FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Political Polarity in US Twitter

Seminar BigData

at the
Friedrisch Schiller University of Jena
Faculty of Mathematics and Computer Science
Graduate Degree Computer Science

submitted to
Prof. Dr. Bücker,
Dr. rer. nat. Bosse and
Herrn Schoder

submitted by
Kenny Gozali,
Chris Gerlach and
Walter Ehrenberger

Jena, January 29, 2023

Abstract

In der vorliegenden Arbeit wird eine Sentimentalitätsanalyse von US amerikanischen Politikern aus dem *House of Representatives* behandelt. Dazu wurden Daten von Twitter der letzten 12 Jahre zu den genannten Repräsentanten *gescrapt* und mithilfe des Big Data Frameworks Spark verarbeitet. Ziel der Sentimentalitätsanalyse war es, Unterschiede der beiden Parteien (Republikaner und Demokraten) zu bestimmten politischen und auch allgemeinen Themen herauszufiltern. Jedoch haben sich in den gegebenen Daten weniger Diskrepanzen zwischen den beiden Parteien erkennen lassen, als zu Beginn erwartet, wie im Laufe dieser Arbeit deutlich wird.

Contents

List of Figures

1. Introduction

Als mächtigste Weltmacht beeinflussen die Vereinigten Staaten nahezu jeden Teil des Globus. Mitunter deshalb und aufgrund der enormen Präsenz in den Medien sowie des Einflusses auf diese fallen Diskrepanzen in der Bevölkerung schneller auf als in anderen Ländern. Aufgrund dieser Stellung wirkt sich die dortige Sentimentalität somit auch auf das Leben in anderen Ländern aus. Der Kapitolsanschlag sowie die Black Lives Matter Proteste der letzten Jahre sind ein Zeichen für die zunehmende Polarität und Unzufriedenheit in der Bevölkerung, wie sich auch in folgender Grafik erkennen lässt 1.1.

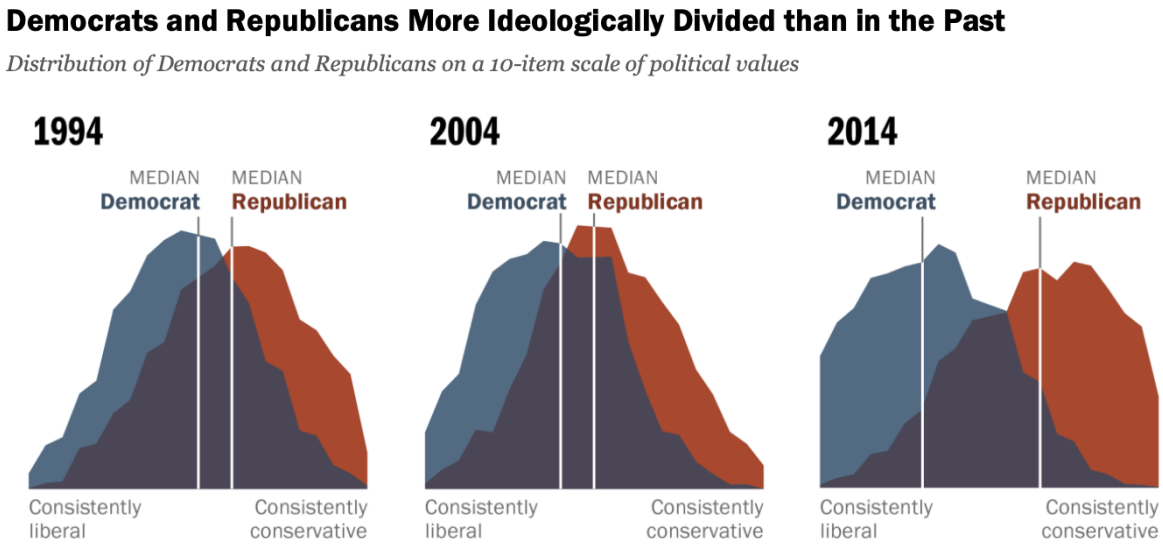

Figure 1.1.: Entwicklung der Polarität politisch engagierter Amerikaner. Basierend auf 10 politischen Metriken werden Demokraten (blau) und Republikaner (rot) hier verglichen. Wie zu erkennen bewegen sich die zu Beginn teils noch überlappenden Ideologien in den letzten 10 Jahren auseinander [Dim14].

Aufgabe dieser Arbeit ist es nicht, sich mit den komplexen und vielschichtigen Hintergründen für diese Entwicklung auseinanderzusetzen. Vielmehr wird hiermit ver-

sucht, eben diese Polarität in den oberen Reihen der amerikanischen Politik genauer zu analysieren.

Unsere Zielsetzung bestand darin, mit den Tweets der letzten 12 Jahre von 420 Politikern des Repräsentantenhauses eine Sentimentalitätsanalyse durchzuführen. Dabei handelt es sich um ein Mittel der natürlichen Sprachverarbeitung, bei dem die Ansicht beziehungsweise Gefühlslage eines Textes quantifiziert wird.

2. Background

2.1. Used Libraries

In diesem Kapitel sollen die verwendeten Bibliotheken und der Grund für ihre Verwendung genauer beleuchtet werden. Damit Daten generiert werden konnten, wurde die Bibliothek *GetOldTweets3* verwendet. Mit einer öffentlich zugänglichen Userliste für Politiker aus Amerika wurden mithife dieser Packages die Daten erhoben. Zur Weiterverarbeitung der Daten wurden *NLTK* und *TextBlob* genutzt. Beides sind Tools für die Verarbeitung von Sprache. Um eine Analyse über die verarbeiteten Ausgaben durch ein MapReduce laufen zu lassen, wurde schließlich Spark verwendet, um eine Zeit effiziente Verarbeitung zu gewährleisten.

2.1.1. GetOldTweets3-Pakage

GetOldTweets3 ist ein kostenloses Python 3 Package mit welchem Twitterdaten ohne API-Schlüssel abgerufen werden können. Mit *GetOldTweets3* können Tweets mit einer Vielzahl von Suchparametern wie Start-/Enddatum, Benutzername(n), Textabfrage und Referenzortbereich extrahiert werden. Außerdem können Tweet-Attribute die einbezogen werden sollen berücksichtigt werden. Beispielsattribute sind Folgende: Nutzername, Tweettext, Datum, Retweets und Hashtags [Yos20]. Die offizielle API von Twitter hat ein beschränktes Zeitfenster, weshalb keine Tweets älter als eine Woche abgerufen werden können. Es gibt einige Tools, die Zugang zu älteren Tweets anbieten, diese sind jedoch meistens kostenpflichtig. Das Forschungsteam hat nach einem anderen Tool gesucht um diese Aufgaben zu übernehmen, wodurch die Wahl auf das Package *GetOldTweets3* gefallen ist [Hen19].

Die Analyse des Codes von *GetOldTweets3* und die Funktionsweise des Searchthrough

Browsers von Twitter zeigt, wie das Package auch an alte Tweets kommt. Wenn auf Twitterseiten oder User gesucht werden, startet ein Scroll-Loader. Das heißt, beim scrollen nach unten tauchen immer mehr Tweets zu den jeweiligen Suchparametern auf. Diese Tweets bekommen sie durch Abfragen an einen JSON-Provider welcher den Searchthrough Browser von Twitter imitiert, um den Scroll-Loader zu starten und zieht sich dann anhand der Abfragen an einen JSON-Provider die JSON-Datei und gibt diese decodiert zurück, um somit alle Tweets anhand der oben gegebenen Parameter herauszufiltern. Dies kann man in dem GitHub-Repository gut nachvollziehen [Hen18]. Somit ist es möglich, sowohl aktuelle als auch sehr alte Tweets zu scrapen.

```
#!/bin/bash
# cat user_list.csv

while IFS="," read -r rec_column1 rec_column2 rec_column3 rec_column4 rec_column5
do
  echo "Writing to data/$rec_column3"
  python GetOldTweets3/cli.py --username $rec_column3 > data/$rec_column3

done < <(tail -n +2 user_list.csv)
```

Figure 2.1.: Code Beispiel für das Scrapen der Tweets
Ist eine Python Bibliothek mit der Twitter Daten durch den Scroll-Loader des Searchthrough Browsers von Twitter als JSON-Datei abgerufen werden können.

So kann durch eine paar Zeilen Code, wie in der Abbildung 2.1 zu erkennen ist, eine Bash-Datei erstellt werden, durch welche die Daten gesucht und abgespeichert werden. Das Scraping kann durch die Größe der JSON-Datei einige Zeit in Anspruch nehmen. 2 Millionen Tweets haben insgesamt eine Laufzeit von ca. 35 Stunden verbucht.

2.1.2. NLTK-Natural language Toolkit

NLTK ist ein Pythonpackage für die Arbeit mit menschlichen Sprachdaten. Es bietet einfach zu bedienende Schnittstellen zu über 50 Korpen und lexikalischen Ressourcen wie WordNet, zusammen mit einer Reihe von Textverarbeitungsbibliotheken für Klassifizierung, Tokenisierung, Stemming, Tagging, Parsing und semantischen Schlussfolgerungen sowie Wrapper für industrielle NLP-Bibliotheken und ein aktives Diskussionsforum [NT23].

Aus diesem Grund bietet *NLTK* sehr viele Möglichkeiten zur Vorverarbeitung und einer Analyse. Benötigt aber auch einen gewissen Zeitrahmen zur Einarbeitung in die Analysen. Aus diesem Grund hat sich das Forscherteam dafür entschieden, NLTK nur zur Vorverarbeitung zu nutzen und TextBlob für die semantische Analyse zu nutzen. Warum sich für TextBlob entschieden wurde, wird genauer in 2.1.3 besprochen. So verwenden wir den Wordkorpus von NLTK für englische Stoppworte, da diese nicht Teil der Analyse sein sollen.

Für eine individuelle und sehr ausführliche semantische Analyse bietet NLTK sehr viele Möglichkeiten durch die Interaktion mit verschiedenen Packages in Python, was aber den oben genannten Zeitrahmen benötigt, zum Einarbeiten. Der Vorteil von NLTK gegenüber TextBlob sind genau diese Interaktionen mit anderen Packages. Für größere Projekte, bei denen man die semantische Analyse auch individuell anpassen möchte, sollte man die NLTK Bibliothek benutzen. NLTK ermöglicht es durch verschiedene Vorverarbeitungsschritte, welche in der Bibliothek eingebaut sind, eine individuelle Pipeline und Analyse zu erstellen.

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Figure 2.2.: Code Beispiel für das Arbeiten mit NLTK
Tokenisierung und Tagging von Texten mit NLTK

2.1.3. TextBlob

TextBlob ist eine Python Bibliothek für die Pythonversionen zwei und drei. Diese Packages arbeiten, ähnlich wie *NLTK*, mit verschiedenen Packages, welche in Python schon

verfügbar sind. In *TextBlob* sind zwei verschiedene semantische Analysen vorhanden. Zum einen die Patternanalyse, welche die Patternbibliothek in Python nutzt, und die Naive-Bayes-Analyse [Lor20a]. Hier stellt *NLTK* zum Beispiel mehr zur Verfügung, aber benötigt damit auch mehr Einarbeitungszeit. Damit bietet *TextBlob* eine besser Übersicht, weshalb sich das Forschungsteam aufgrund der wenigen Zeit für diese Bibliothek entschieden hat.

Ein weiter Grund, warum sich schlussendlich für diese Bibliothek entschieden wurde, sind auch die zwei Outputwerte Polarität und Subjektivität welche eine gute Anwendungsmöglichkeit darstellen [Lor20b]. Die Patternanalyse von TextBlob gibt die Polarität in einem Intervall von [-1, 1] und die Subjektivität im Rahmen von [0, 1] zurück. Ist der Wert bei der Polarität näher an der -1 als an der 1, dann zeigt es einen negative Emotion. Im umgekehrten Fall ist es eine positive Emotion.

Bei dem Wert der Subjektivität beschreibt ein Wert, der gegen null tendiert, einen Fakt oder Faktenwissen und eine Tendenz gegen eins entspricht einer stärkeren subjektive Meinung mit rein [Lor17a] [Lor17b]. Bei der Patternanalyse geht es darum, ein Muster bei negativen und positiven Aussagen zu erkennen und dies dann auf neue Testdaten oder unbekannte Daten anzuwenden. Dabei spielt sowohl die Syntax als auch die Semantik und die Wortwahl eine bedeutende Rolle [boe18]. Mit etwas mehr Zeit hätte man auch noch die Analyse des Naive-Bayes in einem MapReduce verwenden können.

2.1.4. Spark

Spark ist ein Big Data Framework zur Verarbeitung, Filterung und Analyse von großen Datenmengen. Diese Bibliothek vereinfacht die Anwendung eines MapReduce, indem es verschiedene Funktionsweisen und Tools dafür anbietet. So begrenzt sich der Programmcode auf die wesentlichen Funktionen eines MapReduce und verschafft dadurch eine guet Übersicht über den Code. Des Weiteren stellt Spark verschiedene Datenstrukturen zur Verfügung, um die Arbeit mit großen Datenmengen zu erleichtern. Dazu gehört zum Beispiel das Resilient Distributed Datasets (RDD). Ein weiter Vorteil, den Spark bietet, ist die hohe Verarbeitungsgeschwindigkeit der Daten. Aus diesen Gründen hat sich das Forscherteam für das Mapreduce entschieden, welches in dem Forschungsprojekt Anwendung finden soll, diese Bibliothek zu verwenden, um eine einfach und zeit effiziente Verarbeitung der Twitterdaten zu haben.

2.2. Data Sanitization

In diesem Kapitel wird näher auf den Programmcode des vorliegenden Forschungsprojektes eingegangen und erklärt, was genau in der Datensammlung und Datenverarbeitung gemacht wurde. Zu Beginn wurden die Daten wie in Punkt *2.1.1 GetOldTweets3* beschrieben, mit der Bibliothek gescrapt und als CSV-Datei abgespeichert. Wie das Scraping in der Bibliothek genau funktioniert, ist ebenfalls unter dem Punkt *2.1.1. GetOldTweets3* beschrieben. Um die gespeicherten Tweets für das MapReduce vor zu verarbeiten, wurden die zur Verfügung stehenden Bibliotheken *NLTK* und *cleantext*.

```
start = time.time()
directory = '/data/'
directory_sanitized = '/data_sanitized/'

path = os.getcwd()+directory
user_file_names = sorted_alphanumeric(os.listdir(path))

current_user_sanitized = []

for user_file in user_file_names:
    with open(path + user_file, 'r') as file:
        for line in file:
            for word in line.split("\n\n"):
                current_user_sanitized.append(tweetDecomposer(word))
        save_sanitized_file(user_file, directory_sanitized,
                            current_user_sanitized)
        current_user_sanitized = []

print(">>>> JOB DONE, it took " + str(round(time.time() - start, 2)) + " seconds")
```

Figure 2.3.: Sanierungs-For-Schleife der Daten

Die Variable "directory" und "directory_ sanitized" geben den Pfad an, in welchem Ordner die Daten gespeichert werden sollen. Mit der Bibliothek *os* von Python können über "os.listdir(pfad)" alle Dateinamen innerhalb dieses Ordners eingelesen werden. Mit der Variable "user_ file_ names" wird eine alphanumerisch sortierte Liste der Usernamen der Politiker zurückgegeben, welche dann über eine For-Schleife durchgegangen werden kann, da der Name der CSV-Dateien mit folgendem Muster dem Usernamen entspricht, "Name_ D" oder "Name_ R". **D** steht für demokratisch und **R** für republikanisch. In dieser Datensanierungsschleife wird die Funktion *tweetDecomposer* verwendet. Diese Funktion übernimmt in der vorliegenden Datensanierung die Hauptaufgabe.

Mit der Bibliothek *cleantext* wurden *Emojis* aus dem Text entfernt, wie man in Abbildung A.1 in den ersten Zeilen der Funktion sehen kann. Dann werden alle Worte innerhalb eines Tweets kleingeschrieben und aufgetrennt, damit die ID, die Zeitzone und der Username aus dem Tweet entfernt werden können. Mit *NLTK* werden dann die Stoppworte durch ein *join* aus den Tweets entfernt, sodass man zu den letzten Datensanierungsschritten kommen kann.

Da die Annotations und Hashtags gespeichert werden sollen, wurde, wie in Abbildung B.1 zu sehen ist, eine extra For-Schleife eingesetzt. Die For-Schleife läuft über den gesamten Input des Tweets, dazu zählen Datum, Zeit, Username, Hashtag und Emoji. Als Erstes wird überprüft, ob es sich um eine URL handelt oder nicht. Tritt der Fall ein, dass es eine URL ist, wird diese einfach übersprungen und nicht mit abgespeichert. Die Annotations können durch ein "@" erkannt werden, während die Hashtags mit einem "# " erkannt werden. Beide Erkennungsmarker werden nicht mit abgespeichert. Der restliche Inhalt des Hastags und der Annotation werden in einer extra Liste gespeichert.

Als Letztes wurden alle Sonderzeichen wie Punkte, Kommas und andere Zeichen entfernt. Die Funktion gibt dann alle interessanten Daten für die Analyse zurück. Zum Schluss werden diese Daten in einer CSV-Datei gespeichert. Als nächster Schritt folgt die Hautanalyse unseres Projektes in 2.3.

2.3. MapReduce

Folgendes Kapitel geht näher auf die MapReduce Funktionalität dieses Projektes ein. Die sanierten Tweets wurden jeweils im CSV Format in einer Datei pro Politiker*in exportiert. Um auf die einzelnen Spalten zuzugreifen, wird eine Spalte über das Trennzeichen getrennt und auf eine Liste gemappt, wovon die benötigten Spalten in ein Tupel *gewrapt* werden. Mithilfe der Filterfunktionen von *Spark* werden Tweets extrahiert, die in einem gewünschten Zeitraum liegen, Keywords oder bestimmte Hashtags enthalten. Der Output wird nach User und Partei sortiert, je nach Query reduziert und in einer CSV Datei für die Analyse exportiert.

Naivere Funktionen befassen sich mit einer Reduktion auf die Anzahl der Tweets per User oder per angehöriger Partei. Dies funktioniert für eine Reduktion auf die Anzahl der Annotationen (im Tweet referenzierte User) und Anzahl der Hashtags gleich.

Schließlich wurde die durchschnittliche Tweetlänge mit einer Reduktion auf die Summe der Tweetlänge durch ein Teilen mit der Tweetanzahl pro User und Partei gelöst.

Die gleiche Funktionalität wurde ebenso für das Berechnen der durchschnittlichen Polarität und Subjektivität mit der Hilfe von *TextBlob* gelöst. Um die Sentimentalitätsanalyse mächtiger zu machen, ist es möglich, die Sentimentalität eines Tweets auch für bestimmte Hashtags oder oben genannte Keywords zu berechnen. Unter anderem kann so eine Liste von Ländern hergenommen werden, um beispielsweise die Meinung gewisser Politiker*innen zu bestimmten Ländern zu quantifizieren. Allerdings muss angemerkt werden, dass die Analyse dementsprechend stark abhängig von der genannten Bibliothek ist.

3. Data Analysis

3.1. Analysis And visualisation

Analysis and visualisation are an important aspect in conveying information. In this project, we analysed the data that was processed using the Spark framework. It is very important that we use a big data framework for big data projects. One of the main reason, is that since some data that were collected are bigger than the capacity of the RAM itself, the only way to work on it is to use a clustered system. Spark is one of such framework that take advantage of Hadoop Distributed File System.

There are various reason why visualizing data is important. Visualisation is an important method of conveying information and it lets the viewer have a more holistic and detailed view of the data. It is not uncommon to find deep and new insight of the data through visualisation. It is one of the main method to relay information to all stakeholders, in order to perform a Data-Driven Decision Making (DDDM).

Here we have done plenty visualisation, in term of presenting informative graphics towards the reader. We have done TextCloud visualisation and also scatter plot visualisation for the semantic analysis. The data that was procured from the previous steps were then accessed and prepared for the various visualisations using Pandas framework.

3.1.1. Word Analysis and Visualisation

TextCloud is a very versatile method of presenting information. The group of words are easily read and the main idea of the group of words are relatively comprehensible without deeper analysis. The ability to obtain the big picture efficiently when presenting information to a massive audience is crucial.

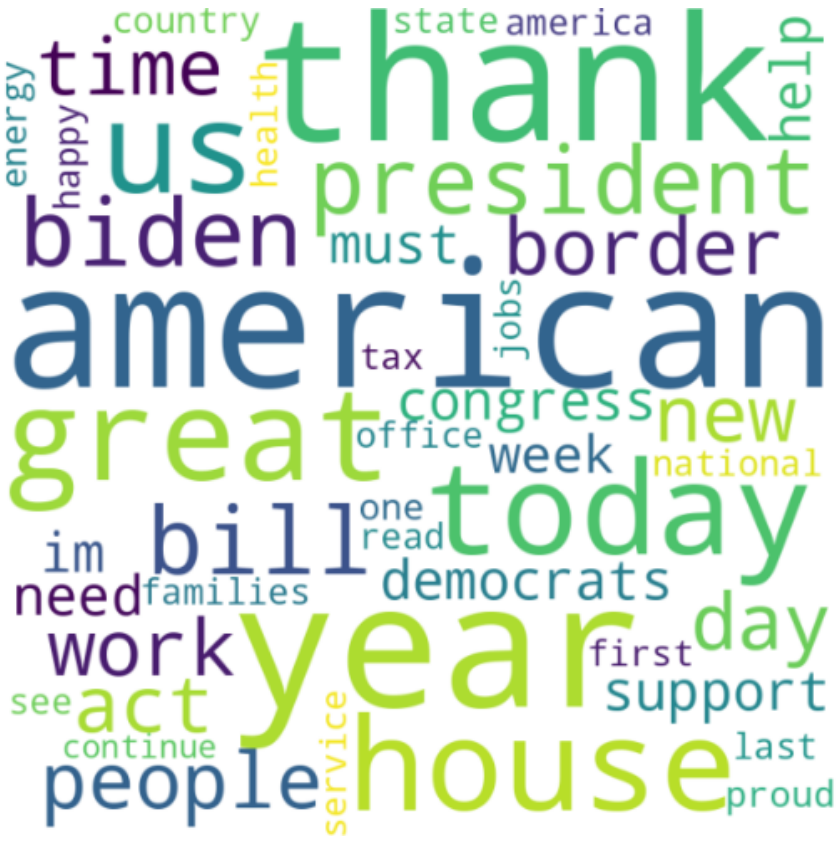

Figure 3.1.: Republican Top 50 most used words

The two American political party, The Republican Party and The Democrat Party are two parties that are relatively known for polarized judgement on various topics. In this project, we try to check the most used words by the two parties in Twitter and then see what are the general big ideas that the two parties are most connected to.

In figure 3.1 and figure 3.2 we are able to see the top 50 most used words for the respective parties. It is interesting to note, that both parties mentions the other parties in their tweets more often than their own. Both the Republican and Democrat party used many positive words, such as people and family.

Here in 3.3 and 3.4, we can see it more clearly, the top 10 most used words by each parties. However in both figure, we are not able to differentiate the word us or U.S. (abbreviation for the United State), since in the preprocessing part of the pipeline, we made the words case-insensitive. These two are plotted and prepared using Pandas and *Matplotlib*.

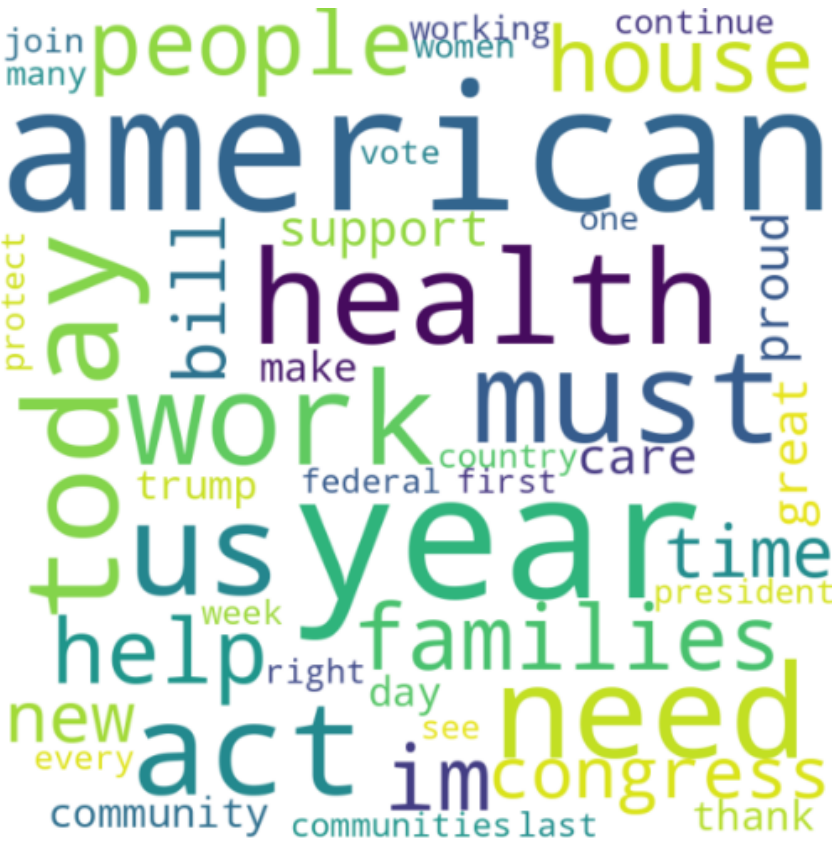

Figure 3.2.: Democrat Top 50 most used words

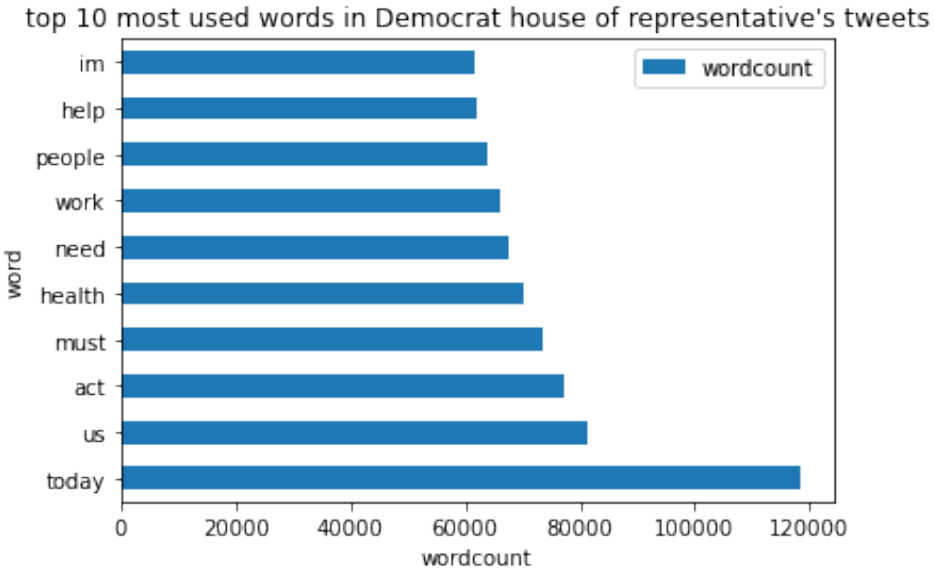

Figure 3.3.: Top 10 most used words by Democrats

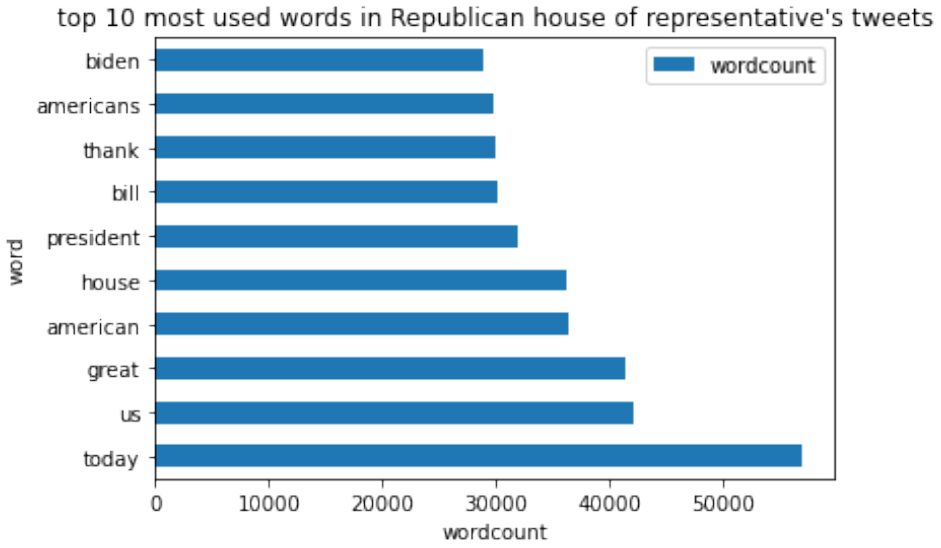

Figure 3.4.: Top 10 most used words by Republicans

3.1.2. Sentiment Analysis

In the project we have done sentiment analysis of the tweets using the library *TextBlob*. The visualisation is made through scatter plots. The X-axis represents the polarity and the Y-Axis represents the subjectivity. The polarity explains the sentiment of the party towards general or a certain topic, whereas the subjectivity explains how subjective or objective the tweets are. Polarity of 1 expresses positiveness and -1 negativeness. Subjectivity of 1 means that the tweet was really subjective and 0 means that the tweet is really based on the truth (according to our *TextBlob* library).

In 3.5, each of the dots represent a member of the House of Representatives. From this figure, it is seen that the polarity tends to be positive. It can be concluded that from this library the sentiment towards Germany is relatively positive. However, the result is bounded by the library *TextBlob* and does not represent perfectly the real sentiment that the party members carry in their tweets. More figures can be seen in the Appendix.

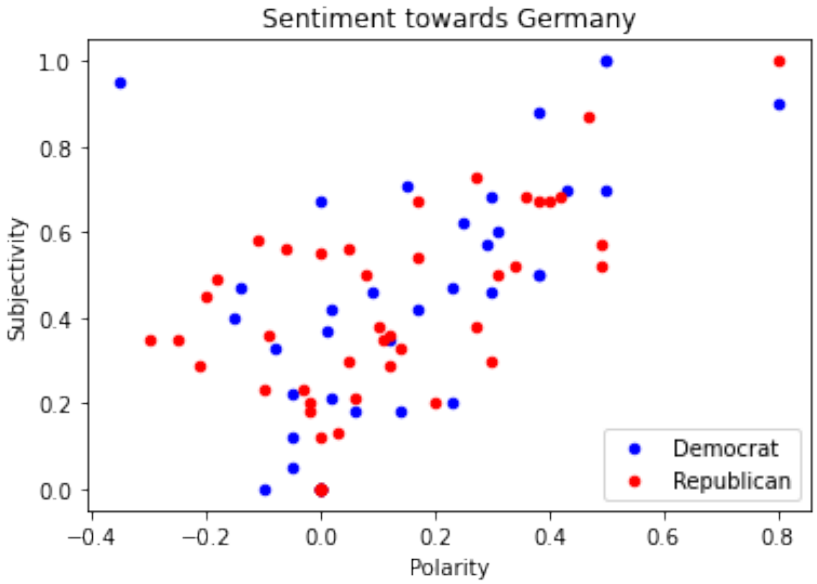

Figure 3.5.: Sentiment towards Germany

4. Discussion

Sentimentalitätsanalysen sind ein mächtiges Mittel, um große Mengen an Text zu quantifizieren und zu kategorisieren. Diese werden von vielen Instituten verwendet um beispielsweise das politische Klima zu Themen wie Pandemien, der Klimaerwärmung oder Krieg zu analysieren [Efu23]. Ergebnisse dieser Arbeit sind allerdings kritisch zu bewerten. Besonders das Berechnen der Sentimentalität ist in dieser Analyse abhängig von *TextBlob*. Durch ein fehlendes Validierungsdatenset sowie fehlende Labels, die Aufschluss über die Grundwahrheit der verwendeten Daten gegeben, ist es schwer, die Genauigkeit der Analyse festzustellen. Auch in Betracht dessen, das nicht klar ist mit welchen Daten die Bibliothek trainiert wurde. Shahul ES. hat in seinem Blog *Sentiment Analysis in Python: TextBlob vs Vader Sentiment vs Flair vs Building It From Scratch* versucht TextBlob und zwei andere Sprachverarbeitungsbibliotheken zu vergleichen und ist für die hier verwendete Biblithek auf einen Genauigket von 56% gekommen [ES23].

Weiterhin lassen die Ergebnisse, die zu Beginn gegebene Hypothese von Michael Dimock et. al. aus dem Paper *Political Polarization in the American Public* von 2014 nicht bekräftigen [Dim14]. Dies liegt zum einen an der verwendeten Stichprobe. Während in dem Paper Amerikaner*innen aus jedem Metier befragt wurden, befasst sich diese Analyse aufgrund von Datenbeschaffungsproblemen ausschließlich mit Politikern*innen. Abgesehen davon sind in dieser Stichprobe wesentlich weniger Menschen vertreten was die Analyse weiterhin verzerrt. Zum anderen werden hier weniger Metriken verglichen, lediglich die zu Beginn kritisch zu bewertende Subjektivität und Polarität aus der Sprachverarbeitungsbibliothek.

In Anbetracht dessen, liefert diese Arbeit ein grundlegendes Werkzeug um Daten aus Twitter zu verarbeiten. In einem weitern Verlauf wären mehr Daten von mehreren unterschiedlichen Usern aus den Vereinigten Staaten eine Möglichkeit um diese Analyse zu erweitern. Dafür sind lediglich die Twitternamen von diesen Usern erforderlich welche ebenfalls durch Scraping erlangt werden können. Zu beachten ist die moralische

Fragwürdigkeit dieses Prozess weswegen unter anderem vorerst darauf verzichtet wurde. Schließlich kann die eigentliche Verarbeitung der Sentimentalität durch die die implementierte Schnittstelle leicht ausgetauscht werden, um so ebenfalls eine genauere oder zumindest andere Analyse im Vergleich durchzuführen.

A. Code Example Sanitization 1

```
def tweetDecomposer(tweet):

    # Stop on last line
    if tweet.find("No more data. finished scraping!!") == 0:
        return

    # Save, then remove emojis
    # emojis = adv.extract_emoji(tweet)
    tweet = clean(tweet, no_emoji=True)

    # Seperate by word and stop on too few lines
    tweet = tweet.lower()
    tweetWords = tweet.split()
    if len(tweetWords) < 4:
        return

    # remove tweet ID
    tweetWords = tweetWords[1:]
    date = tweetWords[0]
    time = tweetWords[1]
    timezone = tweetWords[2]
    # remove time/timezone and tweet username
    tweetWords = tweetWords[4:]

    mentions = []
    hashtags = []
    text = ""

    # removing stopwords
    tweetWords = " ".join([word for word in tweetWords
                           if word not in STOP_WORDS]).split()
```

Figure A.1.: Ausschnitt eins der Sanierungsfunktion der Daten

B. Code Example Sanitization 2

```
    for tweetWord in tweetWords:

        if remove_url(tweetWord):
            continue

        # Annotations
        if tweetWord[0] == "@":
            if tweetWord[1:] == " ":
                continue
            annotation = tweetWord[1:].strip()
            annotation = annotation.strip('.')
            mentions.append(annotation)
            continue
        # Hashtags
        if tweetWord[0] == "#":
            if tweetWord[1:] == " ":
                continue
            hashtag = tweetWord[1:].strip()
            hashtag = hashtag.strip('.')
            hashtags.append(hashtag)
            continue

        # remove special characters
        tweetWord = re.sub('[^A-Za-z0-9 ]+', '', tweetWord)

        if len(tweetWord) == 0:
            continue

        text += tweetWord + " "
    if len(text) == 0:
        return None
    return date, time, text, hashtags, mentions
```

Figure B.1.: Ausschnitt eins der Sanierungsfunktion der Daten

C. Scatter Plots for the Sentiment Analysis

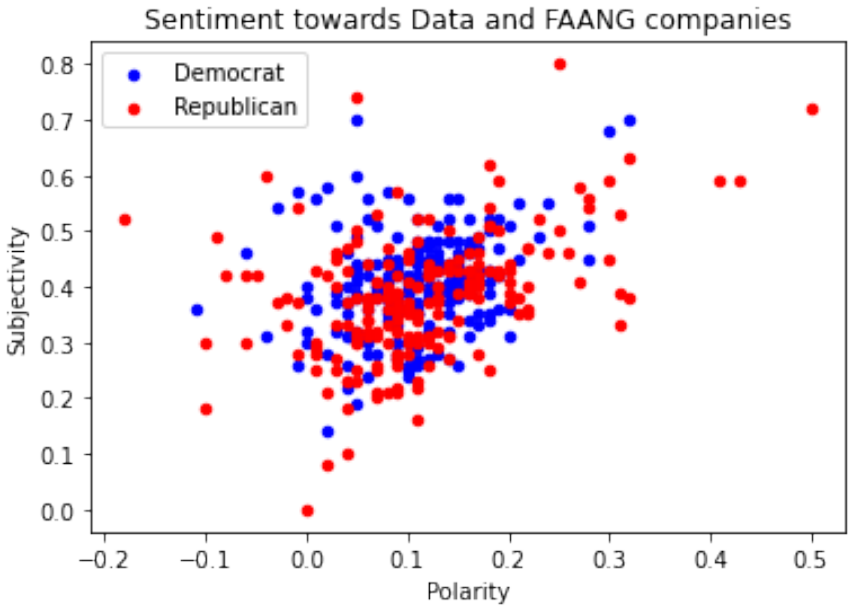

Figure C.1.: Sentiment towards Data and FAANG companies
In general the sentiment is postitive, however there are only minor differences between Republicans and Democrats. The tweets are in general moderately subjective.

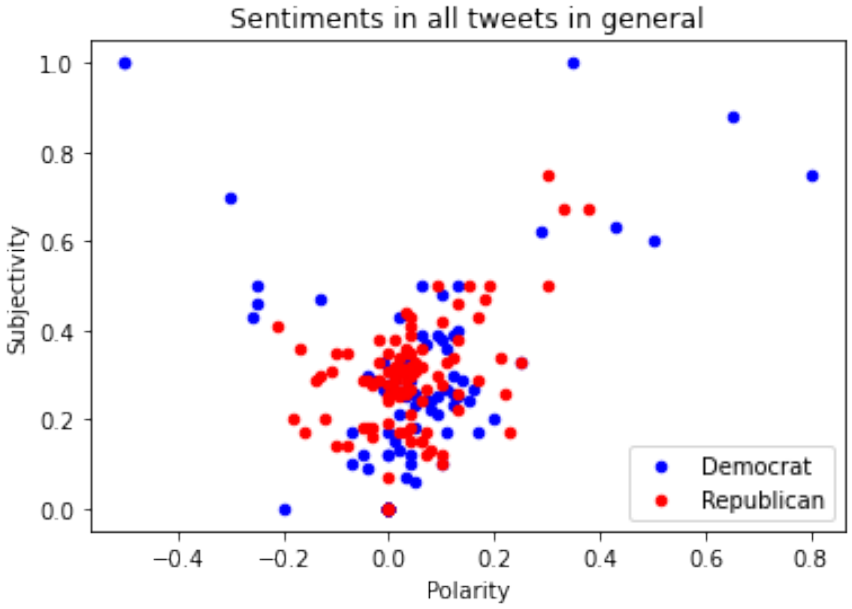

Figure C.2.: Sentiment in all tweets in general
In general the polarity is slightly positive and the subjectivity is relatively low, meaning that according to the library the politicians speak relatively truthful.

Literature

[boe18] BOERSENNEWS: *Patternanalyse*. https://www.boersennews.de/lexikon/begriff/pattern/852/. Version: 2018. – Last visited on 28.01.2023

[Dim14] DIMOCK, Michael: *Political Polarization in the American Public*. 2014

[Efu23] EFU, MIT: *Global Sentiment*. https://www.globalsentiment.mit.edu/. Version: 2023. – Last visited on 29.01.2023

[ES23] ES, Sahul: *Global Sentiment*. https://neptune.ai/blog/sentiment-analysis-python-textblob-vs-vader-vs-flair. Version: 2023. – Last visited on 29.01.2023

[Hen18] HENRIQUE, Jefferson: *GetOldTweets-python*. https://github.com/Jefferson-Henrique/GetOldTweetspython/blob/master/got3/manager/TweetManager.py. Version: 2018. – Last visited on 27.01.2023

[Hen19] HENRIQUE, Jefferson: *GetOldTweets3 0.0.11*. https://pypi.org/project/GetOldTweets3/. Version: 2019. – Last visited on 27.01.2023

[Lor17a] LORIA, Steven: *TextBlob-Sentimentanalysen*. https://github.com/sloria/TextBlob/blob/dev/textblob/en/sentiments.py. Version: 2017. – Last visited on 27.01.2023; Funktion PatternAnalyzer

[Lor17b] LORIA, Steven: *TextBlob-Sentimentanalysen Github*. https://github.com/sloria/TextBlob. Version: 2017. – Last visited on 27.01.2023

[Lor20a] LORIA, Steven: *TextBlob*. https://github.com/sloria/TextBlob/blob/dev/textblob/blob.py. Version: 2020. – Last visited on 27.01.2023; Class TextBlob

[Lor20b] Loria, Steven: *Tutorial: Quickstart*. https://textblob.readthedocs.io/en/dev/quickstart.html. Version: 2020. – Last visited on 28.01.2023

[NT23] NLTK-Team: *NLTK-Documentation*. https://www.nltk.org/. Version: 2023. – Last visited on 27.01.2023

[Yos20] Yoss, Andrea: *GetOldTweets3*. https://andrea-yoss.medium.com/getoldtweets3-830ebb8b2dab. Version: 2020. – Last visited on 27.01.2023