2. Klicken Sie dort auf „Jetzt herunterladen“
3. Führen Sie das heruntergeladene Programm aus.
4. Es sollte sich folgendes Fenster öffnen(es ist ein ad-on auf Chrome):



5.Dann gebe ich Ihnen ein Code, das Sie unter „Connect to another computer“ eingeben.
Geben Sie, wenn Sie danach gefragt werden, dass Passwort, welches ich Ihnen ebenfalls über BigBlueButton mitteile, ein.
6.Sie haben nun Zugriff auf meinen Rechner.

Nach der Studie

1. Beenden Sie die Chrome-remote-desktop Sitzung über das „stop sharing“ in einem Fensters. Sie können Chrome-remote-desktop nun wieder löschen.
2. Schließen Sie den BigBlueButton-Tab in Ihrem Browser.

Vielen Dank für Ihre Teilnahme!

Elham Salajegheh Tezerji

Abbildungsverzeichnis

2.1 Sentiment-Analyseverfahren für Produktbewertung [31] . . . 8
2.2 Übersicht des Web-Crawlers[17] 10

4.1 Struktur des Konzepts 21
4.2 Vereinfachte Struktur eines Repository mit Issues 22
4.3 Seart Tool 23
4.4 Klassifikation der Sentimenanalyse 24
4.5 Ansicht des Konzepts des Plugins 26

5.1 Übersicht der vereinfachten Darstellung 28
5.2 Ansicht der Beziehung der Tabellen 32
5.3 Ansicht der REPOS-Tabelle 34
5.4 Ansicht der ISSUES-Tabelle 35
5.5 Ansicht der COMMENTS-Tabelle 36
5.6 Verfahren des Fremdschlüssels in Repository zz85/glsl-optimize mit Limit 5 38
5.7 Laden der Repositories 39
5.8 Verfahren des Ladens der Kommentare 40
5.9 Übersicht des Child-Processes 43
5.10 Übersicht der Datenanalyse 45
5.11 Ansicht des Plugins 48
5.12 Ansicht der Startseite von der Webseite 49
5.13 Ansicht der Database-Statistic-Seite 49
5.14 Linien-Diagramm für das „zulip“ Repository mit zahlreichen Emotionalitätswerten 51
5.15 Linien-Diagramm basierend auf Emotionalitätswert und Zeitfilter 52

6.1 Übersicht der Forschungsfragen 54
6.2 Nachdenken über ein Stimmungsanalyse-Tool für GitHub . . 58
6.3 Achten darauf, ob die Plattformen ein Plugin haben? 59
6.4 Meinung der Probanden bzgl. Plugin und Webseite 60

Literaturverzeichnis

[1] N. Agarwal and U. Rathod. Defining 'success' for software projects: An exploratory revelation. *International journal of project management*, 24(4):358–370, 2006.

[2] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi. Senticr: a customized sentiment analysis tool for code review interactions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 106–111. IEEE, 2017.

[3] M. S. Andreasen, H. V. Nielsen, S. O. Schrøder, and J. Stage. What happened to remote usability testing? an empirical study of three methods. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1405–1414, 2007.

[4] M. Babik and L. Hluchy. Deep integration of python with web ontology language. In *In Proceedings of the 2nd Workshop on Scripting for the Semantic Web*. Citeseer, 2006.

[5] D. Bartholomew. Mariadb vs. mysql. *Dostopano*, 7(10):2014, 2012.

[6] I. E. Bjerke, M. A. Puchades, J. G. Bjaalie, and T. B. Leergaard. Database of literature derived cellular measurements from the murine basal ganglia. *Scientific data*, 7(1):1–14, 2020.

[7] T. Bray et al. The javascript object notation (json) data interchange format. 2014.

[8] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli. Sentiment polarity detection for software development. *Empirical Software Engineering*, 23(3):1352–1382, 2018.

[9] V. Cosentino, J. L. C. Izquierdo, and J. Cabot. A systematic mapping study of software development with github. *IEEE Access*, 5:7173–7192, 2017.

[10] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[11] A. L. Davis. Databases. In *Modern Programming Made Easy*, pages 163–171. Springer, 2020.

[12] J. H. Dawes, M. Han, O. Javed, G. Reger, G. Franzoni, and A. Pfeiffer. Analysing the performance of python-based web services with the vypr framework. In *International Conference on Runtime Verification*, pages 67–86. Springer, 2020.

[13] R. Elmasri and S. B. Navathe. Fundamentals of database systems, 2011. *Google Scholar Google Scholar Digital Library Digital Library*, 2011.

[14] F. Fischer. *Geschäftsmodelle ausgewählter kommunaler IoT-Anwendungen und deren Bewertung für die Stadt Herrenberg.* PhD thesis, Hochschule für öffentliche Verwaltung und Finanzen, 2019.

[15] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database system implementation*, volume 672. Prentice Hall Upper Saddle River, 2000.

[16] D. Ghimire. Comparative study on python web frameworks: Flask and django. 2020.

[17] S. GOEL, M. BANSAL, A. K. SRIVASTAVA, and N. ARORA. Web crawling-based search engine using python. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 436–438. IEEE, 2019.

[18] D. Graziotin, X. Wang, and P. Abrahamsson. Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ*, 2:e289, 2014.

[19] M. Herrmann. *Automatische Klassifikation von Aussagen in Meetings von Entwicklungsteams.* PhD thesis, Bachelor's thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2020.

[20] S. Heuer. Andreas heuer, gunter saake: Datenbanken: Konzepte und sprache, 1995.

[21] J. Horstmann. Computer-gestützte analyse des kommunikationsverhaltens in entwicklerteams unter berücksichtigung digitaler medien. *Magisterarb. Master's thesis, Gottfried Wilhelm Leibniz Universität Hannover*, 2019.

[22] M. R. Islam and M. F. Zibran. A comparison of software engineering domain specific sentiment analysis tools. In *2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER)*, pages 487–491. IEEE, 2018.

[23] K. Jababo. Database systems.

[24] L. Köhler. Automatisierte analyse und visuelle aufbereitung von datenschutzerklärungen. 2021.

[25] R. S. Lazarus. *Emotion and adaptation*. Oxford University Press, 1991.

[26] E. D. Liddy. Natural language processing. 2001.

[27] 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 *Proceedings of the 40th international conference on software engineering*, pages 94–104, 2018.

[28] B. Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[29] B. Liu et al. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2(2010):627–666, 2010.

[30] P. Lokhande, F. Aslam, N. Hawa, J. Munir, and M. Gulamgaus. Efficient way of web development using python and flask. 2015.

[31] W. Medhat, A. Hassan, and H. Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113, 2014.

[32] A. Meroño-Peñuela and R. Hoekstra. grlc makes github taste like linked data apis. In *European Semantic Web Conference*, pages 342–353. Springer, 2016.

[33] J. Murphy, N. H. Hashim, and P. O'Connor. Take me back: validating the wayback machine. *Journal of Computer-Mediated Communication*, 13(1):60–75, 2007.

[34] 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*, pages 158–168, 2020.

[35] N. Novielli, D. Girardi, and F. Lanubile. A benchmark study on sentiment analysis for software engineering research. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, pages 364–375. IEEE, 2018.

[36] B. Pang, L. Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in information retrieval*, 2(1–2):1–135, 2008.

[37] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*, 2002.

[38] T. Powelske. Analyse von stimmungen in open-source-projekten anhand von text-basierter kommunikation. 2020.

[39] F. Razzoli. *Mastering MariaDB*. Packt Publishing Ltd, 2014.

[40] A. Setiyadi and E. Setiawan. Information system monitoring access log database on database server. In *IOP Conference Series: Materials Science and Engineering*, volume 407, page 012110. IOP Publishing, 2018.

[41] R. Sheldon and G. Moes. *Beginning MySQL*. John Wiley & Sons, 2005.

[42] G. Stepanek. Software programming secrets: Why projects fail. APress,2012.

[43] 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(12):2544–2558, 2010.

[44] A. S. M. Venigalla and S. Chimalakonda. Stackemo: towards enhancing user experience by augmenting stack overflow with emojis. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1550–1554, 2021.

[45] K. Werder and S. Brinkkemper. Meme-toward a method for emotions extraction from github. In *2018 IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pages 20–24. IEEE, 2018.

[46] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational linguistics*, 35(3):399–433, 2009.

[47] T. YANG, C. GAO, J. ZANG, D. LO, and M. R. LYU. Tour: Dynamic topic and sentiment analysis of user reviews for assisting app release.(2021). In *Proceedings of WWW*, volume 21, pages 19–23.

[48] A. Zakiah and M. N. Fauzan. Collaborative learning model of software engineering using github for informatics student. In *2016 4th International Conference on Cyber and IT Service Management*, pages 1–5. IEEE, 2016.

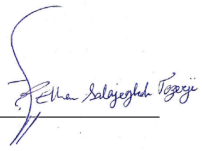
[49] T. Zhang, B. Xu, F. Thung, S. A. Haryono, D. Lo, and L. Jiang. Sentiment analysis for software engineering: How far can pre-trained transformer models go? In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 70–80. IEEE, 2020.

[50] C. Zhou, C. Sun, Z. Liu, and F. Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.

Erklärung der Selbstständigkeit

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

Hannover, den 16.05.2022



Elham Salajegheh Tezerji