Auswerten von Nutzeraktivitäten

Flame-Wars-Erkennung in Sozialen Netzwerken am Beispiel Twitter

1st Benedikt Neubauer
Hochschule Furtwangen
HFU
78120 Furtwangen im Schwarzwald
Deutschland
benedikt.neubauer@hs-furtwangen.de

2nd Philipp Oeschger
Hochschule Furtwangen
HFU
78120 Furtwangen im Schwarzwald
Deutschland
philipp.oseschger@hs-furtwangen.de

***Zusammenfassung*—Nutzer von öffentlichen Netzwerken können aufgrund des Schleiers der Anonymität Beleidigungen veröffentlichen, ohne erkannt zu werden. Um negative Einflüsse vorzubeugen, müssen diese Interaktionen erkannt werden. Diese Arbeit untersucht Nutzeraktivitäten auf dem sozialen Netzwerk Twitter um Interaktionsphänomene wie Flame und Flame-Wars zu erkennen. Dabei werden Daten durch direkte und indirekte Interaktionen mit verschiedensten deutschen Politikern erhoben. Diese Daten werden anschließend analysiert und ausgewertet. Mithilfe von weiteren Werkzeugen werden die gewichteten Daten dargestellt und interpretiert. Durch diese Vorgehensweise konnten Key User, sowie die durchschnittliche Polarität jeder Interaktion festgestellt werden. Der daraus resultierende Negativgraph lässt die Twitter Accounts erkennen, welche im Datenzeitraum, durchschnittlich viel Negative Polarität in Form von Tweets veröffentlicht, oder diese in Form von Replies erhalten haben. Um dies zu erreichen, wurde eine Streaming Pipeline implementiert.**

***Index Terms*—flamewars, twitter, sentiment analysis, graph analysis, social media analysis, streaming data**

I. Einführung

Bei einem Flame spricht man von einem aufstachelnden, ruppigen oder polemischen Kommentar, der in Form einer Beleidigung im Usenet, einer E-Mail, in Chatsitzungen, einem Fronthread oder einem Wiki auftritt [1]. Heutzutage wird Flame gerne für aggressive Beiträge ohne Sachbezug verwendet. Ein Flame-War ist demnach eine kontroverse Diskussion, bei der die Teilnehmer unsachlich und schließlich beleidigend werden. Ein Flame-War entsteht meist aus einer sachlichen Diskussion, die dann in Nebenkriegsschauplätze abrutscht. Typisch ist dabei, dass die "Argumente" Schlag auf Schlag geliefert werden, so dass der Flame-War am Leben bleibt [1].

Da diese Ereignisse auf sozialen Plattformen nicht gerade förderlich sind, und es zudem relevanter wird, mögliche dabei entstandene Drohungen strafrechtlich zu verfolgen, soll in dieser Arbeit vorgestellt werden, wie Hass und mögliche Flame-Wars erkannt werden kann.

Durch die Auswertung der Daten können mögliche Gemeinschaftsstrukturen auf sozialen Plattformen aufgezeigt werden. Die durchschnittliche Polarität der Diskussionen dabei kann ebenfalls dargestellt werden. Dadurch sollen durchweg negative Konversationen aufgedeckt werden, um das Phänomen, Flame und Flame-Wars, zu erkennen.

Diese Arbeit kann für verschiedene Arten von Social-Media-Studien oder die Entwicklung von Tools zur automatischen Erkennung missbräuchlicher Kommentare durch maschinelles Lernen nützlich sein. Sie kann auch als Ausgangspunkt für weitere Forschung in diesem Bereich dienen.

II. Proof of Concept (PoC)

Für diese Ausarbeitung wurde die Plattform Twitter ausgewählt. Hierbei sind 15 Politiker:innen der Bundesrepublik Deutschland von Nöten, um Reaktionen auf dieser Plattform zu erzeugen. Bezüglich der Begrifflichkeiten eine kurze Einführung. Tweets sind veröffentlichte Inhalte auf der Plattform Twitter. Auf diese Tweets kann direkt mit einem eigenen Tweet geantwortet werden. In dieser Arbeit wird dann von einem Reply gesprochen. Tweets, die nun auf Replies reagieren, gelten als indirekte Replies und werden ebenfalls, zusammenhängend mit dem ursprünglichen Tweet, erfasst. Ein Beispiel zur Veranschaulichung: Politiker:in X die/der zu den Politiker:innen mit der größten Reichweite gehört, veröffentlicht einen Tweet. Benutzer:in A antwortet direkt darauf mit einem Reply. Benutzer:in B hat nun etwas zum Reply von Benutzer:in A zu veröffentlichen. Daraufhin werden jegliche Diskussionen dem Tweet von Politiker:in X zugeschrieben.

A. Ziel

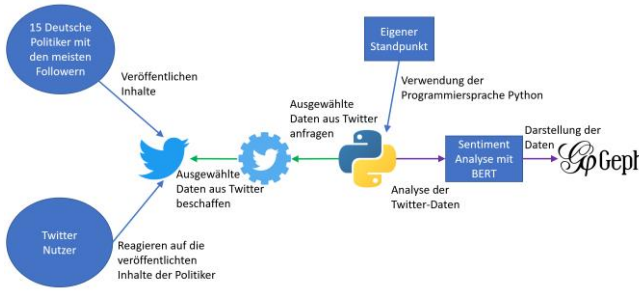
Das Ziel dieser Ausarbeitung ist es Flame und Flame-Wars auf Twitter zu erkennen. Dabei sollen 15 Politiker:innen und deren Tweets in einem bestimmten Zeitraum überwacht werden. Zuzüglich sollen alle Reaktionen, die mit einem solchen Tweet einhergehen mit aufgenommen werden. Durch die Reaktionen untereinander, über einen der Tweets der Politiker:innen, werden ebenfalls Benutzergruppen identifiziert. Dadurch lassen sich die Daten besser darstellen und nachvollziehen.

B. Machbarkeitsanalyse

Damit das genannte Ziel erreicht werden kann, muss ein Zugriff auf dieses Netzwerk erfolgen. Dafür stellt Twitter selbst eine eigene API zur Verfügung [2]. Mit dieser API lassen sich anhand von Twitter IDs und Tweet IDs die jeweiligen Posts den Politikern zuordnen. Jede Reaktion auf einen dieser Posts der Politiker:innen wird ebenfalls mit der ID des ursprünglichen Posts vermerkt. Dadurch lassen sich direkte Reaktionen ausfindig machen. Mithilfe einer zur Verfügung gestellten Konversation ID lassen sich alle impliziten Reaktionen untereinander bezüglich des Posts der Politiker:innen zusammenführen. Für die Umsetzung der Analyse wurde die Programmiersprache Python gewählt, zumal Python eine große Auswahl an Datenverarbeitungs-bibliotheken bietet. Die ausgewerteten und registrierten Daten gehen bis zu einem Punkt X in die Vergangenheit. Twitter bietet die Möglichkeit, neben den Vergangenheitsdaten, auch Echtzeitdaten auszuwerten. Diese sind momentan auf einen maximalen Wert, von 2 Millionen Tweets pro Monat, beschränkt. Mithilfe dieser API sollen die erhaltenen Daten in eigene Dateien über Python geschrieben,

dargestellt und ausgewertet werden. In Abbildung 1 wird der Ablauf bis hin zur Graphischen Darstellung in Gephi abgebildet.

Abbildung 1 Darstellung der Auswertungsstruktur(Quelle: Eigene Darstellung



III. VERGLEICH VON BERT UND TEXTBLOBDE

Die Inhalte der einzelnen Tweets und der Reaktionen darauf soll automatisiert kategorisiert und eingeordnet werden können. Damit nicht jeder einzelne Tweet manuell kategorisiert werden muss, soll die Polarität des Tweets maschinell ermittelt werden. Um dies zu ermöglichen, benötigen wir Sentiment Analyse-Tools, die diese Arbeit erledigen. Zwei mögliche Programme werden im nachfolgenden vorgestellt.

A. TextblobDE

Zum einen das wahrscheinlich bekannteste Sentiment Analyse Tool Textblob. Während der Namensgeber hauptsächlich auf englische texte spezialisiert ist, bietet TextblobDE die gleichen Funktionalitäten für die deutsche Sprache. TextblobDE ist eine Python-Bibliothek, um textuelle Daten zu verarbeiten. Es bietet eine einfache API für den Einstieg in gängige Aufgaben der natürlichen Sprachverarbeitung (NLP) wie Part-of-Speech-Tagging, Extraktion von Substantivphrasen, Stimmungsanalyse, Klassifizierung und Übersetzung [2]. Für diese Arbeit ist allerdings nur die Klassifizierung und Stimmungsanalyse relevant. Dies erfolgt hierbei mithilfe eines Bag of Words Ansatzes [3].

B. Sentiment Analyse mit Bert

German Sentiment-BERT (Bidirectional Encoder Representations from Transformers) ist ein Machine Learning Modell, das für die Sentiment Klassifizierung deutschsprachiger Texte trainiert wurde. Zur Verwendung wird ein Python-Paket zur Verfügung gestellt, welches die Vorverarbeitung des Codes sowie dessen Inferenzierung bündelt [4].

Das Modell verwendet die Bert-Architektur von Google und wurde auf 1,834 Millionen deutschsprachiger Stichproben trainiert. Die Trainingsdaten enthalten Texte aus verschiedenen Domänen wie Twitter, Facebook und Film-, App- und Hotelbewertungen. Die Genauigkeit dieser Analyse wird nach dem folgenden Paper [4] bei einem SB10K Test, mit 68% Genauigkeit angegeben. Bei dem SB10K Test hat man einen Datensatz aus ungefähr 10,000 Deutschen Tweets, welche korrekt kategorisiert werden müssen [5]. Dementsprechend ist die Genauigkeit der gesamten Analyse in diesem Paper mit Vorbehalt zu betrachten.

C. Vorteile Nachteile

Für diese Arbeit ergaben sich bei beiden Tools folgende Vorteile: TextblobDE hatte gegenüber Bert flexiblere Ausgabewerte, beispielsweise berechnet TextBlobDE beim Auswerten der Polarität Gleitwerte zwischen 1 und -1. Außerdem diente die Objektivität als eigener Parameter zusätzlich als Ausgabe. Bei BERT stellten sich folgende Vorteile heraus: Da dieses Modell auf einem recht großen Datensatz beruht und spezifisch für eine Stimmungsanalyse ausgelegt ist, bietet es sich für diese Arbeit an. Außerdem ist der Machine Learning-Ansatz insofern von Vorteil, da noch nie gesehene Wörter klassifiziert werden können [4]. – Dies ist mit TextblobDE nicht möglich und ein negativer Punkt dieser Bibliothek. Da sie weitaus mehr Funktionalitäten besitzt, ist ihre Spezialisierung, bezüglich der Stimmungsanalyse eventuell nicht ausgeprägt genug, im Gegensatz zu BERT. Außerdem kann diese Bibliothek Negierungen nicht erkennen, so wäre beispielsweise „nicht gut“ neutral gewertet worden, wobei „nicht schlecht“ negativ gewertet wird, da immer nur ein Wort analysiert werden kann und nicht der zusammenhängende Kontext einer Negierung [3]. Bei BERT fällt lediglich der beschränkte Werteberiech von negativ neutral und positiv unzureichend auf. Bezogen auf die Analyse in dieser Arbeit, gab Textblob zunehmend neutrale Ergebnisse aus. Das lässt sich entweder darauf zurückführen, dass zum einen viele Wörter oder zum anderen die zusammenhängenden Wörter und Ironie nicht erkannt wurde. Ein aussagekräftigeres Ergebnis als TextblobDE lieferte hingegen BERT. Deswegen wurde für die Arbeit das Sentiment Werkzeug BERT verwendet.

D. Übersetzungen in die Englische Sprache

Da in der Programmierung und bei Modellen oftmals Englisch die Hauptsprache ist, könnte man davon ausgehen, dass es zu besseren Ergebnissen führen könnte, wenn die deutschen Daten erstmal in die englische Sprache übersetzt werden, um diese danach auszuwerten. Da bei dieser Analyse eine Automatisierung stattfinden müsste, mit beispielsweise DeepL, wird eine zweite Fehlerquelle mit hinzu gezogen. Beim stichprobenartigen Überprüfen waren die automatisierten Übersetzungen inhaltlich mehrheitlich korrekt. Allerdings lassen sich beispielsweise Sprichwörter nicht so einfach in andere Sprachen übersetzen. Das Testergebnis dieser Analyse bezüglich der übersetzten Tweets sah den deutschen Ergebnissen sehr ähnlich. Allerdings musste dabei Textblob mit TextblobDE vergleichen werden, was für diese Arbeit weniger gut funktioniert wie die Sentiment Analyse mit BERT.

IV. UMSETZUNG DER DATENANALYSE

Basis für das Erkennen der Konversationsstrukturen, ist ein Datensatz, über welchen diese Strukturen ersichtlich sind und Analyse-Schritte, welche ermöglichen aus der Datenlage Schlüsse zu ziehen.

In den folgenden Kapiteln und Unterkapiteln wird genauer auf die Umsetzung und Hintergründe der einzelnen Verarbeitungsschritte eingegangen.

A. Einsatz der Twitter API

Es existieren mehrere Möglichkeiten, mit den Endpunkten der Twitter-API zu interagieren. Eine Möglichkeit, ist das Entwickeln eines eigenen Clients, um

Daten abzufragen und Streaming-Daten zu erhalten. Hierfür stellt Twitter eigene Codebeispiele bereit [6] [7].

Eine weitere Variante, ist das Verwenden einer (meist Programmiersprachenspezifischen) Open Source-Bibliothek, welche die Zugänge zur Twitter-API in wohl definierten Schnittstellen, Funktionen und Klassen bereitstellen [6].

Für den PoC dieser Arbeit wurden beide genannten Varianten für unterschiedliche Teilprobleme in Betracht gezogen.

Zum Sammeln der Streaming-Daten wurde eine Streaming-Connection-Klasse entwickelt. Für darüber-hinausgehende Datenabfragen wurde die Bibliothek „Tweepy“ eingesetzt [8].

B. Bestimmung der Politiker:innen

Die Politiker:innen wurden mithilfe der Twitter-Liste „MdB (Bundestag)“ bestimmt [9]. Diese Liste beinhaltet eine Sammlung, auf Twitter aktiver deutscher Politiker:innen des Bundestages. Die Politiker:innen dieser Liste wurden ausgelesen und nach Anzahl der Follower (Nutzer:innen, die den Account verfolgen) sortiert. Anschließend wurden die 15 Politiker:innen mit der höchsten Follower-Anzahl extrahiert.

C. Rahmenbedingung zur Aufzeichnung der Daten

Im nächsten Schritt wurden die primären Tweets der Politiker:innen und die direkten und indirekten Replies mithilfe der Twitter-Streaming-API gesammelt. Ein / Eine Nutzer:in hat die Möglichkeit, Retweets (ein / eine Nutzer:in vermerkt den Tweet eines / einer anderen Nutzer:in auf seiner öffentlichen Übersichtsseite) zu veröffentlichen. Da Retweets keine originären Inhalte des/ der Nutzer:in darstellen, spiegeln sie nur bedingt die Gedanken des / der Nutzer:in wider. Eine Betrachtung der Retweets könnte somit die Ergebnisse der Analyse verfälschen. Aus diesem Grund wurden Retweets im Rahmen der Analyse ignoriert.

Die Python-Bibliothek „Tweepy“ stellt eine Klasse für das Empfangen von Streaming-Daten zur Verfügung. Somit wäre eine naheliegende Lösung gewesen, diese Klasse als Ausgang für die Datensammlung zu verwenden [10]. Bedauerlicherweise erlaubt diese Klasse lediglich das Filtern der verfassenden Nutzer und das Filtern nach Stichwörtern. Die Möglichkeit, nach der Konversations-ID filtern zu können, wird nicht angeboten.

Es ist somit nötig, eine eigene Klasse für das Filtern von Twitter-Streaming-Daten zu implementieren, welche die Konversations-ID berücksichtigt.

D. Implementierung des Twitter Streamers

Für die Implementierung der Streaming-Abfragen wurden zwei Klassen konzipiert. Die Klasse „TwitterStreamer“ stellt Grundfunktionalitäten für das Initialisieren des Streaming-Prozesses und für das Empfangen der Daten bereit. Die Klasse „FilteredTwitterStream“ erweitert die Klasse „TwitterStreamer“ mit Methoden zum Filtern von Nutzern und Konversations-IDs.

Bei der Implementierung des TwitterStreamers wurde sich an dem von Twitter zur Verfügung gestellten Code-Beispiel „Filtered Stream“ orientiert [7].

E. Filtered Streams

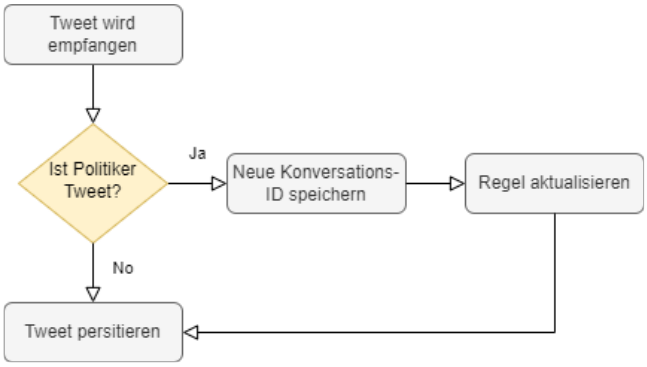
Zum Filtern von Streaming Daten stellt Twitter ein Regelsystem basierend auf einer Menge an Operatoren, kombinierbar mit Boolean-Logik zur Verfügung. Zusätzlich können Regeln mit Tags annotiert werden. Wenn eine Filterregel zutrifft, wird beim Streaming der betroffene Tag an den Tweet angehängt [6].

Die Klasse „FilteredTwitterStream“ implementiert eine Regel für das Beobachten der Politiker:innen. Diese Regel wird mit dem Tag „primary_tweet“ versehen. Zusätzlich wird pro Politiker:in eine Regel implementiert, um dessen Konversationskette zu verfolgen, markiert mit den Nutzernamen der jeweiligen Politiker:innen als Tag.

Im Fall, dass ein Tweet mit dem Tag „primary_tweet“ gekennzeichnet ist, ha ein /eine Politiker:in neue Inhalte veröffentlicht. Die Konversations-Regel des Politikers muss angepasst werden. Als Nächstes wird die betroffene Regel aktualisiert, indem eine neue ID als Konversations-ID hinzugefügt wird. Die Tweet-ID bzw. gleichzeitig die Konversations-ID wird in die Liste der zu beobachtenden Konversationen mit aufgenommen.

Wenn eine der Regel zutrifft, wird der Tweet gespeichert. Dieser Prozess wird in Abbildung 2 veranschaulicht. Um einem Ausfall des Streaming-Moduls vorzubeugen, werden die Konversations-IDs der jeweiligen Politiker:innen in einer Datei persistiert.

Abbildung 2 Prozess der Konversations-Regel Aktualisierung (Quelle: Eigene Darstellung)



F. Aggregieren der Daten

Im Zeitrahmen vom 01.01.2022 bis zum 07.01.2022 wurden primäre Tweets aggregiert. Um eine Reaktion auf die Tweets abwarten zu können, wurden die Replies auf die Tweets bis zum 08.01.2022 aggregiert. Gespeichert wurden die Daten in jeweils einer Datei für primäre Tweets und einer Datei für Replies, im CSV-Format [11].

Insgesamt wurden 102 primäre Tweets von Politiker:innen und 28554 direkte und indirekte Replies auf diese Tweets gesammelt.

G. Tweetformat

Ein Tweet beinhaltet viele Informationen [11] die für Analysen und Auswertungen sein können.
Für den PoC wurden die in

Tabelle 1 dargestellte Eigenschaften berücksichtigt.

Tabelle 1 Aufbau eines Tweets

Attribut	***Beschreibung***
id	Eindeutiger Identifier des Tweets
Create_at	Menschenlesbares Datum, zudem der Tweet erstellt wurde
Full_text	Inhalt des Tweets
Conversation_id	Identifier der Konversationskette
User_id	Identifier des Nutzers
Screen_name	Nutzername
mentions	Nennungen andere Nutzer innerhalb des Tweets
hashtags	Verwendete Abkürzungen oder Stichwörter (gekennzeichnet mit dem Symbol „ #“)

H. Durchführung der Stimmunganalyse

Im nächsten Schritt wurde die Stimmungsanalyse mit German Sentiment Bert für die Textinhalte der Tweets durchgeführt. Die Ausgabewerte „Positiv“, „Neutral“ und „Negativ“ wurden hierbei auf die Werte 1, 0 und -1 abgebildet, um einfacher Berechnungen mit den Ausgaben durchführen zu können. Die Ausgabewerte wurden zusammen mit den analysierten Tweets in einer CSV-Datei gespeichert.

V. GEMEINSCHAFTSSTRUKTUREN ERKENNEN

Um die Ergebnisse der Stimmungsanalyse in Kontext der Nutzerbeziehungen genauer betrachten zu können, war Ziel des nächsten Schritts, eine Gemeinschaftsanalyse durchzuführen. Die Erkennung von Nutzer-Beziehungen und Gruppenbeziehungen hat sich zu einem der Hauptdisziplinen in der Social Media-Analyse entwickelt und begründet den Teilbereich der Social Network-Analyse (SNA) [12].

Mithilfe der SNA können Key User innerhalb eines sozialen Netzwerks erkannt werden. Eine weitere beliebte Problemstellung ist die der Erkennung zusammenhängender Gemeinschaftsgruppen. Hierbei bedient sich die SNA bei den Konzepten und Messzahlen der Graphentheorie [12].

A. Grundlagen der Graphentheorie

Ein Graph [13] ist ein mathematisches Konzept, welches eine Menge an verbundenen Elementen, Knoten (engl. Nodes/ Vertices) genannt, beschreibt. Diese Knoten haben Beziehungen zueinander, welche Kanten (engl. Edges) genannt werden. Wenn diese Kanten genau eine Richtung aufweisen, spricht man von einem gerichteten Graphen. Im Fall, dass Kanten mit nichtnegativen Werten versehen sind, ist von einem gewichteten Graphen die Rede [13]. Knoten eines gerichteten Graphen haben eines Eingangs- und einen Ausgangsgrad. Der Eingangsgrad beschreibt, wie viele Beziehungen zu diesem Knoten gerichtet sind. Der Ausgangsgrad beschreibt, wie viele Kanten weg von diesem Knoten zeigen. Aus der Summe der beiden Grade berechnet sich der Gesamtgrad [13].

B. Generieren des Graphen

Für das Erstellen und Verarbeiten des Graphen wurde die Python-Bibliothek NetworkX [14] verwendet. Mithilfe dieser Bibliothek lassen sich Graphen mit überschaubarem Aufwand erstellen und für spätere Verarbeitungszwecke exportieren und persistieren.

Die Tweets und Replies wurden sequenziell betrachtet und aus ihnen die Nutzer extrahiert. Kern der Beziehungen zwischen den Nutzern sind die Nennungen innerhalb eines Tweets. Jede Nennung beschreibt eine gerichtete Beziehung vom nennenden Nutzer zum genannten Nutzer. Eine Nennung hat ein Gewicht vom Wert 1. Im Fall, dass ein Nutzer einen anderen Nutzer mehrmals nennt, werden die Gewichte addiert. Zusätzlich wurde pro Beziehung die durchschnittliche Polarität resultierend aus der Stimmungsanalyse berechnet. Das Resultat der Verarbeitungsschritte war ein gerichteter gewichteter Graph bestehen aus 12618 Knoten und 27101 Kanten.

Die Limitierung dieser Analyse ist, dass nur Nennungen und keine anderen Faktoren, wie z. B. Folgebeziehungen betrachtet werden. Im Rahmen von Nachforschungen wäre für ein besseres Gesamtbild das Miteinbeziehen von weiteren Nutzerbeziehungen sinnvoll

C. Negativ-Graph

Weiterführend wurde ein „Negativ-Graph“ generiert, der nur Beziehungen, dessen durchschnittliche Polarität unter dem Wert 0 liegt, betrachtet. Somit kann der Fokus stärker auf negative Kommentar-Beziehungen gelegt werden. Resultat ist ein Graph mit 11983 Knoten und 25269 Kanten (siehe Abbildung 4).

Dieser Ansatz spiegelt möglicherweise ein verfälschendes Gesamtbild wider, zumal hierbei die Gewichtungen der Beziehungen im Kontext zur Polarität nicht berücksichtigt werden. Für Nachforschungen ist eine Hinterfragung dieses Ansatzes erdenklich.

D. Eigenschaften von Gemeinschaften im Graph

Allgemein spricht man von Gemeinschaften in Graphen, wenn Knotengruppen existieren, welche innerhalb der Gruppe viele Verbindungen aufweisen und die Knoten dieser Gruppe nur wenige Verbindungen zu Knoten anderer Gruppen aufweisen [15]. In der Disziplin der Gemeinschaftserkennung ist die Aufgabe, qualitative und die optimale Anzahl solcher Gruppenstrukturen zu finden. Hierfür wird ein Qualitätsmaß vorausgesetzt.

E. Modularität

Ein etabliertes Qualitätsmaß, in der Analyse von Gemeinschaften ist die Modularität. Sie vergleicht die Dichte der Verbindungen innerhalb von Gemeinschaften mit der Dichte der Verbindungen zwischen Gemeinschaften [15] [16]. Die Modularität wird üblicherweise mit dem Symbol Q gekennzeichnet.

Wenn das Verhältnis von Kanten innerhalb der Gemeinschaften nicht besser ist als das gleiche Verhältnis innerhalb eines zufällig konfigurierten Netzwerks, gilt $Q=0$. Wenn Q über 0.3 liegt, kann von signifikanten Gemeinschaftsstrukturen innerhalb des Netzwerks ausgehen [15] [16].

Das Maß der Modularität kann eingesetzt werden, um durch Maximierung dieses Wertes hochwertige Gemeinschaftsbeziehungen festzustellen. Bedauerlicherweise ist das Problem der Modularitäts-maximierung nachweislich NP-Vollständig [17] und somit sind Approximationsalgorithmen nötig, um signifikante Gemeinschaftsstrukturen mit annäherungsweise optimaler Modularität zu bestimmen.

F. *Graphendarstellung und Gemeinschafterkennung mit Gephi*

Gephi ist ein Open-Source-Tool zur Graphen-Analyse, welches es ermöglicht Graphen in Echtzeit darzustellen [18].

Für die Maximierung der Modularität stellt die Software Gephi einen Annäherungsalgorithmus bereit, welcher für die Analyse des Graphen eingesetzt wurde [19]. Der Algorithmus setzt sich aus zwei Phasen zusammen. In der ersten Phase wird, durch lokale Gemeinschaftswechsel der Knoten, die Modularität maximiert. In der zweiten Phase werden die Partitionen aggregiert, um ein neues Netzwerk bestehend aus Gemeinschaften zu erstellen. Diese beiden Phasen werden wiederholt, bis keine Verbesserung der Modularität mehr feststellbar ist. Das Ergebnis ist ein schnell konvergierender Algorithmus für eine effektive und qualitative Gemeinschaftserkennung.

Die Abstandsbeziehungen werden mit dem Graphen-Layout-Algorithmus ForceAtlas2 dargestellt, ein kräftebasiertes Layout, welches die Ideen verschiedener Algorithmen in sich vereint [20]. Während Knoten sich gegenseitig abstoßen, verhalten sich Kanten hingegen wie Federn. Diese Zusammensetzung soll eine bessere Dateninterpretation ermöglichen.

VI. Resultate und Ergebnisse der Datenanalyse

In Abbildung 4 wird der in V.C beschriebene „Negativ-Graph“ dargestellt, auf dem eine Modularitätsmaximierung mit Gephi durchgeführt wurde.

Das Ergebnis der Analyse ist eine Gemeinschaftseinteilung mit *Q=0.509*. Hiermit sind signifikante Gemeinschaftsstrukturen im Netzwerk erkennbar. Resultat der Modularitätsmaximierung sind 105 Gemeinschaften.

A. *Interpretation des Graphen?*

Das folgende Unterkapitel bezieht sich auf Abbildung 4
Den Erwartungen entsprechend weißen die Politiker:innen, Gegenstand der Betrachtung, die größten gewichteten Grade auf, zumal sie Basis für die Datensammlung waren. Für die hohen Werte sind hierbei fast ausschließlich die Eingangsbeziehungen verantwortlich. Zwischen manchen Gemeinschaftsgruppen sind starke Übergangsbeziehungen erkennbar. Beispielsweise existieren viele Nutzer-beziehungen zwischen der gelb markierten (Christian Lindner) und der rot markierten Gemeinschaft (Karl Lauterbach). Ebenfalls sind Gemeinschaften erkenntlich zwischen denen wenige Schnittmengen vorherrschen. Beispiel hierfür ist die hellrot markierte Gemeinschaft (Kevin Kühnert), welche nur wenige Stränge zu anderen Gemeinschaften aufweist.

Bei der genaueren Betrachtung wiesen einige Nutzer, außerhalb der betrachteten Politiker:innen hohen gewichteten Eingangs und Ausgangsgrad-Wert auf. Diese Nutzer wurden als Key User identifiziert.

B. *Key User*

Anhand des generierten Graphen durch die Nutzerdaten konnten neben den Reaktionen auf die Tweets der Politiker:innen und den daraus resultierenden Strukturen Schlüssel Benutzer identifiziert werden. Diese Schlüssel Benutzer konnten durch die Anzahl der häufigen Interaktionen im gewählten Zeitraum ausfindig gemacht werden. Dabei wurden jeweils diejenigen Nutzer herausgesucht, die entweder die meisten indirekten, beziehungsweise direkten Replies abgesetzt haben. Auf der anderen Seite wurden die Nutzer herausgesucht, welche die meisten Replies auf ihre Reaktion von Politiker-Tweets erhalten haben. Dabei wird im Folgenden der Fokus auf die fünf Accounts gelegt, welche die meisten Interaktionen hervorgerufen haben. Dabei soll veranschaulicht werden, wie das Verhältnis der eigenen Tweets und der resultierenden Replies der jeweiligen Nutzer ist. Außerdem soll dabei die Polarität der jeweiligen Interaktionen dargestellt werden. Anschließend werden die dazugehörigen Twitter-Accounts auf der Plattform selbst angeschaut, um einen eigenen Eindruck der Lage zu erhalten, was die Themen der jeweiligen Accounts waren. Dies soll nicht dazu dienen die jeweiligen Personen hinter den jeweiligen Accounts zu kategorisieren, oder in eine politische Richtung gedrückt werden. Stattdessen soll damit der generierte Graph verständlicher gemacht werden. In der nachfolgenden Tabelle 2 werden die fünf wesentlichsten Key User dargestellt.

Tabelle 2 Nutzeraktivitäten der Key User

Twitter ID	***Anzahl***	***Polarität***
erlangerfranke	376 eigens verfasste Tweets	0.0637
	520 Erhaltene Replies	-0.14
Henry53246076	422 eigens verfasste Tweets	-0.13
	49 Erhaltene Replies	-0.16
Mgowan18	354 eigens verfasste Tweets	-0.12
	259 Erhaltene Replies	-0.1262
Abnonhar	219 eigens verfasste Tweets	-0.05
	268 Erhaltene Replies	-0.1269
SabineB3r	373 eigens verfasste Tweets	-0.14
	79 Erhaltene Replies	-0.19

Aus der Tabelle 2 lässt sich entnehmen, dass es jeweils Unterschiede gibt bezüglich des Verhältnisses von Tweets und Replies. Der Account mit dem Namen Henry, beispielsweise veröffentlicht weitaus mehr Tweets selbst, als das er darauf Reaktionen bekommt. Allgemein lässt sich sagen, dass egal wie neutral oder nicht ganz so negativ die veröffentlichten Tweets sind, die Replies sind immer negativer. Wenn man sich die einzelnen Accounts nun auf Twitter anschaut, sind die Accounts alle verschieden.
Beispielsweise ist der erste Account „erlangerFranke“ auf Twitter nicht mehr zu finden. Entweder wurde dieser Account von den Seiten Twitters oder vom Betreiber selbst gelöscht. Es können deswegen keinerlei weitere Interpretationen bezüglich des Accounts angestellt werden.

Der Account mit dem Namen Henry kommentiert nahezu alle Themen. Vorzugsweise verteidigt dieser Account den von der Politik eingeschlagenen Weg und versucht anderen Meinungen kontra zu bieten. Dabei beleidigt der Account auch gerne mal die Person dahinter oder stellt die jeweilige Intelligenz in Frage. Der Account Mgowan18 hat kein Verständnis für die Impfung und ist prinzipiell unzufrieden mit den Krisenmanagement Entscheidungen der Politik. Der Account mit dem Namen Abnonar unterstützt laut eigenen Angaben das Recht von Kindern und sieht die Entscheidungen, die für Kinder bezüglich des Umgangs mit dem Virus getroffen wurden als fehlerhaft an. Der Account SabineB3r veröffentlicht unter anderem „Politische Bilder“ die von der Gesellschaft als Kontroverse Meinung verstanden werden kann. Darüber hinaus heißt dieser Account eine Impfung gegen das Coronavirus nicht gut und gibt dies öffentlich Preis.

All diese Accounts haben eigene Standpunkte und Meinungen, die sich voneinander unterscheiden. Aufgrund dessen rufen sie jeweils verschiedenste Reaktionen hervor, was zu der dargestellten Analyse führt. Bezüglich des übergeordneten Themas Flame-Wars detection können prinzipiell Polaritätstendenzen festgestellt werden, ob der Austausch neutral oder eben eher negativ ist.

C. Ergebnisse der Daten bezogen auf Politiker:innen

Neben den identifizierten Schlüsselfiguren und den jeweiligen Strukturen dahinter, dienten die Tweets der Politiker:innen als Grundlage der Auswertung von Nutzeraktivitäten. Die durchschnittliche Polarität der jeweiligen Reaktionen wird in folgender Tabelle 3 zusammengefasst. Zu sehen sind nur 13 der 15 Politiker:innen, da zwei der ursprünglich 15 Politiker:innen im Zeitraum zur Erfassung der Daten keinerlei Tweets veröffentlichten.

Tabelle 3 Polarität der ausgewählten Politiker:innen

Name der Politiker:innen	***Durchschnittliche Polarität der direkten und indirekten Replies***
1. Kevin Kühnert	-0.30746687788018434
2. Armin Laschet	-0.29779121525153274
3. Christian Lindner	-0.23075869789363623
4. Friedrich Merz	-0.2141485021003093
5. Goering Eckardt	-0.20510887772194303
6. Gregor Gysi	-0.19607843137254902
7. Heiko Maas	-0.19230769230769232
8. Annalena Baerbock	-0.1718038302277433
9. Cem Özdemir	-0.16354587728811332
10. Sarah Wagenknecht	-0.1478205041185104
11. Alice Weidel	-0.1475022258225912
12. Katja Kipping	-0.14583333333333334
13. Karl Lauterbach	-0.11809388174468166

Allgemein lässt sich feststellen, dass prinzipiell die Replies auf die Tweets, laut der Sentiment Analyse, durchweg negativ ist. Der Grad der Negativität unterscheidet sich hingegen. Daraus lässt sich erkennen, dass die durchschnittlich negativsten Replies Herr Kühnert erhalten hat. Auf der anderen Seite hat Herr Lauterbach trotz dessen, dass er die meisten Replies während des Zeitraums erhalten hat, die durchschnittlich am wenigsten negative Polarität zurückbekommen.

Die Politiker:innen wurden wie in Kapitel V.F beschrieben zugeordnet und entsprechend in Tabelle 4 eingeteilt:

Tabelle 4 Prozentualer Anteil der Benutzergruppen von Gesamtknoten im Graphen

Name der Benutzergruppe	***Prozentualer Anteil der Gesamtknoten***
Karl Lauterbach	44,69%
Christian Lindner	20,83%
Alice Weidel	9,68%
Friedrich Merz	5,27%
Cem Oezdemir, Annalena Baerbock, Heiko Maas	3.71%
Armin Laschet, Goering Eckardt, Katja Kipping	3.2%
Sarah Wagenknecht	2,96%
Kevin Kühnert	2,8%
Gregor Gysi	In keiner Signifikanten Gemeinschaft

Damit wird auf einen Blick sichtbar, dass Karl Lauterbach für die meisten Reaktionen verantwortlich war. Außerdem sind nicht alle Politiker:innen ihren tatsächlich Politischen Gruppen zugeordnet. Laut Gephi haben beispielsweise Heiko Maas und die Accounts, die auf ihn reagieren, sehr ähnliche Beziehungen wie die beiden Grünen Politiker:innen, Oezdemir und Baerbock. Eine weitere Ausnahme spielt Katja Kipping, Goring Eckardt und Armin Laschet die laut Gephi der gleichen Gruppierung angehören. Somit muss es durch die Modularitäts-Analyse genügend Ähnlichkeiten untereinander geben.

Die thematischen Zusammenhänge und Zugehörigkeiten werden in Form von Hashtags festgehalten. Durch die Hinzunahme dieser Infromationen kann man die allgemeine Gesprächsthematik sowie die aktuellen Trends grob nachvollziehen. Die Hashtags mit dem höchsten Vorkommen sind in nachfolgender Tabelle 5 aufzufinden:

Tabelle 5 Meistgenutzte Hashtags der Datenanalyse

Hashtag	***Vorkommen***
Omikron	133
Impfpflicht	98
Lauterbach	77
FDP	76
LongCovid	72

Somit resultieren die meisten Reaktionen aufgrund von Corona und den momentan Vorherrschenden Varianten. Sowie mit aktuellen politischen Entscheidungen und dazugehörigen Schlüsselpositionen.

VII. Streaming-Architektur

Die bisherige Grenze der Verarbeitungsarchitektur ist, dass Daten als Batch gesammelt und in Batch-Form verarbeitet werden. Um die Daten in Echtzeit empfangen und verarbeiten zu können, sollte die bisherige Architektur erweitert werden.

Weiterführend zum PoC wurde eine Streaming-Architektur entwickelt, die in Echtzeit die Daten empfängt, aufbereitet, verarbeitet und persistiert.

Für diesen Zweck wurde Apache Kafka eingesetzt. Kafka bietet eine solide Grundlage für ein erweiterbares, hochperformantes und möglicherweise verteiltes System im Rahmen von Nachforschungen [21].

Die Daten werden nach der Verarbeitung in einer Graphen-Datenbank gespeichert. Hierfür wurde Neo4J [22] verwendet.

A. *Kafka Architektur*

Es wurde die Bibliothek Kafka-Python verwendet. Ein Python-Client, der die Interaktion mit einem Kafka-System ermöglicht [23]. Da bereits in vorigen Schritten mit Python-spezifischen Bibliotheken gearbeitet wurde und da das eingesetzte Sentiment-Analyse-Modell Python spezifisch ist, war die Verwendung eines Python-Clients naheliegend, um ungewollte Nebeneffekte zu vermeiden.

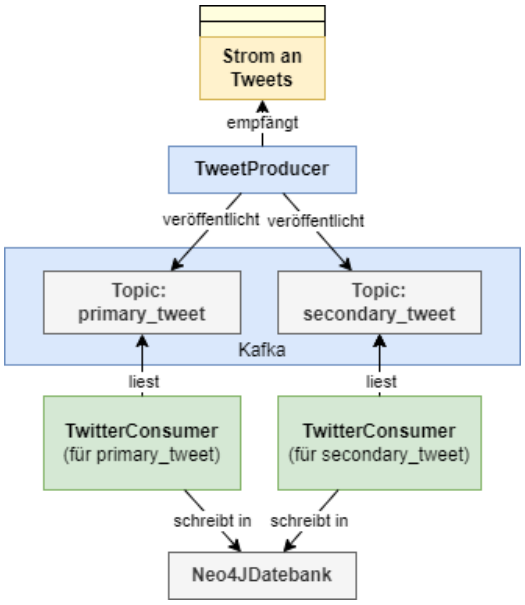
Die bestehende Architektur beschrieben in IV.D wurde durch die Klassen „TwitterProducer“, „TweetConsumer“ und „Neo4JHelper“ erweitert.

Diese Klasse „TwitterProducer“, übernimmt die Aufgabe, Daten (Tweets) im Kafka-System zu veröffentlichen. Für das Empfangen der Tweets sind Objekte der Klasse „TweetConsumer“ zuständig. „TweetConsumer“ können über ein „Neo4JHelper“ mit der Neo4J-Datenbank interagieren. Das Zusammenspiel der Komponente wird in Abbildung 3 dargestellt.

B. *Topics*

Es wird zwischen zwei Topics unterschieden, „primary_tweet“ und „secondary_tweet“. Wenn einer der 15 Politiker:innen einen Tweet veröffentlicht, fällt dieser Tweet unter das Topic „primary_tweet“. Direkte und indirekte Replies auf diese Tweets werden mit Topic dem „secondary_tweet“ gekennzeichnet.

Abbildung 3 Streaming-Architektur (Quelle: Eigene Darstellung)



C. *PrimaryTweetProducer*

Die Aufgabe des „PrimaryTweetProducers“ ist es, Tweets entweder mit dem „primary_tweet“ oder dem „secondary_tweet“ Topic im Kafka-System zu veröffentlichen. Die Twitter-Streaming-Funktionalität übernimmt die Klasse von der in IV.E vorgestellten Klasse „FilteredTwitterStream“.

D. *TweetConsumer*

Von dieser Klasse werden zwei Instanzen gebildet. Eine Instanz abonniert das Topic „primary_tweet“ und die zweite Instanz das Topic „secondary_tweet“. Zusätzlich übernimmt die Klasse die Aufgaben der Tweet Verarbeitung und das Einleiten des Speicherprozesses.Der Tweet Consumer erhält per Dependency-Injection eine Abhängigkeit zu einem „Neo4JHelper“-Objekt, über das es mit der Datenbank interagieren kann.

Der erste Schritt der Verarbeitung ist die Stimmungsanalyse des Tweet-Textes. Als Nächstes wird, falls noch nicht vorhanden, der Verfasser des Tweets als User-Objekt in der Datenbank angelegt.

Danach werden die Nennungen von Nutzern im Tweet betrachtet. Für jede Nennung wird, falls nötig ein User-Objekt angelegt. Nutzer bekommen je nach Topic des Tweets den Typ „primary_user“ oder den Typ „secondary_user“ in der Datenbank zugeschrieben, um später unterscheiden zu können, welche Nutzer Ausgangspunkt der Betrachtung waren.

Im nächsten Schritt wird überprüft, ob bereits eine Beziehung zwischen den Nutzern besteht. Falls das nicht zutrifft, muss eine neue Beziehung angelegt werden und für diese der durchschnittliche Polaritätswert und das Gewicht initialisiert werden. Falls es zutrifft, muss der durchschnittliche Polaritätswert, mit dem Ergebnis der Stimmungs-Analyse, aktualisiert werden. Zusätzlich muss auf das Gewicht der Beziehung eins aufaddiert werden.
Um zu einem späteren Zeitpunkt genauere Nachforschungen ermöglichen zu können, wird zusätzlich zum Nutzer ein Tweet Objekt mit einer Beziehung zum verfassenden Nutzer erstellt. Der Tweet enthält, zuzüglich aller in

Tabelle 1 genannten Eigenschaften, den berechneten Polaritätswert der Stimmungsanalyse.

Durch diese Schritte existiert nach jedem Verarbeitungsschritt ein gültiger Graph mit allen aktuellen Nutzerbeziehungen in der Graphen Datenbank. Der Graph enthält den gleichen Informationswert, wie das Resultat aus der in IV beschriebenen Umsetzung.

E. Neo4JHelper

Der „Neo4JHelper“ bietet Funktionalität zum Anlegen oder Erfragen von Nutzern (Knoten) oder Beziehungen (Kanten).

F. Auslesen des Graphen

Mithilfe eines Streaming-Plugins ist es möglich, direkt von der Neo4J-Datenbank aus, Graphen in Gephi zu importieren [24]. Hiermit ist es möglich, Zwischenschritte zu vermeiden und die in der Arbeit behandelten Graphen-Analyseschritte durchzuführen.

G. Limitierungen

Das aktuelle Streaming Modell weist ein wesentliches Problem auf. Falls Bedarf besteht, Beziehungen zwischen Nutzern systematisch nach Zeitpunkten der Replies wieder abzubauen, ist das im aktuellen Zustand nur schwer oder gar nicht umsetzbar. Die Zeitinformation der in der Nutzerbeziehung enthaltenen Tweet geht beim Speicherprozess in der Datenbank verloren. Schlimmstenfalls müsste der gesamte Graph zurückgesetzt werden. Ein Ansatz, der nur in wenigen Anwendungsfällen zielführend ist. In Folgeprojekten wäre erdenklich, Beziehungsstrukturen in der Graph-Datenbank durch die Beziehungen der Tweets zueinander auszudrücken. Die Tweet-Objekte der Datenbank könnten schließlich nach Datum gefiltert und die Beziehungen zwischen Nutzern indirekt ermittelt werden.

VIII. FAZIT

Zusammenfassend lässt sich festhalten, dass mithilfe von Datenanalysen Flame und Flame-Wars erkannt wird. Allerdings ist der bisherige Ansatz nicht ausgereift. Die beiden vorgestellten Werkzeuge sind bisher valide Möglichkeiten eine solche Nutzeraktivitäten-Analyse zu spezifizieren. Bei der bisherig verwendeten Sentiment Analyse ist die Genauigkeit nicht perfekt. Hierfür wären Weiterentwicklungen von Nöten, um die Genauigkeiten der Analysen zu erhöhen. In Kombination mit einer Analyse der Gemeinschaftsgruppen konnten ergänzend mögliche Zusammenhänge zwischen den in Flame involvierten Nutzern festgestellt werden. Um weitere Einblick zu erhalten, wäre eine Veränderung des Zeitraums für die Datenerfassung von Nöten. Beispielsweise wird ein Zeitraum von mehreren Wochen untersucht in dem der Fokus dabei nicht zwingend auf der Politik und deren Vertreter:innen sondern bei anderen kontroversen Themen liegt. Es war möglich im Rahmen der Arbeit eine Streaming Pipeline aufzusetzen, welche die Datenanalyse zunehmend automatisiert.

IX. LITERATURVERZEICHNIS

[1] academic, „academic dictionaries und encyclopedias,“ [Online]. Available: https://de-academic.com/dic.nsf/dewiki/446063. [Zugriff am 09 02 2022].

[2] M. Killer, „ textblob-de,“ textblob + Sphinx, 2019. [Online]. Available: https://textblob-de.readthedocs.io/en/latest/. [Zugriff am 13 02 2022].

[3] S. Loria, „TextBlob: Simplified Text Processing,“ 2020. [Online]. Available: https://textblob.readthedocs.io/en/dev/. [Zugriff am 10 02 2022].

[4] A.-K. S. F. B. H.-J. B.̈ Oliver Guhr, „Training a Broad-Coverage German Sentiment Classification Model for Dialog Systems,“ 11-16 05 2020. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2020/pdf/2020.lrec-1.202.pdf. [Zugriff am 10 02 2022].

[5] J. M. D. D. E. F. U. Mark Cieliebak, „A Twitter Corpus and Benchmark Resources for German Sentiment Analysis,“ paperswithcode, 2017. [Online]. Available: https://paperswithcode.com/paper/a-twitter-corpus-and-benchmark-resources-for. [Zugriff am 10 02 2022].

[6] Twitter, „Twitter API documentation,“ [Online]. Available: https://developer.twitter.com/en/docs/twitter-api. [Zugriff am 10 02 2022].

[7] twitterdev, „Twitter API v2 sample code,“ [Online]. Available: https://github.com/twitterdev/Twitter-API-v2-sample-code. [Zugriff am 10 02 2022].

[8] Tweepy.org, „Tweepy,“ [Online]. Available: https://www.tweepy.org/. [Zugriff am 10 02 2022].

[9] M. Fuchs, „MdB (Bundestag),“ Twitter, [Online]. Available: https://twitter.com/i/lists/912241909002833921. [Zugriff am 10 02 2022].

[10] Tweepy.org, „Tweepy Documentation — tweepy 4.5.0 documentation,“ [Online]. Available: https://docs.tweepy.org/en/stable/. [Zugriff am 10 02 2022].

[11] F. Pfaffenberger, Twitter als Basis wissenschaftlicher Studien, Wiesbaden: Springer Fachmedien Wiesbaden, 2016.

[12] L. D.-X. A. B. C. N. S. Stieglitz, „Social Media Analytics: Ein interdisziplinärer Ansatz und seine Implikationen für die Wirtschaftsinformatik,“ in *Wirtschaftsinformatik, Band 56*, Heidelberg, Spring Link Heidelberg, 2014, pp. 101-109.

[13] J. Y. u. P. Z. J. L. Gross, Handbook of graph theory, London, England: CRC Press, 2018.

[14] NetworkX , „NetworkX Documentation,“ Networkx.org, [Online]. Available: https://networkx.org/. [Zugriff am 10 02 2022].

[15] M. E. J. a. G. M. Newman, „Finding and evaluating community structure in networks,“ 2003.

[16] M. E. J. N. C. M. Aaron Clauset, „Finding community structure in very large networks,“ *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, p. 066111, 2004.

[17] D. D. M. G. R. G. M. H. Z. N. D. W. Ulrik Brandes, „Maximizing Modularity is hard ⋆,“ Arxiv.org, 2006. [Online]. Available: https://arxiv.org/pdf/physics/0608255.pdf. [Zugriff am 12 02 2022].

[18] S. H. u. M. J. M. Bastian, „Gephi: An Open Source Software for Exploring and Manipulating Networks,“ International AAAI Conference on Weblogs and Social Media, 2009.

[19] J.-L. G. R. L. E. L. V. D. Blondel, „Fast unfolding of communities in large networks,“ *J. Stat. Mech.*, p. P10008, 2008.

[20] T. V. S. H. M. B. M. Jacomy, „ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software,“ *PLoS One*, p. e98679, 2014.

[21] T. P. u. R. S. . Shapira, Kafka - the definitive guide: Real-time data and stream processing at scale, Sebastopol, CA, USA: O’Reilly Media, 2021.

[22] Neo4j Graph Data Platform, „Graph data platform,“ 05 2020. [Online]. Available: https://neo4j.com/. [Zugriff am 11 02 2022].

[23] Readthedocs.io, „afka-python --- kafka-python 2.0.2-dev documentation,“ [Online]. Available: https://kafka-python.readthedocs.io/en/master/. [Zugriff am 11 02 2022].

[24] neo4j, „Export to Gephi - Neo4j Graph Database Platform,“ 2022. [Online]. Available: https://neo4j.com/labs/apoc/4.1/export/gephi/. [Zugriff am 11 02 2022].

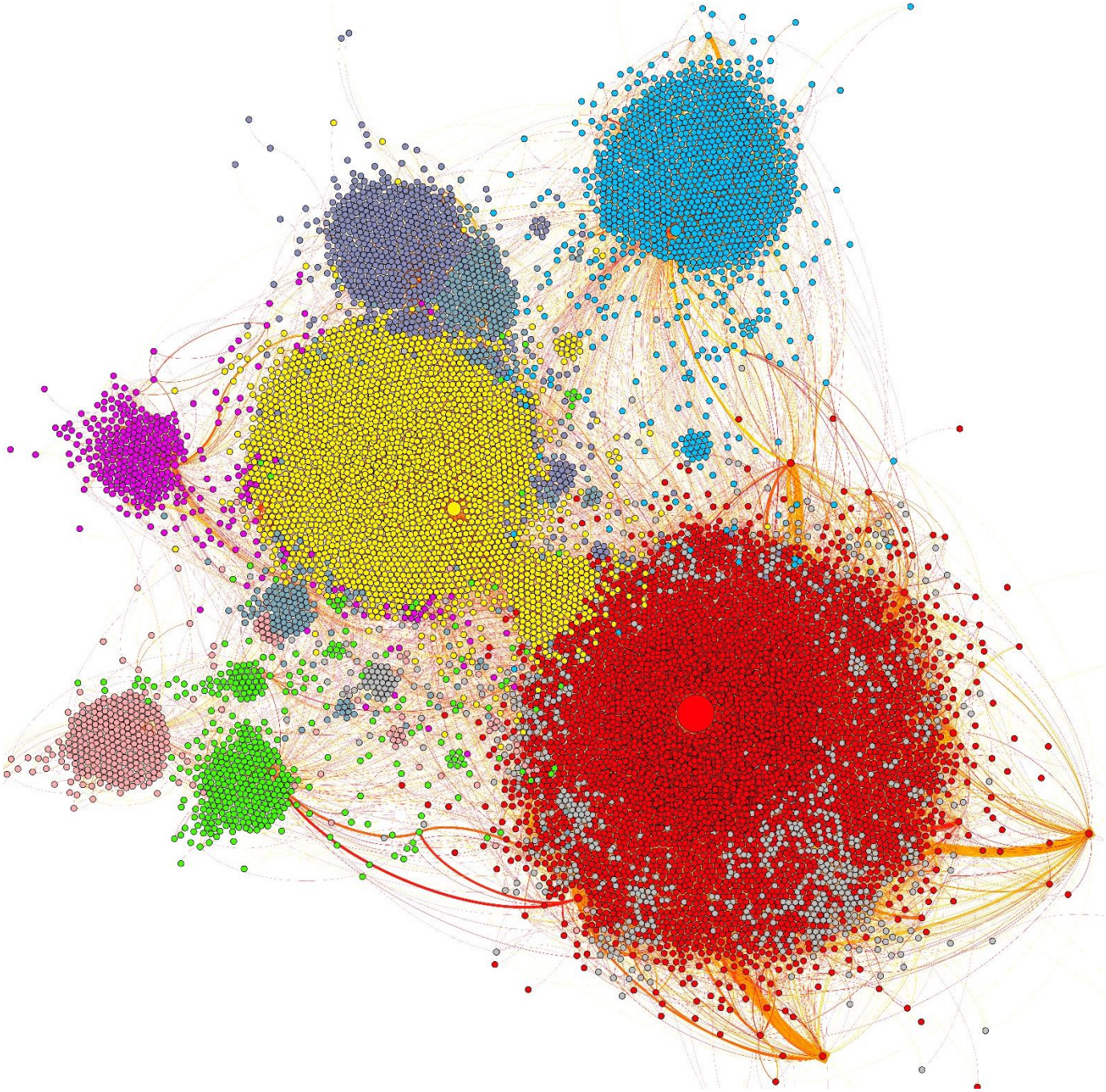


Abbildung 4 Negativ-Graph