

Heinrich-Heine-Universität
- Düsseldorf -

Projektarbeit

Thema:
**Programm zur Analyse von WhatsApp-Chatdaten mit
Hilfe verschiedener NLP-Tools**

Prüfer:
Kilian Evang

Bewertung:
95/100 Punkten (Note 1.0)
Kilian Evang, 04.11.2022

Verfasser:
Yunus Oscar Renz
Kölner Landstr. 197
40591 Düsseldorf

Bachelor Computerlinguistik

Eingereicht am:
31. Oktober 2022

Inhaltsverzeichnis

Inhaltsverzeichnis	1
Einleitung	3
1.1 Abstract	3
1.2 Fragestellung und Ziel	3
2 Theoretischer Rahmen	4
2.1 Vorgehensweise	4
2.1.1 Datenbeschaffung	4
2.1.2 Projektplanung	4
2.1.3 Projektumsetzung	5
2.2 Verwendete Technologien	6
2.2.1 Python Bultin-Modules	6
2.2.2 Installierte Fremd-Libraries	6
3 Praktische Umsetzung	8
3.1 Aufbau des Repositorys	8
3.1.1 Data Ordner	8
3.1.2 Database Ordner	8
3.1.3 Utils Ordner	8
3.1.4 Restliche Dateien im Repository	9
3.2 Funktionsweise des Programms	9
3.2.1 Entgegennahme	9
3.2.2 Umwandlung & Verarbeitung der Rohdaten	10
3.2.3 Analyse des Chatverlaufs	10
3.2.4 Datenvisualisierung	11
3.2.5 Datenausgabe	11
4 Handhabung und Ergebnisse	13
4.1 Verwendung des Programms	13
4.1.1 WhatsApp-Chat exportieren	13
4.1.2 Analyse durchführen	13
4.1.3 PDF-Report erhalten	14
4.1.4 Export der einzelnen Analysedaten	14
4.2 Erstellte Grafiken	14
4.2.1 Matplotlib Diagramme und Graphen	14
4.2.2 WordCloud Grafiken	15
4.3 Aufbau der Ausgabe	16
4.3.1 Genereller Abschnitt	16
4.3.2 Absenderspezifischer Abschnitt	17

5	Schluss teil und Fazit	18
5.1	Stand des Projekts	18
5.2	Ausbaumöglichkeiten	18
5.3	Kritische Betrachtung der Ergebnisse	19
5.4	Persönliche Erkenntnisse	19
5.5	Resümee	21
	Anhang	22
	Anhangsverzeichnis	22
	Literaturverzeichnis	29
	Ehrenwörtliche Erklärung	30

Einleitung

1.1 Abstract

Instant Messaging Dienste werden weltweit und rund um die Uhr benutzt, um Nachrichten zu versenden. Auf WhatsApp, die populärste Messaging-App mit zwei Milliarden aktiven Nutzern (Singh, 2020), werden rund 100 Milliarden Nachrichten pro Tag verschickt. Deutschland hat weltweit die sechstmeisten Benutzer von WhatsApp (Dean, 2022).

In wenigen Monaten des Nachrichtenaustausch entstehen riesige Mengen an Nachrichten, die jedoch genauso schnell im Chatverlauf immer weiter nach oben rutschen und in Vergessenheit geraten. Dabei ist die Frage entstanden, welche interessanten Schlüsse man aus diesen Daten ziehen kann. Ein WhatsApp Analysator-Programm, welches eine Analyse durchführt und Informationen für einen Nutzer übersichtlich darstellt, wäre nicht nur eine anspruchsvolle Programmieraufgabe, sondern kann dem einzelnen Nutzer Aufschluss über sein Chat-Verhalten und Gewohnheiten zeigen. Es kann dazu führen, Verhaltensweisen zu erkennen und abzuändern.

1.2 Fragestellung und Ziel

Im Rahmen des Projektes wird die folgende Forschungsfrage behandelt: In welcher Art und Weise lässt sich ein WhatsApp Chatverlauf mit der Verwendung von Tools zur Verarbeitung von natürlicher Sprache analysieren?

Zur Beantwortung der Forschungsfrage ist vorab zu erläutern, wie ein solches Projekt aufgebaut und strukturiert werden kann und wie die erhobenen Analysedaten nutzerfreundlich und leicht verständlich dargestellt werden können.

Als Ziel wird die Entwicklung eines Programmes gesetzt, welches Chatverläufe einliest und analysiert. Im Rahmen der Realisierung soll eine grafisch-ansprechende Analyse für den Nutzer ausgegeben werden. Die Speicherung persönlicher Daten wird in diesem Verfahren ausgeschlossen. Zusammengefasst wird im Rahmen der Projektarbeit der Verlauf des Projektes, das finale Repository, dessen Funktionsweise und die Ergebnisse sowie Erkenntnisse beschrieben.

2 Theoretischer Rahmen

2.1 Vorgehensweise

2.1.1 Datenbeschaffung

Als erster Schritt wird das Thema der Datenbeschaffung herangezogen. In diesem ist zu erläutern, aus welcher Quelle die entsprechenden Analysedaten beschafft werden können. Da es sich hier um sensible personenbezogene Daten handelt, werden hauptsächlich Konversationen zwischen dem Verfasser der Projektarbeit und dessen Bekanntenkreis verwendet. Dabei ist von jeder beteiligten Person eine Einverständniserklärung eingeholt worden und zusätzlich wurden Chatnachrichten anonymisiert. Dies wurde durch das Unkenntlich Machen von persönlichen Daten wie Namen, Nummern, Adressen, Links und anderen identifizierbare Daten durch Beispieldaten realisiert.

Da die Aufgabe darin besteht, ein Programm zu entwickeln, welches dem Nutzer die Möglichkeit gibt, den eigenen Chatverlauf zu analysieren, braucht es keinen großen Korpus an Konversationen. Somit ist eine geringe Menge an Testmaterial ausreichend, um das Programm auf Fehler zu testen und die Qualität der erstellten Berichte zu gewährleisten. Die für den Test anonymisierten Chatverläufe sind zum Großteil deutschsprachige Konversationen, jedoch wurde ebenfalls darauf geachtet, Chats zu verwenden, bei denen Teile der Nachrichten anderssprachig sind. Dadurch soll geprüft werden, wie das Programm und die verschiedenen NLP-Tools auf Chats zwischen bilingualen Chatpartnern oder Jugendlichen, die oft Anglizismen und ganze englische Phrasen verwenden, performen.

2.1.2 Projektplanung

Im Rahmen der Projektplanung werden die zu realisierenden Features erarbeitet. Außerdem wird darüber entschieden, welche Statistiken erhoben werden und auf welche Art die Datenausgabe an den Nutzer erfolgt. Zusätzlich wird die Grobstruktur des Projektes festgelegt.

Das Programm soll benutzerfreundlich sein und möglichst einfach bedienbar sein. Deswegen kann es lediglich eine Eingabe entgegennehmen und daraufhin die Analyse autonom im Hintergrund durchführen. Der Nutzer soll nur eine Ausgabe bekommen, bei der alle Daten und Statistiken zusammengefasst werden.

Mögliche erfassbare Statistiken reichen von einfach zu komplex, von der Anzahl der Sender und Nachrichten bis zu named-entity-recognition (auf Deutsch: Eigennamen-Erkennung). Doch auch vermeintliche unspektakuläre Werte wie die Länge der Nachrichten sind in dem Kontext von Instant Messaging Diensten wie WhatsApp interessant. Zwar sind auf einem Instant Messaging Dienst wie WhatsApp die Nachrichten hauptsächlich kurz, dennoch gibt es erwähnenswerte Nachrichten, welche in Ihrer Zeichenslänge ungewöhnlich sind, wie z.B. Liebesbriefe oder Streitgespräche. Die Sentimentanalyse ist ein sehr beliebtes NLP-Thema, welches auch im Kontext von Textnachrichten eine große Rolle spielt.

Aber auch die Verteilung von Nachrichten und Medien sind wichtige Werte, die bei einem WhatsApp-Analysator nicht fehlen sollten. Ebenfalls ist der zeitliche Kontext eine wichtige Komponente: Zu welchen Zeiten (Wochentagen, Uhrzeiten) werden viele und wann wenige Nachrichten versendet? An wie vielen Tagen wurden Nachrichten und an wie vielen keine versendet? Wie lang sind die Zeitspannen ohne Nachrichten?

All diese Fragen kann man aus den Chat-Daten beantworten. Allerdings ist bei einer solchen Menge an unterschiedlichen Daten eine übersichtliche und verständliche Ausgabe ebenso wichtig wie die Analyse selbst.

Das Portable Document Format (PDF) ist ideal, um eine Ausgabe zu gestalten, da es sich auf jedem Gerät anzeigen lässt und dabei nicht das gewünschte Format verliert. Außerdem sind sie leicht zu teilen und in allen Sektoren der Standard für diese Art von Dateien. Zudem unterstützen PDFs die Darstellung aller möglichen Datentypen, dazu kommt, dass das Erstellen solcher Dateien simpel und zeiteinsparend ist. (Adobe, 2016)

2.1.3 Projektumsetzung

Da das Projekt auf GitHub hochgeladen wird, ist eine deutliche Trennung von Daten und Code als sinnvoll bewertet worden. Dafür wird ein data-Ordner angelegt, indem die Chatverläufe gesichert werden. Damit keine ungewollten Daten in das Repository hochgeladen werden, wird zusätzlich eine .gitignore Datei verwendet, welche direkt in dem Projektverzeichnis liegt. In der Datei wird die Zeile data/* eingetragen, sodass GitHub die Inhalte des data-Ordners ignoriert und ein Hochladen ausgeschlossen ist.

Für den Code wird ein utils-Ordner angelegt, in dem fortan Funktionen in einzelnen Python-Modules gespeichert werden. Diese werden nach den Aufgabenbereichen unterteilt, um somit für eine bessere Lesbarkeit des Codes zu sorgen. Der Code ist somit auch für die Wiederverwendbarkeit geeignet.

Auf den genaueren Aufbau wird in dem Abschnitt 3.1 *Aufbau des Repositorys* eingegangen.

2.2 Verwendete Technologien

Als Code-Hosting-Plattform und zur Kontrolle verschiedener Versionen wird wie beschrieben die Lösung GitHub verwendet. Dieses Projekt wird in der Programmiersprache python in der Version 3.9.7 geschrieben. Der benutzte Package Manager ist pip in der Version 22.3.

2.2.1 Python Bultin-Modules

Der Code greift auf einige in python enthaltene Packages zu, wie zum Beispiel das collections-Module, welches das Zählen von hashfähigen Objekten erleichtert, das re-Module, um reguläre Ausdrücke (regex) zu verwenden, das timeit-Module, um Zeitmessungen durchzuführen und das os-Module, um auf die Funktionalitäten des Betriebssystems zuzugreifen.

2.2.2 Installierte Fremd-Libraries

Zusätzlich werden Fremd-Libraries verwendet, welche zuerst importiert werden müssen. Diese sind mit der verwendeten Version in der requirements.txt Datei im Repository zu finden.

Alle gelisteten Libraries können mit dem Terminal-Befehl `python -m pip install -r requirements.txt` installiert werden. Zuvor sollte jedoch ein Virtual Environment erstellt werden.

Als Speichermedium der Chat-Daten und der Statisten sowie für die umfangreiche Datenanalyse, Datenmanipulation und für numerische Berechnungen wurde pandas (1.5.0) sowie numpy (1.23.4) verwendet.

Die spacy-Library (3.4.2) wendet mit dem de-core-news-sm (3.4.0) Sprachmodell die Verarbeitung von natürlicher Sprache und die Erkennung von Eigennamen an. Die nltk-Library (3.7) liefert in diesem Projekt deutschsprachige Stoppwörter für die WordCloud.

Für die Stimmungsanalyse in der deutschen Sprache wurden verschiedene Libraries verwendet, jedoch hat es hierbei unterschiedliche Probleme gegeben. Letztlich konnte textblob-de (0.4.3) bzgl. der Ergebnisse und hinsichtlich der Performance überzeugen, worauf im Abschnitt 5 *Schluss teil und Fazit* eingegangen wird.

Die `unidecode`-Library (1.3.6) wird für ASCII-Transliterationen verwendet und `emojis` (0.6.0) zur Emoji Handhabung und Umwandlung.

Die Grafik-Bibliothek `matplotlib` (3.6.1) visualisiert Daten in Form von verschiedensten Diagrammen und `wordcloud` (1.8.2.2) ist zuständig für die WordCloud-Erzeugung. Mit der `fpdf`-Library (1.7.2) ist die PDF-Erzeugung und Bearbeitung möglich.

Außerdem sollten zwingend `setuptools` (65.5.0) für die Bedienung möglicher Abhängigkeiten und `wheel` (1.8.2.2), um die Installation von Python-Paketen zu beschleunigen, installiert sein.

3 Praktische Umsetzung

3.1 Aufbau des Repositorys

Bei dem Aufbau des Repositorys wurde entschieden, den Code in verschiedene Python-Modules, je nach Funktionen und Nutzen aufzuteilen. Im Folgenden werden kurz die Aufteilung und Aufgaben der verschiedenen Modules erläutert. Auf die detaillierte Umsetzung und Funktionsweise wird im späteren Abschnitt 3.2 *Funktionsweise des Programms* eingegangen. Für eine übersichtliche Darstellung des Repositorys siehe *Abbildung 1: Repository-Tree* im Anhang.

3.1.1 Data Ordner

In dem data Verzeichnis befindet sich die `sample_chat.txt` Datei, ein Beispielchat, welcher zum Testen verwendet werden kann. Das `data/output` Verzeichnis wird als Speicherort für die erstellten Grafiken verwendet, dort ist auch der finale PDF-Report zu finden.

3.1.2 Database Ordner

Der database-Ordner dient als Datenbank. In ihm befinden sich zwei Dateien: `constants.py` (c.), eine Datei, die Konstanten, zum Beispiel Regex-Pattern oder Statistikbezeichnungen enthält. Und `variables.py` (v.), welche Variablen enthält, die beim Durchlaufen des Programms gefüllt werden. Der Speicherort wurde gewählt, damit die Daten für alle Modules abrufbar und bearbeitbar sind.

3.1.3 Utils Ordner

Im `utils`-Ordner befinden sich alle Funktionen, die benötigt werden, um eine WhatsApp Chatverlauf-Datei einzulesen, zu analysieren und anschließend in Form eines PDFs auszugeben. Damit die Funktionalität des Codes verständlicher und organisierter ist, habe ich die Funktionen in mehrere Modules, je nach Aufgabe, aufgeteilt.

Vor der eigentlichen Analyse nimmt das `getter.py`-Module Nutzereingaben in Form eines Dateipfades entgegen und prüft, ob es sich um das richtige Dateiformat handelt. Das `converter.py`-Module wandelt die Chat-Rohdaten in ein geeignetes Format um. Welche anschließend von dem `analyzer.py`-Module für die umfangreiche Analyse der Konversation verwendet wird. Während das `plotter.py`-Module für die grafische Darstellung der Ergebnisse der Analyse zuständig ist, ist das `outputter.py`-Module für das

Erstellen der finalen Ausgabe, dem PDF-Dokument verantwortlich. Außerdem befindet sich in dem `utils`-Ordner das `helper.py`-Module, welches verschiedene Funktionen enthält, die von allen Modules gebraucht werden, die jedoch nichts mit der eigentlichen Aufgabe des Programms zu tun haben.

3.1.4 Restliche Dateien im Repository

Die `main.py` Datei ist die Hauptdatei und greift auf alle Modules zu und führt den Code aus. In dieser Datei sieht man übersichtlich die verschiedenen Schritte, die durchgeführt werden, um einen Chatverlauf einzulesen, zu analysieren und final eine Ausgabe zu erstellen.

Die `README.md` Datei dient als Leitfaden, der Personen, die nicht an diesem Projekt beteiligt sind, eine detaillierte Beschreibung meines GitHub-Projekts liefert und beschreibt, wie das Programm funktioniert und was man für die Verwendung braucht.

Die `LICENSE.txt` Datei spezifiziert die Lizenz für Verwendung, Bearbeitung und Weitergabe des Codes innerhalb dieses Repositories.

In der `requirements.txt` Datei befinden sich die benötigten Python Libraries mit den entsprechenden benötigten Versionen. In dem Abschnitt *2.2 Verwendete Technologien* wird gezielt auf die einzelnen Libraries eingegangen und wofür diese verwendet wurden.

3.2 Funktionsweise des Programms

Das `main.py`-Module führt die Analyse der WhatsApp Daten durch, in dem die Daten eingenommen, umgewandelt, analysiert, visualisiert und anschließend ausgegeben werden. Für die Analyse werden die verschiedenen Module, die sich in dem `utils`-Ordner befinden, benötigt. Die `main.py` greift auf diese Funktionen zu und speichert die Analysedaten in Platzhaltern, die sich in der `variables.py` Datei (v.) befinden.

3.2.1 Entgegennahme

Die Eingabe der Daten erfolgt über die Nutzereingabe des Dateipfades in dem `getter.py`-Module, hier wird auch das meiste Error-Handling durchgeführt. Die `get_file()` Funktion nimmt den Dateipfad entgegen bzw. erfragt mit der `get_path()` Funktion diesen über eine Terminaleingabe. Der Pfad wird durch `check_file_format()` auf Dateiformat und Richtigkeit überprüft. Falls der Pfad leer ist, wird der Beispieltext `data/sample_chat.txt` verwendet. Falls der Dateipfad auf eine `.zip`-Datei weist, wird diese erst extrahiert und der Dateipfad der extrahierten Datei verwendet.

Anschließend wird der Inhalt eingelesen und als Liste von Zeilen zurückgegeben. Beim Eintreten eines Fehlers wird mithilfe der `helper.off()` Funktion über das Terminal eine simple und verständliche Fehlermeldung ausgegeben.

3.2.2 Umwandlung & Verarbeitung der Rohdaten

Das `converter.py`-Module ist für die Umwandlung der Chat-Rohdaten zu einem `pandas.DataFrame` zuständig. Die `convert_list()` Funktion wandelt die einfache Zeilen-Liste in eine komplexere Liste von Listen um, indem sie jede Zeile der `convert_ln()` Funktion übergibt. Diese unterteilt den String mit Hilfe von einem regex-Muster (siehe *Abbildung 7*) in die verschiedenen Bestandteile und gibt sie als Liste zurück, bestehend aus `[date, datetime, sender_name, message, sentiment]`. Die Sentimentanalyse wird von `Textblob-de` durchgeführt. Die erzeugten Listen werden in eine Liste von Listen eingetragen, welche schlussendlich mit `convert_to_df()` in ein `pandas.DataFrame` umgewandelt und in den `v.chat` Platzhalter gespeichert wird.

Für dieses Speichermedium wurde sich entschieden, da `pandas` bereits viele benötigte Funktionen enthält, die andernfalls manuell nachentwickelt werden müssten. Zusätzlich sind `DataFrames` einfacher zu benutzen, schneller und leistungsfähiger als in Python integrierte Speichermedien. (Hagedorn et al., 2021)

3.2.3 Analyse des Chatverlaufs

Das `analyzer.py`-Module ist für die umfangreiche Datenanalyse zuständig und trägt die erhobenen Analysedaten in die vorgesehenen Platzhalter in der `variables.py`-Datei im `database` Ordner.

Nachdem die `prepare_database()` Funktion grundlegende Vorbereitungen trifft, wird mit der `analysis_per_sender()` und der `calc_remaining_stats()` Funktion die eigentliche Analyse durchgeführt. Zuerst werden mit `analysis_per_sender()` für jeden Absender Nachrichten, die keine Textnachrichten sind, von der `clense_df()` Funktion entfernt; die Anzahlen werden in `v.stats` eingetragen. Die Funktion verwendet dafür die in `c.STATS_PATTERN` definierten regex Muster. Im nächsten Schritt zählt die `count_emojis()` Funktion die verschiedenen Emojis mithilfe der `emojis` Library. Dann werden nicht-lateinische Buchstaben entfernt, um die Nachrichten in `pyfpdf` darstellen zu können und damit keine Fehler bei der weiteren Verarbeitung passieren werden.

Anschließend führt die `nlp_analysis()` Funktion eine `spacy` Analyse der linguistischen Merkmale durch. Indem die Vorkommen unterschiedlicher Wörter, Lemmas, Stopwörter, nicht-Stopwörter, part-of-speech (POS)-Tags (Deutsch: Wortarten) und

named entities (Deutsch: Eigennamen) gezählt und gelistet werden. Daraufhin werden weitere Analysedaten, wie durchschnittliche Wortlänge, Nachrichtenlänge und Sentiment, von der `calc_stats()` Funktion hergeleitet.

Die restlichen Statistiken werden mit weiteren komplexen Dataframe Operationen von der `calc_remaining_stats()` und der `calc_time_stats()` Funktion berechnet.

3.2.4 Datenvisualisierung

Das `plotter.py`-Module für die Visualisierung von Analyse-Ergebnissen zuständig, indem es mit der `matplotlib` Library die Grafiken erzeugt, die für das PDF-Dokument benötigt werden.

Die `plot_data()` Funktion sorgt dafür, dass die Funktionen, die für das Zeichnen der Graphen zuständig sind, richtig aufgerufen, geplottet und gespeichert werden. Zuerst werden die absenderspezifischen und danach die generellen Darstellungen erstellt.

Die `msg_pie()`, `grouped_media_bars()`, `message_time_series()`, `activity_heatmaps()`, `sent_pies()`, `violinplot()`, `boxplot()` und `word_cloud()` Funktionen erstellen die eigentlichen Grafiken und speichern sie in den `generalpage` bzw. `senderpages` Ordner im `/data/output/images` Verzeichnis. Die `c_m()` Funktion hält eine einheitliche Farbverteilung, die sich an den Farben von WhatsApp orientiert, unter den Absendern aufrecht.

Auf die unterschiedlichen Grafiken gehe ich im Abschnitt *4.2 Erstellte Grafiken* ein.

3.2.5 Datenausgabe

Das `outputter.py`-Module ist für die finale Ausgabe des Programmes zuständig, indem es den PDF-Report erstellt und in dem `data/output` Ordner speichert. Für die Erzeugung wird die `pyfpdf` Library verwendet.

Beim Aufrufen der `make_pdf_report()`-Funktion werden zuerst durch die `create_reports()` Funktion die Platzhalter `v.txt_reports`, `v.time_reports` und `v.nlp_reports` gefüllt. Generelle, zeitliche und NLP-Statistiken werden zu Texten geschrieben, welche für jeweils jeden Absender und einen, der für alle Teilnehmer gilt, erstellt werden. Um den richtigen Numerus von Nomen auszugeben, werden die `calc_num()`-Funktionen verwendet.

Danach wird das PDF gestaltet. Die Hilfsfunktion `new_section()` ändert Text-Eigenschaften, wie Text-Größe, -Stil und -Farbe, als auch die Position, an welche der Text geschrieben werden soll. Diese verschiedenen `pyfpdf`-Funktionen in einer

zusammenzufassen, sorgt für mehr Übersichtlichkeit und weniger redundanten Code. Auf der angerufenen Seite fügt die `add_title_footer()` Funktion den Titel, die Fußzeile und die Seitenzahl hinzu.

Die Anzahl der gesendeten Anhänge für einen Sender fügt die `insert_media_table()` in Form einer Tabelle ein. Und die `commons_table()` Funktion erstellt eine Tabelle aus dem übergebenen Frequenzen DataFrame. Leider unterstützt `pyfpdf` keine Emojis, also werden diese mithilfe von der `emojis`-Library die einzelnen Emojis zu den schriftlichen Äquivalenten umwandelt. (siehe *Abbildung 13*) Falls unter der Bezeichnung nichts verstanden wird, kann diese ausgewählt werden, um zu einer Google-Bildersuche weitergeleitet zu werden. Die `ts_info()` Funktion schreibt weitere Informationen unterhalb des Zeitstrahl-Graphen.

Außerdem fügt die `add_nlp_page()` eine ganze Seite mit ausschließlich NLP-Analyse-daten zu dem übergebenen Absender zu dem Dokument an. `make_pdf_report()` ist die Funktion, welche alle genannten Statistiken, Tabellen, Texte, Graphen und Grafiken in dem vorgesehenen Format in ein PDF-Dokument schreibt und anschließend dieses in die gewünschten Ordner `data/output` speichert.

Die Formatierung des PDFs wird in dem Abschnitt *4.3 Aufbau der Ausgabe erläutert*.

4 Handhabung und Ergebnisse

4.1 Verwendung des Programms

4.1.1 WhatsApp-Chat exportieren

Für die Analyse eines Chatverlaufes verwendet das Programm Daten, die vom Nutzer direkt aus WhatsApp exportiert werden können. Dies geht bei WhatsApp sehr einfach:

Dazu ist der gewünschte Chat aus, navigieren Sie auf die Kontaktdetails bzw. die Informationsseite dieser Konversation und wählen Sie "Export Chat" und "Without Media".

Sie erhalten eine `.zip`-Datei, aus der man nach dem Entpacken die `.txt`-Datei enthält, mit der die Analyse durchgeführt werden soll. Sie umfasst alle Nachrichten von allen Absendern in diesem Chat. Zum Analysieren können die entpackte `.txt`-Datei, aber auch die `.zip`-Datei verwendet werden. Das Programm funktioniert jedoch nur, wenn der Chat aus iOS und ohne Mediendateien exportiert wurde. Laden Sie vorzugsweise die Originaldatei hoch.

4.1.2 Analyse durchführen

Nachdem Sie den Chat exportiert haben, muss für die Verarbeitung das Programm den Chat importieren. Es gibt zwei Möglichkeiten dafür: Entweder können Sie in der `main.py` Datei den Dateipfad der `.txt`- oder der `.zip`-Datei an der vorgesehenen Stelle angeben (gekennzeichnet durch einen Pfeil im Kommentar). Oder Sie führen die Datei `main.py` aus und geben den Dateipfad im Terminal an, wenn Sie dazu aufgefordert werden.

Das Programm wird dann die Datei verarbeiten. Das kann etwas dauern, dies ist abhängig von der Größe der Datei, der Anzahl der Sender und Ihrem Computer.

Während der Analyse gibt das Programm die durchlaufenen Schritte und die für den jeweiligen Schritt benötigte Zeit aus. Nach Abschluss der Analyse oder falls ein Fehler währenddessen passiert, gibt das Programm eine leicht lesbare und verständliche Nachricht über das Terminal aus und beendet sich anschließend. Beispiele dafür sind *Abbildung 2: Erfolgreiche Analyse* und *Abbildung 3: Fehlermeldungen* im Anhang.

4.1.3 PDF-Report erhalten

Nach Abschluss der Analyse erhält der Nutzer einen PDF-Bericht, der alle hergeleiteten Daten und Statistiken in einem leserfreundlichen Format enthält. Der Bericht wird dafür im output-Ordner abgelegt. Die PDF-Datei trägt den Namen `Report.pdf`.

Der Bericht ist in zwei Abschnitte untergliedert. Der generelle Abschnitt enthält Information, die für alle Nutzer gelten. Dieser Abschnitt ist immer vier Seiten lang. In dem absenderspezifischen Abschnitt sind pro Chat-Teilnehmer Informationen spezifische für diese Person enthalten. Die Länge dieses Abschnittes ist von der Anzahl an Chat-Teilnehmern abhängig, pro Teilnehmer werden drei weitere Seiten Informationen ausgegeben.

Die Abschnitte des PDF-Dokuments, welche Informationen diese enthalten und auf welche Weise sie dargestellt sind, erkläre ich in dem Abschnitt *4.3 Aufbau der Ausgabe*.

4.1.4 Export der einzelnen Analysedaten

Es gibt auch die Möglichkeit, alle hergeleiteten Analysedaten zu exportieren. Dazu muss entweder die Zeile `export_database()` am Ende der Datei `main.py` auskommentiert werden. Das Programm wird dann alle abgeleiteten Statistiken in Dateien exportieren. Sie finden die Dateien in dem neu erstellten Ordner `data/exports`.

4.2 Erstellte Grafiken

Die Grafiken in diesem Projekt werden in dem `plotter`-Module mit der `matplotlib` und der `wordcloud` Library erstellt.

Das visuelle Darstellen von Daten wecken nicht nur Aufmerksamkeit und Interesse, sondern Daten werden so besser verstanden und schneller aufgenommen als eine triste Tabelle mit Zahlen (Grant, 2009).

4.2.1 Matplotlib Diagramme und Graphen

Matplotlib hat zwei Haupt-Anwendungsschnittstellen bzw. Arten der Nutzung der Bibliothek. Die explizite `axes`-Schnittstelle, die Methoden für die Erstellung von Artists (Grafiken) verwendet. Die implizite `pyplot`-Schnittstelle verfolgt die erstellte Figur und die erstellten Achsen und zu dem Objekt Artists hinzufügt. (Hunter et al., 2022)

Die `msg_pie()` Funktion plottet ein Kreisdiagramm, welches die Verteilung der gesendeten Nachrichten auf die Absender zeigt. Die prozentuale Anzahl wird in die

entsprechenden Stücke geschrieben. Eine Legende mit den Absendernamen wird hinzugefügt. Die `sent_pies()` Funktion macht dasselbe, nur für die Anzahl der gesendeten Nachrichten, die positiv bzw. negativ bewertet wurden. Eine Version pro Absender wird erstellt. (siehe *Abbildung 4*)

In einem Balkendiagramm zeichnet die `grouped_madiaBars()` Funktion die Anzahl der einzelnen Anhängertypen pro Absender. Die totale Anzahl wird bei ausreichend Platz über den Balken angezeigt. Die davor verwendeten Farben werden verwendet. (siehe *Abbildung 5*)

Ein Zeitstrahl, welcher die Anzahl der gesendeten Nachrichten im Laufe der Zeit zeigt, wird mit der `message_time_series()` Funktion erstellt. Die X-Achse nur an bestimmten Monaten Beschriftungen. Eine Version pro Absender und eine generelle wird erstellt. (siehe *Abbildung 6*)

In Form einer Heatmap visualisiert die `activity_heatmaps()` Funktion die Verteilung der gesendeten Nachrichten auf die Stunden der Wochentage. Dunkle Farben stehen für viele, helle für wenige Nachrichten, dies macht das Erkennen von Kommunikationsmustern einfacher.

Totale Anzahl steht in dem jeweiligen Kästchen geschrieben, die Textfarbe passt sich dem Hintergrund an und Nullen werden ausgeblendet. Zusätzlich wird eine Colorbar-Legende angezeigt. Erstellt werden eine Version pro Absender und eine generelle. (siehe *Abbildung 8*)

Mit Hilfe von Boxplots wird die Verteilung der Charakter- und Wort-Anzahl pro Nachricht zwischen den Chat-Teilnehmern verglichen. Die `boxplot()`-Funktion zeichnet pro Teilnehmer jeweils einen Graphen und stellt diese nebeneinander, um die Verteilung anschaulich zu machen. Die `violetplot()`-Funktion macht dasselbe und stellt die Informationen in einem Violetplot dar. (siehe *Abbildung 9 und 10*). Die Ausreißer im Boxplot werden bewusst nicht angezeigt, da sehr lange Nachrichten die Visualisierung so gut wie unbrauchbar machen (*Abbildung 11 und 12*).

4.2.2 WordCloud Grafiken

Für die verschiedenen Absender wird jeweils eine WordCloud mit der `word_cloud()`-Funktion erstellt. Große Wörter gehören zu den oft vorkommenden Wörtern, kleine zu den seltener vorkommenden. `nltk.corpus` liefert deutschsprachige Stopwörter. Die Farbpalette orientiert sich an den restlichen Grafiken (siehe *Abbildung 12*).

4.3 Aufbau der Ausgabe

Der generelle Abschnitt enthält Information, die für alle Nutzer gelten. Dieser Abschnitt ist dabei immer vier Seiten lang. In dem absenderspezifischen Abschnitt sind pro Chat-Teilnehmer Informationen spezifische für diese Person enthalten.

4.3.1 Genereller Abschnitt

Die Informationen auf den ersten vier Seiten beziehen sich auf die von allen Benutzern im Chat gesendeten Nachrichten. Die erste Seite dient als generelle Übersicht mit anschaulichen Grafiken und kurzen Beschreibungen, auf der zweiten Seite werden die Charakter- und Wortanzahl pro Nachricht in verschiedenen Plots dargestellt, die dritte Seite zeigt meist benutzte Wörter und Emojis und die vierte Seite ist mit NLP Statistiken gefüllt.

Auf der ersten Seite findet man die Anzahl der Sender sowie die Anzahl der gesendeten Nachrichten, Medien und Emojis. Die Verteilung der gesendeten Nachrichten wird in Form eines Kreis-Diagramms angezeigt. Die durchschnittliche Länge einer Nachricht und der längsten Nachricht wird in Anzahl an Wörtern und Charakteren angegeben.

Außerdem wird die Anzahl an gelöschten Nachrichten ausgegeben. Auch wird die Anzahl an Tagen, an denen mindestens eine sowie gar keine Nachrichten geschickt wurden und die längste Zeitspanne, in der keine Nachrichten geschickt wurden, angezeigt.

Die Anzahl der einzelnen gesendeten Medientypen wird in einem Balkendiagramm dargestellt, jeder Sender hat seinen eigenen Balken, die Anzahl steht über dessen Balken. Ein Zeitstrahl zeigt die Anzahl der Nachrichten im Laufe des Chatverlaufes. Darunter sieht man das Datum, an dem die erste und die letzte Nachricht sowie das Datum, an dem die meisten Nachrichten geschickt wurden. In einer Heatmap sieht man die Anzahl an Nachrichten pro Tag und pro Uhrzeit.

Auf der zweiten Seite werden Charakter und Wortanzahl pro Nachricht zwischen den Chat-Teilnehmern verglichen. Diese Daten werden pro Teilnehmer in Box-Whisker-Plots (Kastengrafiken) nebeneinandergestellt, um die Verteilung zu zeigen und einen Eindruck darüber zu vermitteln, in welchem Bereich die Daten liegen. Außerdem werden die Daten pro Teilnehmer auch noch in einem Violin-Plot nebeneinander veranschaulicht, wodurch die Verteilung noch besser zur Geltung kommt.

Auf der dritten Seite befindet sich eine Word Cloud, die viel benutzte Wörter groß und weniger benutzte Wörter klein darstellt. Darunter findet man die 20 meistbenutzten Wörter sowie Emojis und die Anzahl der Vorkommnisse in Form einer Tabelle. Die

Gesamtanzahl an unterschiedlichen Emojis wird darunter angezeigt. Auf der vierten Seite werden Statistiken zur natürlichen Sprachverarbeitung angezeigt.

Diese sind jeweils die totale Anzahl und die Anzahl an unterschiedlichen Wörtern, Lemmas, Stoppwörtern und nicht-Stoppwörtern. So wie die durchschnittliche Wortlänge und die Anzahl an erkannten Eigennamen. Hier findet man auch die totale Anzahl an unterschiedlichen Worttypen (Nomen, Verben, Adjektive usw.).

Dort befinden sich auch die 20 meistbenutzten Lemmas und Stoppwörter und die Anzahl der Vorkommnisse in Form einer Tabelle.

4.3.2 Absenderspezifischer Abschnitt

Die Informationen auf diesen Seiten sind spezifisch für den Absender, der im Untertitel und in der Fußzeile angegeben ist. Für jeden Absender gibt es drei Seiten, die erste mit allgemeinen Informationen über die von diesem Absender gesendeten Nachrichten, die zweite mit zeitlichen Statistiken und Grafiken und die dritte mit NLP-Daten.

Auf der ersten Seite ist die Anzahl der gesendeten Nachrichten, Medien und Emojis zu sehen. Die Anzahl der unterschiedlichen Medientypen für den Sender ist in einer Tabelle darunter angezeigt. In einem Kreisdiagramm wird die Verteilung von positiv und negativ bewerteten Nachrichten dargestellt. Auch hier ist die durchschnittliche Länge einer Nachricht und der längsten Nachricht in Anzahl an Wörtern und Charakteren angegeben. Darunter findet man die 20 meistbenutzten Wörter sowie Emojis und die totale Anzahl der Vorkommnisse in Form einer Tabelle. Die Gesamtanzahl an unterschiedlichen Emojis wird darunter angezeigt.

Auf der zweiten Seite zeigt ein Zeitstrahl die gesendeten Nachrichten im Laufe des Chatverlaufes für diesen Nutzer. Das Datum der ersten und letzten Nachricht sowie das Datum, an dem die meisten Nachrichten von dieser Person geschickt wurden, befinden sich darunter. Ebenfalls wird die Anzahl an Tagen, an denen diese Person keine bzw. mindestens eine Nachricht gesendet hat, als auch die längste Zeitspanne ohne Nachrichten und die durchschnittliche Anzahl an Nachrichten pro Tag angegeben.

Die darunter liegende Heatmap und WordCloud sind ebenfalls für diese Person spezifisch. Auf der dritten Seite sind die gleichen NLP-Statistiken zu finden, wie auf der letzten Seite des generellen Abschnittes, nur dass diese sich jetzt nur auf den einen Sender beziehen.

5 Schlussteil und Fazit

5.1 Stand des Projekts

Aufgrund des beschränkten zeitlichen Rahmens habe ich mich auf Einzelgespräche konzentriert und bei dem Einlesen und der Ausgabe nicht gesondert auf Gruppengespräche geachtet. Beim Testen sind einige Gruppengespräche nicht zu erfolgreichen Analyse-Erlebnissen gekommen. Theoretisch können alle Funktionen auch Konversationen mit mehr als zwei Personen verarbeiten, jedoch müssten einige Funktionen umgeschrieben werden.

Außerdem habe ich für das Entwickeln und das Testen nur von meinem iPhone aus exportierte Chats verwendet. Daher kann es sein, dass Chats, die von Telefonen mit anderen Betriebssystemen (z.B. Android, KaiOS) exportiert werden, nicht richtig analysiert werden bzw. zu Fehlern führen. Um solche Probleme zu umgehen, sollten nur von iOS-Geräten aus exportierte Chatverläufe verwendet werden. Falls man kein iPhone hat, kann man einfach eine Person mit einem iPhone fragen, ob diese den gemeinsamen Chat für einen exportieren kann. Was für ein Gerät der Chatpartner hat, spielt keine Rolle.

5.2 Ausbaumöglichkeiten

Ausbaumöglichkeiten wären zum Beispiel eine umfangreiche Unterstützung von Gruppenchats als auch von anderen Exportmedien als iOS.

Um anderssprachige Chatverläufe analysierbar zu machen, müssen nur die Sprachmodelle von spacy, numpy und texblob angepasst werden. Die Funktionen und Module sollten dank der unspezifischen und globalen Schreibweise ohne Adaptionen funktionieren. Dies könnte man einfach über eine Config-Datei lösen.

Eine multilinguale Lösung, die Sprachen erkennt und daraufhin entsprechend analysiert, stelle ich mir herausfordernd, aber nicht unmöglich vor.

Weiterführend können auch noch weitere linguistische Statistiken und Daten, wie Satzbau oder Abhängigkeiten, implementiert werden. Auch Text-Klassifikation ist in diesem Kontext sehr interessant, um beispielsweise Konversationsthemen zu extrahieren. Damit könnte eine mögliche Korrelation zwischen dem Verhältnis der Chatpartner und Konversationsthemen untersucht werden. Jedoch stellt sich hier dann die Frage, welche Testdaten verwendet werden und vor allem woher man diese bekommen soll.

5.3 Kritische Betrachtung der Ergebnisse

Angesichts der Tatsache, dass WhatsApp Konversationen meistens kurze Nachrichten enthält, ist es nicht überraschend, dass oft sprachliche und paralinguistische Abkürzungen in den Nachrichten verwendet werden. (Varnhagen et al., 2009) Laut der undergraduate Forschungsarbeit von Driscoll aus dem Jahr 2002 kommen in Kurznachrichten auch andere linguistisch schwer analysierbare Merkmale wie Clips, Akronyme, Zusammensetzungen, Suffixe, Präfixe, stimmhafte Pausen, alphanumerische Substitutionen, alternative Schreibweisen und feste Ausdrücke vor.

Die eindeutigsten Merkmale hat die germansentiment-Library geliefert, welche jedoch Probleme hatte, große Datenmengen als mehr als nur einige Sätze zu verarbeiten.

Diese sind für die meisten Sprachmodelle schwierig zu analysieren. Zum Beispiel konnten nur wenige Modelle bei der Sentimentanalyse aussagekräftige Ergebnisse liefern. Bei den meisten deutschsprachigen Modellen wurde nur einem kleinen Teil der Nachrichten (etwa 10%) überhaupt ein Wert zugeordnet. Die restlichen Nachrichten wurden mit 0 (neutral) bewertet, jedoch ist in diesem Fall nicht klar, ob die Nachricht nun wirklich als neutral eingestuft wurde oder keine Einstufung stattgefunden hat.

Ein weiteres Problem in diesem Zusammenhang ist das semantische Multitasking bzw. die Verflechtung von Themen innerhalb einer Konversation (Walkowska, 2010) Einen Überblick bzw. Verständnis eines Diskurses auf einem Instant Messenger zu erhalten ist in einigen Fällen selbst für die direkten Adressaten nicht einfach.

Bei einer Analyse eines Chats mit 1500 Nachrichten wurde ungefähr ein Drittel der Nachrichten von text-blob-de bewertet (nicht 0). Natürlich kommt es hierbei auch auf die Länge und Art der Nachrichten selbst an. Kurze, unvollständige Nachrichten haben erfahrungsgemäß keine Sentimentanalyse zugelassen.

5.4 Persönliche Erkenntnisse

Die Analyse meiner eigenen Chatverläufe hat mir verschiedene Erkenntnisse gebracht.

Zum Beispiel ist mir aufgefallen, dass ich in den meisten Chats weniger Nachrichten, Emojis und Anhänge versendet habe als meine Chatpartner. Dies hat auch meine persönlichen Annahme bestätigt, da ich generell nicht gerne per WhatsApp Unterhaltungen bzw. Smalltalk halte, sondern eher dies telefonisch oder persönlich tue.

Die Länge von Nachrichten war bei älteren Menschen (meine Eltern, Tanten, Onkeln, etc.) meist länger als die von mir. Ich vermute, dass ein Grund dafür sein kann, dass

diese Generation mit Briefen und später mit Telegrammen/Emails aufgewachsen ist. Bei dieser Art von Kommunikation handelt es sich oft um einzelne Nachrichten bei denen meistens der Adressat und Absender sowie eine Begrüßung, Verabschiedung und die Anliegen der Nachricht ausformuliert werden. Währenddessen werden bei SMS und Instant-Messages, die Nachrichten eher kurzgehalten werden und überflüssige Informationen wie Adressat und Absender weggelassen. Außerdem wird oft ein Anliegen in mehrere zusammengehörende Nachrichten unterteilt und teils von anderen nicht relevanten Nachrichten (z.B. Small-Talk) unterbrochen.

Ebenso konnte ich feststellen, dass die Anzahl an unterschiedlichen Nachrichten bei Kontakten wie meinen Eltern oft höher ausfiel als bei mir. Bei Kontakten, die in meinem Alter sind, konnte ich dies nicht feststellen. Bei Kontakten, die in meinem Alter sind, konnte ich dies nicht beobachten. Die Anzahl an unterschiedlichen Wörtern und Lemmas war oft ähnlich hoch.

An der Verteilung von den Nachrichten im zeitlichen Rahmen, kann man verschiedene Schlüsse, wie in welcher Relation die Konversationspartner stehen, ziehen. Zum Beispiel konnte ich in Chatverläufen mit Kommilitonen oft längere Chatpausen (vor allem in semesterfreien Zeiten) und bei Familie und Freunden keine großen Lücken in dem Chat erkennen. Dies gilt auch für die Verteilung auf die Wochentagen und Uhrzeiten. Mit Arbeits- und Universitätskollegen habe ich weniger Nachrichtenverkehr in der Nacht und am Wochenende gesehen. Mit Freunden wiederum hauptsächlich Nachrichtenverkehr in den Abendstunden dafür kaum am Morgen oder am Vormittag.

Die meistbenutzten Wörter bzw. Lemmas und Stoppwörtern und Emojis zeigt mir eine generelle Gemütslage und den Ton in dem Chat.

Für mich stellt sich auch die Verwendungsanzahl von bestimmten Wörtern (zum Beispiel Anreden, Stoppwörtern, ich und du) als eine interessante Informationsquelle dar. Diese sagt viel über die Art und Weise der Konversation aus und verdeutlicht die Intentionen der einzelnen Absender.

Viele der genannten Features sind auch auf eine linguistische Betrachtungsweise interessant, dazu kommen noch die Anzahl der gezählten Eigennamen und die Anzahl der unterschiedlichen Wortarten.

5.5 Resümee

Allgemein konnte ich viele verschiedene Erkenntnisse aus der Analyse meiner Chatverläufe gewinnen und ich glaube, dass dies auch für mögliche Benutzer gilt.

Abschließend kann ich behaupten, dass mein Projekt zu Ende gebracht und mein mir gesetztes Ziel weitestgehend erreicht ist. Mit dem von mir entwickelten Programm lassen sich WhatsApp Chatverläufe analysieren und veranschaulichen.

Das Repository befindet sich in einem funktionalen Zustand und analysiert WhatsApp Chats auf die unterschiedlichsten Statistiken und Werte. Die Ergebnisse der Analyse werden in ein PDF-Dokument geschrieben. Beim Umsetzen und Testen habe ich einige Erkenntnisse gewonnen.

Zum einen müssen für die Verarbeitung von natürlicher Sprache einige Kriterien erfüllt sein, um gültige Daten zu erhalten und zum anderen, dass WhatsApp Nachrichten diese Kriterien oft nicht erfüllen.

Bei der Gefühlslagen-Bestimmung werden zum Beispiel oft ganze Sätze bzw. Passagen benötigt, um das Sentiment sicher bestimmen zu müssen. Dazu kommt noch, dass manche Modelle keine eindeutigen Ergebnisse liefern (siehe *Abbildung 14*) bzw. einen Satz nur auf Grund eines vorkommenden Wortes negativ oder positiv bewerten. Wiederum liefern Statistiken, die hauptsächlich auf Rechenoperationen beruhen, konstante Ergebnisse.

Probleme wie das Darstellen von Emojis in fpdf oder das richtige Einlesen verschiedener Chat-Formate können mit Behelfslösungen behoben werden.

Die Ausgabe des Programms, also die Visualisierung der Analysedaten, geben die gewonnenen statistischen Daten einer Konversation kompakt wieder. Chatverläufe mit Hunderten Nachrichten aus Jahren von Konversationen können in wenigen Minuten analysiert und dargestellt werden.

Dies gibt einem Nutzer die Möglichkeit, über sein Chatverhalten zu reflektieren und gegebenenfalls Änderungen an der eigenen Ausdrucksweise durchzuführen.

Schlussendlich glaube ich, dass eine solche Software für viele Menschen ein interessantes Tool sein kann und dass manch eine Person auch Spaß daran haben könnte, sein Chatverhalten analysieren zu lassen.

Anhang

Anhangsverzeichnis

Anhang 1: Baum-Darstellung des Repository Aufbaus.....	23
Anhang 2: Terminalausgabe erfolgreiche Analyse	23
Anhang 3: Terminalausgabe Fehlermeldungen.....	24
Anhang 4: Kreisdiagramme Beispiele	24
Anhang 5: Balkendiagramm Beispiel	24
Anhang 6: Zeitstrahl Beispiele	25
Anhang 7: Regex-Muster	25
Anhang 8: Heatmap Beispiel.....	25
Anhang 9: Boxplot & Violin-Plot Beispiel.....	26
Anhang 10: Boxplot mit und ohne Ausreißern	26
Anhang 11: WordCloud Beispiel	27
Anhang 12: Tabelle Beispiel	27
Anhang 14: Auszug Sentiment-Nachrichten	28

Anhang 1: Baum-Darstellung des Repository Aufbaus

```
$ tree WhatsAppReports/
```

```
WhatsAppReports/
├── data
│   ├── output
│   │   ├── images
│   │   │   ├── generalpage
│   │   │   ├── ...
│   │   │   └── senderpages
│   │   │       ├── ...
│   │   └── Report.pdf
│   └── sample_chat.txt
├── database
│   ├── constants.py
│   └── variables.py
├── utils
│   ├── analyzer.py
│   ├── converter.py
│   ├── getter.py
│   ├── helper.py
│   ├── outputter.py
│   └── plotter.py
├── LICENSE
├── README.md
├── main.py
└── requirements.txt
```

```
7 directories, 14 files
```

Quelle: Eigendarstellung

Anhang 2: Terminalausgabe erfolgreiche Analyse

```
Please enter the path to the chat file
Only .txt or .zip files are supported
Enter 'sample' if you do not have a file at hand
Enter the path here: sample

Analyzing file @ 'data/sample_chat.txt'

Converting chat to a pandas.DataFrame took 1.175931 sec
Analyzing chat for sender 1 / 2 took 0.417198 sec
Analyzing chat for sender 2 / 2 took 0.14808 sec
Calculating remaining statistics took 0.408721 sec
Visualizing data for sender 1 / 2 took 3.958665 sec
Visualizing data for sender 2 / 2 took 3.908745 sec
Visualizing remaining data took 5.897048 sec
Finishing final PDF Report took 7.054167 sec

✅ Success: Analysis finished ✅
Analyzing took 22.96862 seconds in total.
The PDF Report is located here: '.../WhatsAppReports/data/output/Report.pdf'
```

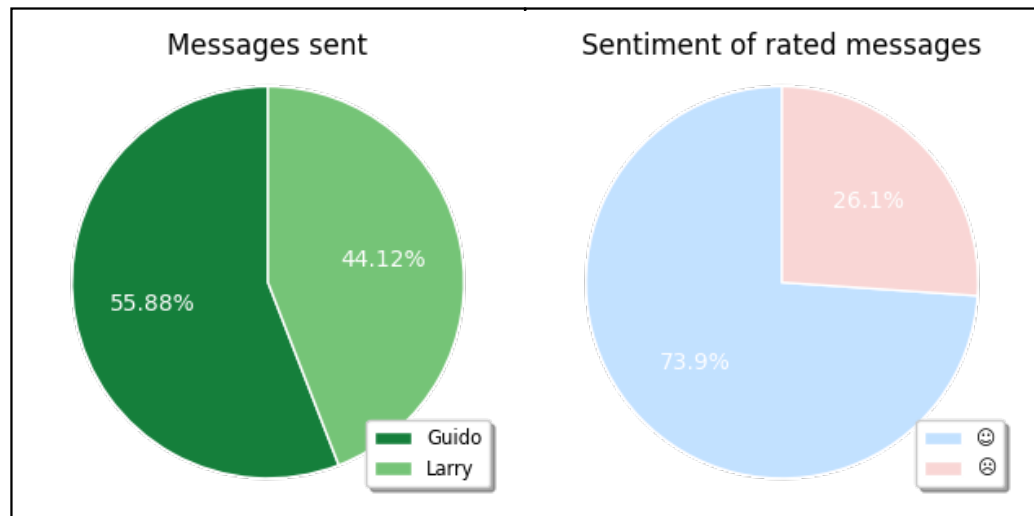
Quelle: Eigendarstellung

Anhang 3: Terminalausgabe Fehlermeldungen

<pre>Please enter the path to the chat file Only .txt or .zip files are supported Enter 'sample' if you do not have a file at hand Enter the path here: there_is_no_file.txt Analyzing file @ 'there_is_no_file.txt' Error File not found Check the path to the file</pre>	<pre>Please enter the path to the chat file Only .txt or .zip files are supported Enter 'sample' if you do not have a file at hand Enter the path here: not_a_chat.txt Analyzing file @ 'not_a_chat.txt' Error No messages found in that file. Did you select the correct file?</pre>	<pre>Please enter the path to the chat file Only .txt or .zip files are supported Enter 'sample' if you do not have a file at hand Enter the path here: wrong_file_type.csv Analyzing file @ 'wrong_file_type.csv' Error Wrong file type Only .txt or .zip files are supported</pre>
--	---	--

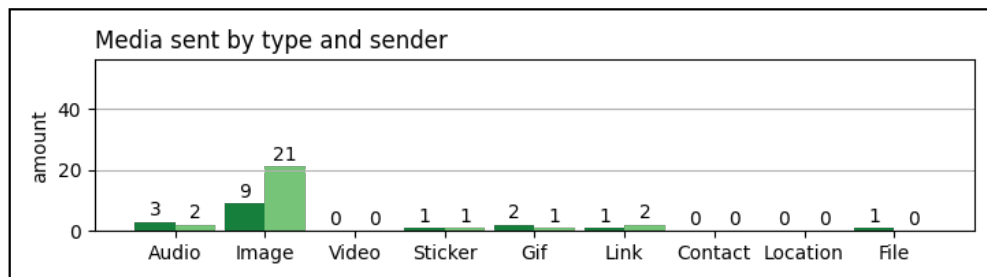
Quelle: Eigendarstellung

Anhang 4: Kreisdiagramme Beispiele



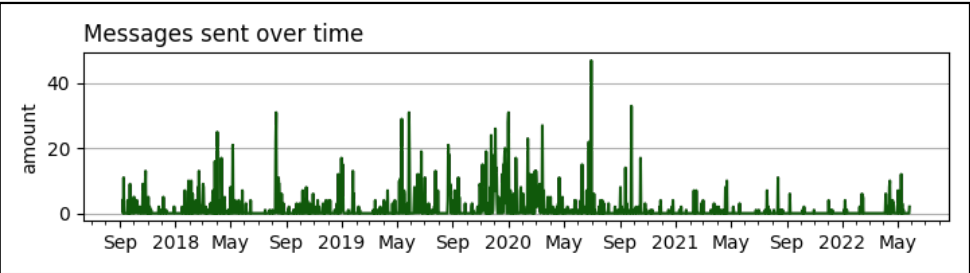
Quelle: Eigendarstellung

Anhang 5: Balkendiagramm Beispiel



Quelle: Eigendarstellung

Anhang 6: Zeitstrahl Beispiele



Quelle: Eigendarstellung

Anhang 7: Regex-Muster

Group1 (Date)

Group2 (Time)

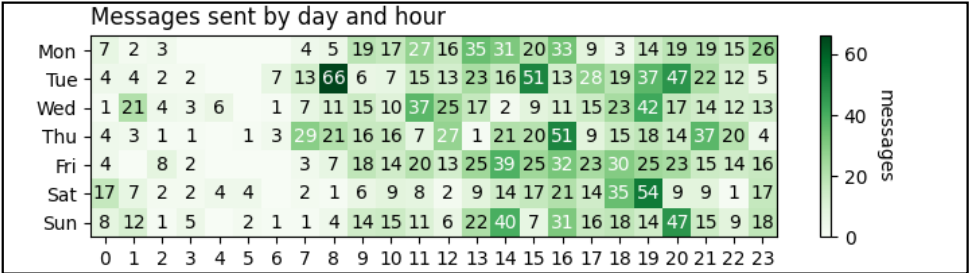
Group3 (Sender)

Group4 (Message)

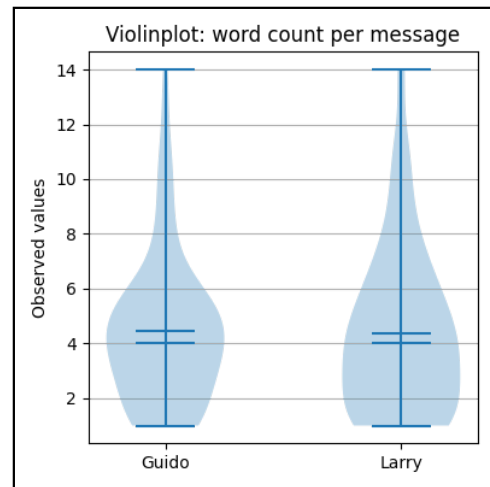
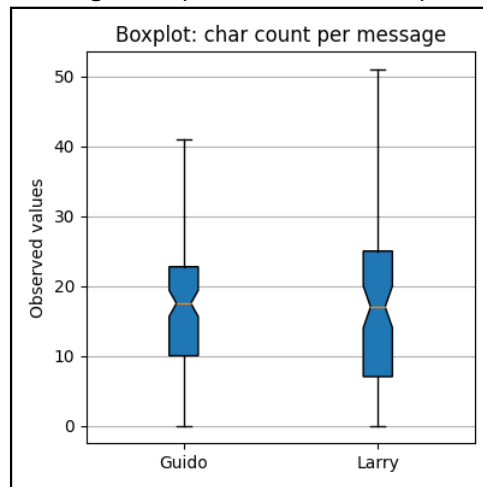
```
^\u200E?[(\d{1,2}.\d{1,2}.\d{2}), (\d{2}:\d{2}:\d{2})\] \u200E?(\b[\w ]*\b): (.*?)$
^.\? ?\[(\d{1,2}.*), (\d{1,2}.*)\] (.\?*\u202A?): (\u200E?.*)$
```

Quelle: Eigendarstellung

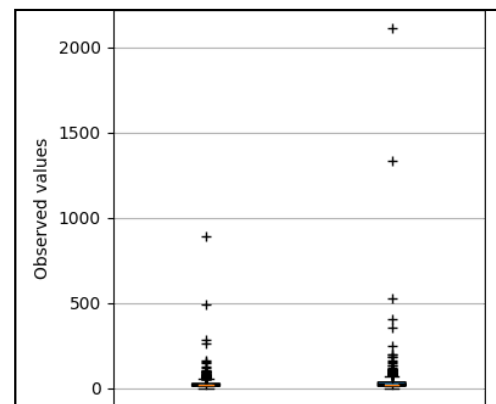
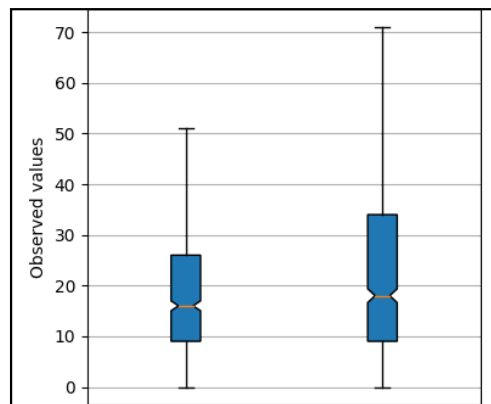
Anhang 8: Heatmap Beispiel



Quelle: Eigendarstellung

Anhang 9: Boxplot & Violin-Plot Beispiel

Quelle: Eigendarstellung

Anhang 10: Boxplot mit und ohne Ausreißer

Quelle: Eigendarstellung

Anhang 11: WordCloud Beispiel



Quelle: Eigendarstellung

Anhang 12: Tabelle Beispiel

Top 20 Words	Frequency	Top 20 Emojis	Frequency
ich	219	kissing heart	132
du	149	clap	38
die	102	heart eyes	23
in	94	hearts	19
und	89	heart	11
das	86	smiling face with three hearts	9
ist	81	pray	7
nicht	79	tada	6
ok	65	rofl	6

Quelle: Eigendarstellung

Anhang 13: Auszug Sentiment-Nachrichten

Die Verbindung ist schlecht	-1	Ne schon gut	1
Hab alles fertig bis auf das Fazit	-1	War voll gut	1
Hab den bei Peek & Kloppenburg gekauft	-0.7	Super ich freue mich	1
Hab mir einen Pulli gekauft wie ist er?	-0.7	Auchso ja klar	1
Hast du in Berlin was gekauft?	-0.7	Hey guten Morgen	1
Ich bin jetzt fertig	-1	Ok sei bitte pünktlich in der Schule	1
Ich bin sehr enttäuscht und sehr sauer auf dich.	-1	Ich habe gleich 2 Stunden frei	0.7
Ich hab euch auch vermisst	-0.7	Hier ist direkt Lidl	0.7
Ich muss dann Physiotherapie absagen	-1	Bist du wach?	0.7

Quelle: Eigendarstellung

Literaturverzeichnis

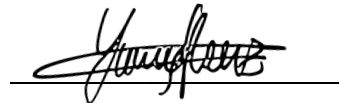
- Adobe Systems Incorporated. (2016). *PDF Reference fifth edition: Adobe® Portable Document Format Version 1.6*. Internet-Archive. <https://web.archive.org/web/20160304052118/http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf>
- Dean, B. (2022, January 5). *WhatsApp 2022 User Statistics*. Backlinko. <https://backlinko.com/whatsapp-users>
- Driscoll, D. (2002). The Ubercool morphology of internet gamers: A linguistic analysis. *Journal for the Human Sciences*, 1. <http://www.kon.org/urc/driscoll.html>
- Galov, N. (2022, April 6). *25 WhatsApp Statistics to Show You What's up in 2022*. WebTribunal. <https://webtribunal.net/blog/whatsapp-statistics/>
- Grant, R. (2019). Pretty Persuasion: The Advantages of Data Visualisation. *Impact*, 2019(2), 19–23. <https://doi.org/10.1080/2058802x.2019.1633814>
- Hagedorn, S., Kläbe, S., & Sattler, K. (2021). Putting Pandas in a Box. *Conference on Innovative Data Systems Research*. http://cidrdb.org/cidr2021/papers/cidr2021_paper07.pdf
- Hunter, J., Dale, D., Firing, E., Droettboom, M., & matplotlib development team. (2022, September 16). *Matplotlib Release 3.6.0*. Matplotlib. <https://matplotlib.org/3.6.0/Matplotlib.pdf>
- LEWIS, C., & FABOS, B. (2005). Instant messaging, literacies, and social identities. *Reading Research Quarterly*, 40(4), 470–501. <https://doi.org/10.1598/rrq.40.4.5>
- Reuter, K. (n.d.). *Der Violin-Plot*. TQU Verlag. Retrieved October 1, 2022, from <https://www.tqu-group.com/we-dokumente/informieren/QualityAppsTexte/ViolinPlotReuter.pdf>
- Singh, M. (2020, October 30). *WhatsApp is now delivering roughly 100 billion messages a day*. TechCrunch. <https://techcrunch.com/2020/10/29/whatsapp-is-now-delivering-roughly-100-billion-messages-a-day/?guccounter=1>
- Varnhagen, C. K., McFall, G. P., Pugh, N., Routledge, L., Sumida-MacDonald, H., & Kwong, T. E. (2009). lol: new language and spelling in instant messaging. *Reading and Writing*, 23(6), 719–733. <https://doi.org/10.1007/s11145-009-9181-y>
- Walkowska, J. (2010). An NLP-Oriented Analysis of the Instant Messaging Discourse. *Text, Speech and Dialogue*, 576–583. https://doi.org/10.1007/978-3-642-15760-8_73

Ehrenwörtliche Erklärung

Hiermit erkläre ich, Yunus Oscar Renz, dass die vorliegende Projektarbeit mit dem Titel „Programm zur Analyse von WhatsApp-Chatdaten mit Hilfe verschiedener NLP-Tools“ und das dazugehörige Programm von mir selbständig verfasst bzw. entwickelt ist und ich diese zuvor an keiner anderen Hochschule und in keinem anderen Studiengang als Prüfungsleistung eingereicht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen der Projektarbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht.

Düsseldorf, den 21. Oktober 2022

Ort, Datum

A handwritten signature in black ink, appearing to read 'Yunus Oscar Renz', is written over a horizontal line.

Unterschrift