UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

WeST People and Knowledge Networks
Institute for Web Science and Technologies

Time Series Influences in Political Communication

Masterarbeit

zur Erlangung des Grades eines Master of Science (M.Sc.)
im Studiengang Web Science

vorgelegt von
Tobias Thesing

Erstgutachter:	Prof. Dr. Steffen Staab Institute for Web Science and Technologies
Zweitgutachter:	Dr. Oul Han und Dr. Sarah de Nigris Institute for Web Science and Technologies

Koblenz, im Dezember 2019

Erklärung

Hiermit bestätige ich, dass die vorliegende Arbeit von mir selbstständig verfasst wurde und ich keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe und die Arbeit von mir vorher nicht in einem anderen Prüfungsverfahren eingereicht wurde. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium (CD-Rom).

	Ja	Nein
Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐
Der Text dieser Arbeit ist unter einer Creative Commons Lizenz (CC BY-SA 4.0) verfügbar.	☐	☐
Der Quellcode ist unter einer GNU General Public License (GPLv3) verfügbar.	☐	☐
Die erhobenen Daten sind unter einer Creative Commons Lizenz (CC BY-SA 4.0) verfügbar.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Anmerkung

- If you would like us to contact you for the graduation ceremony,

  please provide your personal E-mail address: .....

- If you would like us to send you an invite to join the WeST Alumni

  and Members group on LinkedIn, please provide your LinkedIn ID : .....

Zusammenfassung

Aktuelle politische Angelegenheiten spiegeln sich oft in Diskussionen in den sozialen Medien wider, wo Politiker and Wähler auf gemeinsamen Plattformen zusammentreffen. Da diese auch die allgemeine öffentliche Wahrnehmung der Politik beeinflussen können, besteht ein großes wissenschaftliches Interesse an den Funktionsweisen und Hintergründen solcher Debatten. Diese Masterarbeit befasst sich mit Nutzerbeiträgen aus einem aktuellen und relevanten Datensatz in Form von Zeitreihen, die dann auf thematische Inspiration und Themensetzung hin analysiert werden. Das *Institute for Web Science and Technologies* der *Universität Koblenz-Landau* hat Daten auf Twitter gesammelt, die im Vorfeld der Europawahl 2019 von den Kandidaten generiert wurden. In dieser Arbeit werden die Daten aufbereitet und auf unterschiedliche Eigenschaften hin untersucht, wobei der Fokus auf dem Einfluss von Politikern und Medien auf Online-Debatten liegt. Es wird ein Algorithmus vorgestellt, der Tweets in Themenstränge einteilt. Anschließend werden sequentielle Assoziationsregeln erstellt, die ein breites Spektrum an möglichen Einflussverhältnissen hervorbringen, sowohl zwischen Akteuren als auch zwischen Themen. Die erarbeitete Methodik kann mit verschiedenen Parametern angepasst werden und ist gut erweiterbar, was die Funktionalität und den Anwendungsbereich betrifft.

Abstract

Current political issues are often reflected in social media discussions, gathering politicians and voters on common platforms. As these can affect the public perception of politics, the inner dynamics and backgrounds of such debates are of great scientific interest. This thesis takes user generated messages from an up-to-date dataset of considerable relevance as Time Series, and applies a topic-based analysis of inspiration and agenda setting to it. The *Institute for Web Science and Technologies* of the *University Koblenz-Landau* has collected Twitter data generated beforehand by candidates of the European Parliament Election 2019. This work processes and analyzes the dataset for various properties, while focusing on the influence of politicians and media on online debates. An algorithm to cluster tweets into topical threads is introduced. Subsequently, Sequential Association Rules are mined, yielding wide array of potential influence relations between both actors and topics. The elaborated methodology can be configured with different parameters and is extensible in functionality and scope of application.

Contents

1. Introduction

Online communication and digital media have become more and more important over the past decades, even in politics [1]. Many politicians have realized by now that social media activity can be helpful not only for campaigning, but also in order to make their own political goals part of the public discussion and thus more relevant. As a byproduct, thereby generated digital data has opened up new opportunities for political and social research. [2]

The online platform Twitter is well suited for surveys about these phenomena, as it features high political activity and relatively easy access to data via API [3]. However, online debates based on short text messages can quickly get complex due to a high number of actors and interwoven messages, and analyzing them requires a well designed approach. In the field of *Computational Social Science*, efforts have been made to find the structure of these kinds of discussion on Twitter. Jungherr compared political communication on Twitter to the one on traditional media, but he found out that the logic and temporal dynamics are generally different [4]. Jackson and Lilleker analyzed the role and usage of Twitter in political communication and its actors [5]. However, both studies used datasets from 2009, and considering the rapid changes in social media usage, their findings might not apply today in the same way.

1.1. Motivation

This thesis aims to figure out who influences the topic and course of online debates, and how and when this is done. Those aims include *Agenda Setting* as a form of power in political discussion, which has recently been investigated in [6]. Rossiter's work focuses on traditional, structured debates, which are usually different from the Twittersphere, as they have a limited number of participants and a clear course of debate. Social media discussions on the other hand assemble multiple topic threads, where anyone can join or leave the conversation at any time.

To start further analysis, a large dataset of tweets is automatically reviewed for topics, authors and time frames of political Twitter messages. Connections between political tweets and temporally successive events and debates are key for this research. Therefore methods known from *Information Retrieval* and *Data Mining* are applied to a analyze a large number of political tweets, viewed as a time series, in order to look for potential *Temporal Association Rules*. By linking authors who repeatedly tweet about the same topics, association rules can tell which politicians tend to address similar topics, while the temporal aspect shows who was first and therefore likely to have brought an issue to public attention.

The found association rules and other observations are evaluated and processed. The elaborated methodology is supposed to be reproducible and adaptable for a variety of

political datasets.

1.2. Research Questions

As a guide to what to look for, I have defined the following research questions:

a Which politicians have the highest influence on online debates?

b Which politicians are able to cause response in common media?

c Do certain media sources preferably react to specific politicians or parties?

d Which topics are addressed and discussed by which political parties?

e What are the effects of upcoming elections on Twitter interaction, regarding frequency and order of tweets?

f Are there media outlets that work tightly together, e. g. influence each other?

While aiming to answer these questions, I look for the effects of various possible actors and factors, as follows:

- politicians (a, b)
- media (c, f)
- topics (d)
- external events (e)

2. Data

The study uses a dataset of political tweets that has been collected by the *Institute for Web Science and Technologies* of the *University Koblenz-Landau* [7]. The data has been live streamed from Twitter for an extended period of time, using *Tweepy* and *Twitter4J*, and is stored in a *MongoDB* database. Any program code that is part of this thesis is specifically designed to access this database as source.

2.1. Source

A *MongoDB* collection called "*deatpol*" contains the core data. It has been assembled using the accounts of all candidates for the *European Parliamentary Election 2019* as handles. All tweets from these accounts, as well as such that mention the account (for example in requests or answers) have been streamed into this collection from 2019-03-13 onwards.

In addition, there is another collection called "*german*" available that contains all tweets in German language posted within the same time frame.

A composed dataset containing only the relevant data - author, timestamp, content including hashtags and references - is analyzed in this thesis.

The composed dataset consists of:

- tweets contained in the *deatpol* collection
- tweets that have been referenced there and can be found in the *german* collection
- tweets from a list of media accounts [8] that can be found in the *german* collection

Referenced tweets are an important addition to the dataset for this research, as they often are part or even the origin of discussion topic chains.

Large parts of the analysis only concern the election phase, using tweets from 2019-03-13 through 2019-06-02 (one week after the election on 2019-05-26). This decision is further explained in the following sections.

The first and for this thesis later abandoned (see 3.2) and dataset contains tweets posted in a timeframe around the *German Parliamentary Election ("Bundestagswahl") 2017*, specifically 2017-07-05 to 2017-09-30. Tweets have been collected from a list of observed accounts, which contains politicians, political parties and other involved organizations. The politician list is compiled of candidates for all parties that made it into the German Bundestag (CDU/CSU, SPD, AfD, FDP, Die Linke and Bündnis 90/Die Grünen). The organization list contains party and faction accounts, as well as important media for political coverage. Media accounts can be seen as indicators for public interest in specific topics and events.

2.2. Temporal Distribution

Gathering profound knowledge of the dataset is key to plan and later interpret its analysis. The temporal distribution and density of data points is a core feature to look out for. In this case, I have counted the number of tweets in the composed dataset for each day. There are a total of 1.708.994 tweets in the focus timeframe of the election, going from 2019-03-13 to 2019-06-02.

Figure 1 shows the daily activity for these roughly 12 weeks of observation. The numbers appear relatively consistent over large parts of the time frame, averaging at 20.841 tweets per day. However, a major spike in late March is clearly visible. This is assumedly due to the discussion about the EU Copyright Reform [9] and its controversial article 13 (later article 17). This discussion was especially vivid on YouTube and Twitter, with the two highest peaks in activity being on March 23rd, the day when EU wide rallies regarding this issue were scheduled [10], and March 26th, the day when the official vote

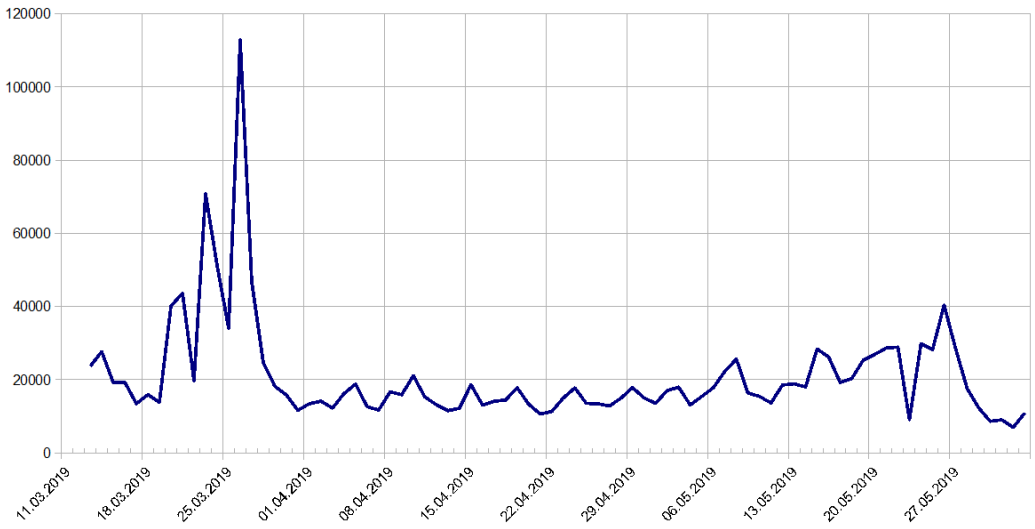

Figure 1: Daily Activity in Tweet Database

within the European Parliament took place [9]. This context is substantiated by the most used hashtags for the first two weeks of the data streaming being *#artikel13* and *#Uploadfilter*, both heavily related to criticism on the Copyright Reform.

Another visible, but much smaller peak is building up towards the actual election day on May 26th. The week before, there is a consistent increase of activity in the dataset, with the exception of May 23rd, which is most likely caused by technical issues and trouble with the university server's power supply. Overall activity drops quickly after the election, stabilizing at an average below 10.000 daily for the next two months. Taking into account that even more severe technical issues in streaming corrupted the available data of June and July, most of the analysis in this thesis will focus on the previously mentioned time period from March through May.

2.3. Trends

Knowing the most discussed topics in a time period helps to understand the context of the underlying data analysis. For this reason, I have extracted the most used hashtags of each calendar week fully covered within the focus timeframe. Table 1 shows the top 20 entries per week, starting with calendar week 12 (from 2019-03-18) and ending with calendar week 22 (till 2019-06-02).

Most of the hashtags found here can be assigned to at least one the following categories.

1) Hashtags can represent a party (e. g. *#CDU*) or an attitude towards a party (e. g. *#NieMehrCDU*). These are found in abundance and are usually not helpful as topic indicators. However, the frequency of a party or attitude hashtag can be affected by ongoing events and discussions, making them worth considering in the analysis. An interesting observation is the prominent ranking of *#AfD* in multiple weeks, which exceeds the relative presence and size of the party. Investigations within the database on the most frequent users of party hashtags have shown that the majority of the tweets with an *#AfD* hashtag are posted by accounts of the party themselves or their members, assumedly to boost its perceived relevance. Other parties show this tendencies as well, but to a far lesser degree. Interestingly, for *#SPD* the most common hashtag users are primarily not party members, but media accounts.

2) Hashtags can represent a person of public interest, like *#Merkel*. This either concerns a recent issue related to that person (see category 5) or is a comment to a statement made by them, likely outside of Twitter, hence the direct reply function can not be used.

3) Multiple trending hashtags in the observed timeframe directly refer to the upcoming European Parliamentary Elections (such as *#EUWahl2019* or *#gehtwählen*). While the election is clearly not only a relevant discussion topic, but arguably the most important one in this time, its omnipresence distorts its value for topic based analysis.

4) Hashtags can stand for a discussed topic of long term importance. These are particularly interesting, as they cover extensive discussions and can be brought back up by specific actors after losing relevance. Examples of this kind are *#Brexit* and the overall number one hashtag *#Artikel13* along with others related to the Copyright Directive. Only a few other abstract issues are represented in the top list by hashtags, such as Climate Change with *#FridaysForFuture* or Privacy with *#DSGVO*. However, when covering the full dataset, there are, of course, a lot more of this type (e. g. *#Mietpreisbremse*).

5) Some hashtags mark a very current event or affair which caused a burst of related tweets. Examples here are *#NotreDame* and *#Rezovideo*. As they have an impactful external event as common influence, agenda setting is not relevant here. Nevertheless it is very interesting to see which politicians post about the topic, acknowledging its relevance to them. A noteworthy information about this kind of hashtag is that names of locations (primarily cities) or even persons can stand as a proxy for a recent event.

2.4. Quality

While the dataset is very viable and represents not only important political topics, but many actors from different parties, there are several observable issues that reduce the overall suitability for further analysis.

Dealing with retweets is a major concern when examining Twitter data, especially in an

Table 1: Trending Hashtags per Calendar Week

Rank	**Week 12**	**Week 13**	**Week 14**	**Week 15**	**Week 16**	**Week 17**
1	#Artikel13	#artikel13	#artikel13	#AfD	#Uploadfilter	#AfD
2	#Uploadfilter	#Uploadfilter	#Uploadfilter	#Uploadfilter	#AfD	#ZiB2
3	#Artikel13Demo	#NieMehrCDU	#NieMehrCDU	#Cumex	#SPD	#Europawahl2019
4	#Demosold	#Uploadfillter	#NieMehrSPD	#urheberechtsreform	#NotreDame	#Vilimsky
5	#EP	#CopyrightDirective	#EU	#EU	#Artikel13	#Strasbourg
6	#NieMehrCDU	#EP	#NoPePP	#Lanz	#Europawahl	#Europawahl
7	#SaveYourInternet	#Article13	#Brexit	#Europa	#NieMehrSPD	#EU
8	#artikel17	#Uploadfilters	#Europawahl	#SPD	#NieMehrCDU	#EUWahl2019
9	#bots	#GehtWählen	#AfD	#Europawahl	#CDU	#puls4
10	#CDU	#Bots	#SaveYourInternet	#Brexit	#EU	#NordStream2
11	#Memes	#EU	#eprivacy	#FCKArt13	#Europawahl2019	#Uploadfilter
12	#axelsurft	#NieWiederCDU	#copyright	#Artikel13	#Artikel17	#CSU
13	#copyright	#saveyourinternet	#Artikel13Demo	#FPÖ	#Europa	#Europa
14	#EU	#Berlin	#dsgvo	#Europawahl2019	#CopyrightDirective	#Klimawandel
15	#brexit	#SPD	#SPD	#europadelbuonsenso	#Seehofer	#FDP
16	#Article13	#CDU	#Europa	#EAPN	#Urheberrechtsreform	#Klimaschutz
17	#Demogeld	#Brüssel	#niemehrcsu	#Enteignung	#FridaysforFuture	#EVP
18	#orban	#copyright	#Berlin	#OffenerBrief	#GroKo	#FPÖ
19	#yes2copyright	#Brexit	#Germany	#NieMehrCDU	#CSU	#CDU
20	#Fidesz	#gehtwaehlen	#FarmVille	#CDU	#Essen	#Weber

Rank	**Week 18**	**Week 19**	**Week 20**	**Week 21**	**Week 22**
1	#AfD	#Europa	#AfD	#AfD	#Uploadfilter
2	#JEC	#wahlarena	#tvDuell	#Europawahl2019	#Mussolini
3	#SPD	#EU	#TellEurope	#EP2019	#SPD
4	#Europawahl2019	#AfD	#europawahl2019	#EU	#ep2019
5	#Europawahl	#Europawahl2019	#Europa	#SPD	#AfD
6	#Europa	#SPD	#EU	#GreenWave	#Zensur
7	#1Mai	#Orban	#StracheVideo	#Europa	#Europawahl2019
8	#FlorenceDebate	#Europawahl	#EurovisionDebate	#Rezo	#Nahles
9	#EU	#Nazi	#AnneWill	#Europawahl	#CDU
10	#TagderArbeit	#germany	#strache	#Klimaschutz	#rezo
11	#Pressefreiheit	#terror	#FPÖ	#Rezovideo	#SPANIEN
12	#EU2019	#war	#Mailand	#CDU	#GreenWave
13	#EUWahl2019	#CDU	#Europawahl	#NieMehrCDU	#hartaberfair
14	#europaistdieantwort	#Bremen	#Vilimsky	#annewill	#Merkel
15	#Pforzheim	#Uploadfilter	#euwahl	#Kurz	#EU
16	#Artikel13	#Barley	#SPD	#Meuthen	#Union
17	#CDU	#Merkel	#Schlagabtausch	#FridaysForFuture	#Brandenburg
18	#EP2019	#Weber	#EP2019	#EUWahl	#greta
19	#Orban	#EP2019	#euwahl2019	#FPÖ	#Europawahl
20	#Kühnert	#Artikel13	#2iG	#EUelections2019	#Artikel13

environment of political communication. 47,6% of all tweets in the database (1094318 out of 2297593), and 52,3% of the tweets within the focus timeframe (859045 out of 1708994) are retweets. On one hand, retweets display a political opinion or message that should not be left out. On the other hand, retweets emerge as unproportionally predominant in analysis based on topics and time series, despite containing no original content. Retweets have the same assigned topics as their source and are often posted within a short time period, both of which favor not only prevalence, but also evaluation within temporal association rules of the involved topics and authors.

One extreme example of the impact of retweets is the hashtag ranked 20th on week 14 in table 1: *#FarmVille*. An unaware observer might assume some kind of hype around the Facebook game of that name, but in reality it was a single skit by parliamentarian Tiemo Wölken (*@woelken*). His post criticized the arrangement to let the agriculture department decide about copyright affairs and happened to be retweeted over 300 times.

However, retweet behavior and favorite sources are, although rather trivial, part of my analysis, so completely discarding them could not be justified. Instead, multiple runs with different parameters can help tackle the issue.

Another common problem when dealing with user generated content is flooding, where authors post in an unnecessary high quantity in an attempt to make their content more visible. Often combined with excessive overuse of hashtags, flooding can significantly skew the results of a study like this if it is not properly dealt with. A prime example is the user *@Crypto_Schurke*, who repeatedly posted tweets full of the same hashtags, addressed to different observed politicians. As a result, these hashtags are overrepresented in certain calculations, like association rules to find out which hashtags are often used together.

3. Methodology

In this section, the general approach as planned is described, as well as the detailed procedure during the Data Processing and Analysis part.

3.1. Approach

The work on this thesis is separated into two main tasks. Both of them need program code to process the large amount of data involved. As primary programming language, I use *Python (v3.7)*.

My first goal is to assign certain topics to each of the tweets. Therefore I transfer them from the eDemocracy database into a local database, using only the information I need, which are author, timestamp, topics and retweet or reply information. Since topics have to be mined from the tweet content text (which also includes hashtags and links), this is

also important to extract, but does not need to be directly stored any more once topic discovery is done.

The most basic way of assigning topics is using the hashtags. This can act as a simple solution to *Topic Discovery* in order to get started with the second and more important step. However, hashtags usually do not offer a very precise topic description, and many of the collected tweets do not contain any hashtags at all.

Once the tweets have been assigned topics, they can be combined into topic chains. These may include discussions using the response and retweet functions of Twitter, but more importantly new tweets that share the topic and are close in terms of time. Within a topic chain, *Temporal Association Rules* [11] are to be generated to see which politicians interact with or are inspired by each other. Each topic chain can also be seen as a *Time Series* [12], giving insight to the general activity and the frequency in which certain politicians and media are covering the topic.

For the generation of association rules, a suitable algorithm has to be found. Topics can be treated as transactions and authors as items. A simple example for an association rule of this kind is "if author A talks about topic X, then author B does so as well". To involve the time aspect, an important restriction is that the antecedent has to be earlier than the consequence in a transactions in order to support the rule.

In the reverse way, association rules can also be generated to see what topics are similar, since the same people talk about them. In this case, authors are transactions and topics are items. "Authors who tweet about topic A also like to tweet about topic B" would be an example of such a rule. While this does not directly serve to learn about agenda setting, it can help to refine the topic discovery by extrapolating similarity of topics from the politicians who put emphasis on them.

A secondary goal is to detect *Topic Drifts* [13] within discussion chains, and to determine who is responsible for these [6]. As a very recent example, the topic can drift from "copyright reform" to "freedom of speech", depending on the course of the debate and protests. Overlaps and reply activity involving two or more topic chains are indicators that a topic drift may have happened. Once a topic drift has been found, further investigation of a compiled time series can identify timeframe and responsible authors of the drift.

The last step is to process the results of the analysis and draw conclusions from comparing the effects of various factors and actors. This includes answering the research questions and backing up the answers with results, while considering possible sources that affect the development of Twitter conversations in the specific setting that this thesis examines. These sources are politicians, media, topics and external events. Their relative effect sizes can be compared by observing quantified topic chains and drifts.

In addition, I discuss any other findings that can be considered relevant to political and social research.

3.2. Data Processing

To handle more than one million tweets efficiently, the data is first processed and transmitted into a more convenient database. To accomplish this, I wrote a script, using *Python 3.7* as programming language and *Anaconda* with *Spyder* as IDE. The script utilizes modules for database connections, more specifically *pymongo* for the source database and *mysql* for the workspace database. To maintain a server connection via SSH, *sshtunnel* is employed, while the *bgtunnel* module helps to keep the connection between the two involved servers alive.

A Python class (*dbconnect.py*) handles connection and authorization for both databases. It has two ways to access the workspace database, remotely via SSH or locally when run directly on the server. The former allows easier testing on a personal machine, but the latter is a lot faster due to fewer remote connections. Therefore productive data transfer is done by running the script directly on the server.

In the data handler (*datahandler.py*) the tweets are read from the *MongoDB* source, processed when necessary, and written into a slimmer *mySQL* database. The processing includes the conversion of data formats such as timestamps, and the protection of key characters that otherwise would cause trouble within SQL statements. For each entry, a query is generated to insert the tweet into the workspace database.

The following columns are kept in the *Tweets* table:

- id: The ID of the tweet as used in the Twitter API [3].
- time: The date and time the tweet has been posted.
- author: The screen name of the user who posted the tweet.
- text: The actual content of the tweet, including hashtags, links and all text; for tweets that are marked as truncated, the full text is loaded.
- retweet: If the tweet is a retweet, the ID of the original tweet is saved here; value 0 means it is not a retweet.
- reply: If the tweet is posted in reply to another tweet, the ID of that tweet is saved here; value 0 means that the tweet is not posted as a reply.
- quote: If the tweet contains a quote and is posted as retweet with comment, the ID of the quoted tweet is saved here; value 0 means that there is no quoted tweet.
- flags: An extra value for internal use during the analyzing process.

An issue worth mentioning is that the dataset from the 2017 German Federal Election, which was the original object of study, was assembled during the time where Twitter started to loosen the 140 character limit [14]. After some investigation, I found out that

this has resulted in a significant number of tweets being truncated without having the full text stored in the database (see D). This made the dataset unusable for the purpose of this thesis, and the consequent decision was to focus on only the new dataset instead, based on the 2019 European Parliament Election.

The data structure of the new dataset in the *MongoDB* source is different from the old one, as it is directly based on the Twitter API [3]. This required a few adaptations within the data handler script, making it now incompatible with the old and for this purpose obsolete dataset.

In addition to the transfer, the script parses the text of all tweets and extracts hashtags from them. While the source database has hashtags available as a separate field, I found out that the data provided there is incomplete. Specifically, hashtags appear to not be detected whenever they are in the second part of a truncated text. For this reason, direct parsing from the full text is the best option, although it comes with further efforts. The found hashtags are stored as hashtag-based topics in a specific table (*Topics*), and linked to the tweets that contain them in another one (*Links*). Hashtag-based topics are optimized by removal of noise characters.

In the next step, referenced tweets are added to the database. To achieve this, the script takes all retweet and reply IDs from available political tweets and subtracts the the IDs already in the database from this list. The remaining entries represent original tweets that have been replied to or retweeted from by observed politicians, but have not been transferred to the database yet. The *MongoDB* server contains a collection with all German language tweets from the relevant time frame. By iterating through that collection, some of the missing entries can be retrieved and transferred to the database.

However, a missing tweet might not be in the collection for the following reasons, listed in order of the predicted number of cases: a) the original tweet is not in German language; retweeting of English language content is not uncommon in German political discussion; b) the original tweet is too old and has been posted before the collection timeframe; c) the collection of German tweets missed out on that particular entry.

To include tweets from relevant media outlets, as it has been done in the original dataset, I requested the list of account handles from the corresponding project [7]. All tweets posted by these accounts that could be found in the above mentioned full German collection are also copied into the workspace database.

Due to optimization or corruption of relevant data, multiple iterations have been run to create a suitable database excerpt. Details and numbers regarding these iterations are briefly described in the appendix (E).

3.3. Analysis

Definition 1 *A Tweet (or Post) p in the database has an author a(p), a timestamp t(p) and a set of Topics (or Hashtags)* $H(p) = \{h_1(p), h_2(p), ...h_n(p)\}$.

As a basis for the analysis, I created a Python class (*analysis.py*) with different functions for extracting useful information.

The first step is to find all topics a specific author is involved in and how often he is tweeting about each, and to find out which authors have talked about a given topic and when. Both can be achieved by relatively simple SQL queries.

One use of association rules on the dataset is an attempt to find similar topics. It implements the *Frequent Pattern Growth* algorithm and employs the *pyfpgrowth* Python module. With tweets as transactions and topics as items, some meaningful association rules can be found, depending on required support and confidence. Pure retweets are ignored for this measure to prevent support and confidence from becoming skewed.

However, the focus of this research and more complex endeavor is to find out which authors are influenced by each other. This requires at first for each topic a list of all discussing authors, including time stamp of their tweets. These lists are here defined as *Topic Chains* and can be seen as sequential transactions in an effort to find association rules.

Definition 2 *A* ***Topic Chain*** *C is a sequence of Tweets* $p_1, p_2, ..., p_n$ *where* $(\forall p_i \, \exists h \in \bigcap_{H(p_1),H(p_2),...,H(p_n)}) \wedge (t(p_i) \leq t(p_{i+1}))$.

There has been plenty of research on terms of *Temporal Association Rules*, but none of the methods I found is optimal for the purpose of this master thesis. Worth considering was the classic approach by Rainsford and Roddick [15] and the extending ARMADA algorithm [16], however it is heavily focused on interval-based data, while the Twitter database only provides point-based items. I made deliberations for constructing intervals by linking tweets of the same author and topic in close temporal proximity, but decided that this process could quickly get too inconsistent and arbitrary.

As a result, I opted for an efficient way to determine influences, by employing *Sequential Association Rules* [17] and using the full time stamp only to determine topic chains. As gathered from the database, the discussions on the most viral hashtag-based topics contain thousands of tweets, which is far from an optimal size of a sequential transaction in association rule mining. This issue is tackled by the introduction of a time delta.

Definition 3 *A maximum interval Δt splits a Topic Chain C into a set of* Subchains $C_S = \{S_1, S_2, ..., S_n\}$. *A split between two Tweets p_i and p_{i+1} happens iff $t(p_i) + \Delta t \leq t(p_{i+1})$.*
Vice versa, a Subchain S_i fulfills the requirement $t(p_i) + \Delta t \geq t(p_{i+1})$ as well as the definition of a Topic Chain.

Topic chains can be split into subchains in different ways, depending on the time delta. For this reason, my function for creating sequences works with a flexible parameter that defines the time delta in multiples of 1 hour. All subchains are treated equally after the splitup, in the same way normal topic chains are, regardless of the underlying topics. To learn sequential association rules from the generated chains, an external tool called SPMF [18] is employed. SPMF is published and kept up to date by Fournier-Viger, one of the developers of the CMRules algorithm [17] used here. As it only works with integer numbers as items, authors are translated into an ID for the calculation and reverted later. When calling SPMF, a minimum support and confidence can be chosen, whereas the support is interpreted as a fraction relative to the whole dataset and therefore must be lower than the confidence.

In total, there are four variable parameters to the process that can influence the resulting association rules.

1. The **Selector Mode** determines which tweets are considered for the calculation of association rules - either all tweets, all original tweets (no retweets), all tweets except those from the media and organization handle, or all original tweets except those from that handle.

2. The **Time Delta** in hours sets the maximum time interval between any two tweets in a chain, as described above.

3. The **Minimum Support** tells SPMF which percentile of Topic Chains is required to support a rule in order to be considered.

4. The **Minimum Confidence** tells SPMF which percentile of Topic Chains where the antecedent occurs also have to contain the consequent in order to be considered.

Depending on these parameters, various results can be observed (see 4.2).

4. Results

This section covers and discusses all findings arising from the analysis.

4.1. Association Rules for Topics

The first and most simple application for association rules are the topics themselves. Especially for hashtag-based topics it is interesting to see which are commonly assigned

jointly to a tweet.

Using a standard FPGrowth algorithm with a minimum support of 100 and a required confidence of 0.5, a total of 57 associations appear. Some link the numerous hashtags about the EU Copyright Directive, for example *(#SaveYourInternet, #Uploadfilter) -> (#Artikel13)*. Many of the rules are nevertheless different combinations around a *#cannabis* hashtag, which are attributable to a single user, a flooding issue already described in the Data section (2.4). These rules are overshadowing the ones that are actually meaningful, like *(#May) -> (#Brexit)* or *(#NieMehrSPD, #NiemalsAfD) -> (#NieMehrCDU)*.

The impact of flooding can be reduced by basing the association rule mining on authors, rather than single tweets. This is very similar to a typical recommender algorithm, as in "authors who posted about topic A, also posted about topic B". In this case, the previous settings for minimum support and confidence yield too many rules, so more restrictive values are applicable to cut their quantity in favor of quality, thus stronger, more meaningful association rules. With 500 for support and 0.8 for confidence as minimal requirement, 189 rules emerge. The results can provide more insight into hashtag-based topics, showing which ones are possibly related, for example *(#eprivacy) -> (#DSGVO)*, *(#Campact) -> (#Attac)* and (#Germany, #terror, #war) -> ((#Nazi).
All association rules for topics can be found in the appendix (A.1).

4.2. Sequential Association Rules for Authors

Sequential association rules to establish who might influence whom are the primary focus of this thesis.

The minimum support is set to 0.05%, meaning that for every 2000 transactions, at least one must contain the observed items. This appears low on first glance, but given the diversity and vast amount of actors in Twitter discussions, test runs have proven that higher numbers are insufficient to get any results. The minimum confidence is 50%, providing the latitude for rules while keeping a "more often than not" level of validity.

The full tables of mined association rules can be found in the appendix (**??**).

With a standard selector mode and a time delta of 24 hours, a total of 2422 rules have been found. Authors in both the antecedent and the consequent repeat a lot, indicating networks of people and organizations who share interest in topics. However, on a closer look to some of the most common accounts found in the consequents, it appears that the vast majority of their posts are retweets. While this grants an interesting insight on who favors which party based on retweets, figuring this out could be done with a lot less effort by directly using twitter mechanics.

Reducing the time delta to 3 hours significantly lowers the number of output rules, to a total of 108. The reason for this is that a smaller time delta means an increase in the number of topic chains, which also increases the minimum support required, as it is relative to the total number of transactions. Meanwhile, topic chains are shorter, many of them reduced to a single entry and therefore impractical for rule mining.

When ignoring retweets while going back to a 24 hour time delta, there are still 807 rules. All of the earlier observed common accounts have been eliminated, proving that these networks are in fact primarily based on retweets. Nevertheless, within this setting media accounts predominate the generated association rules. Since media strive to post as early as possible about current topics, a relation based on discussed topics within a close time window is not surprising. Association rules between different media outlets are found with a time delta as low as 3 hours, but their number then drops significantly, down to a mere 9 rules.

Running the script while excluding both retweets and media handle accounts does not result in any rules found, for time deltas of both 24 hours and 3 hours.

A lower minimum support might provide better results here, but unfortunately SPMF freezes when using these settings, which occurred consistently and reproducibly at my system. In the Performance section of the SPMF homepage [18], author Fournier-Viger presented some test runs, and the plots show that the CMRules algorithm indeed can stop working if the minimum support is too low. He also suggests that an algorithm called RuleGrowth "is much more efficient" [18] in these cases. RuleGrowth has been elaborated by Fournier-Viger et al [19] and is based on a pattern-growth approach, specifically aimed at improving performance for large scale datasets with relatively low minimum support requirements.

Repeating the mining with exclusion of both retweets and media handle accounts with RuleGrowth and a minimum support of 0.02% results in 1370 sequential association rules for a 24 hour time delta (see A.2, Run 8). Within these, there are still media accounts left (since they are not only collected by the separate handle, but also when answered to or retweeted from within the original parliamentarian candidate handle), but they are no longer predominating. Many names show up repeatedly in both the antecedent and the consequent, but it is important to consider that the absolute numbers for rule support in this run can be as low as 10.

4.3. Agenda Setting and Inspiration Phenomenon

To get an overview on who is most likely to set the topic of discussion, a function in my script shows, for each author in the antecedent, all distinct other authors that appear in a corresponding consequent in any rule. The result is a set of everyone who repeatedly talked about the same topics as the respective author shortly after. A very basic

measurement of agenda-setting power as described by Rossiter [6] can be obtained by counting the number of elements in the set of authors who follow the same topics. This can be seen as a measurement of how many others an author has potentially inspired. Table 2 shows these numbers for the previously discussed run with a 24 hours time delta and exclusion of both retweets and accounts from the media handle, which I found to be the most interesting.

Of course, these results have to be taken with caution. The exact confidence and support of individual association rules are not considered, and flooding can, though not being as impactful as in other calculations, skew the rankings. Author *@JfVec1* for example shows tendencies towards flooding, while also participating normally in discussions on the other side. In general the ranking is predominated by the discussion about the copyright directive, which has by far the highest involvement within the observed timeframe, as established in the Data section (2.2).

That being said as a disclaimer, the ranking indeed points out active and dedicated politicians who one can give credit for steering the online conversations. Top ranked author Sven Giegold (*@sven_giegold*) of the Green party has posted an average of 2.75 tweets per day with only 9% of them being retweets. He also used hashtags appropriately, allowing for an extensive assignment of relevant topics to his tweets.

The full ranking lists can be found in the appendix (A). Looking at the general distribution of inspired peers per author, a similarity can be observed between the aforementioned ranking and that of an unfiltered selection (including retweets). In spite of individual entities being ranked completely different, both plots pictured in Figure 2 share characteristics with a typical Zipf distribution, featuring only a few leading elements and a long tail. The numbers here are not of a sufficient magnitude to prove or further investigate this hypothesis, but considering the Zipf distribution is a pattern observed frequently in the area of user generated content and network dynamics, it is both notable and plausible nonetheless.

4.4. Summary

Table 3 shows a summary of all recorded mining runs, including parameters, number of generated rules and reference to the appended table.

4.5. Caveats

While the two observed main sources of rules - retweets and media - had to be expected and only provide limited information, they prove that the general method works as intended. However, without explicitly eliminating these sources, there is a significant lack of rules that reveal agenda setting and inspiration dynamics. The possible reasons for

Table 2: Number of Inspired People per Author (Top 30)

Author	**Inspired**
sven_giegold	23
Lars9596	20
ZDFheute	20
OggoleMurx	16
drjdvalentin	16
woelken	13
JfVec1	12
KrahMax	12
BytePirat	11
PhilHackemann	10
FritzFizz	10
JuttaPaulusRLP	10
Typo87	8
dneuerer	8
schdrahlemann	8
jan_buehlbecker	8
michabl	8
guidoV4	7
nicolabeerfdp	7
AfD	7
watch_union	6
Gruene_Austria	6
Tagesspiegel	6
2GRIMREAPER3	5
Langhaar_Andy	5
nichtvermietbar	5
RicoTV7	5
SCHIEDER	5
faznet	5
bueti	5

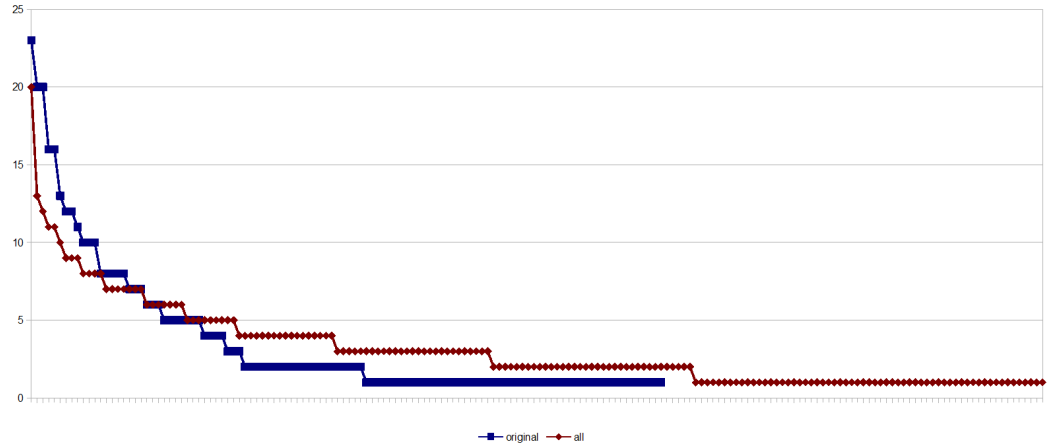

Figure 2: Agenda Setting Power Distribution

Table 3: Summary of Recorded Sequential Association Rules Mining Runs

Selector	Δt	***minSup***	***# Rules***	***Table Ref***
all	24	0.05%	2422	A.2, Run 1
all	3	0.05%	108	A.2, Run 2
original	24	0.05%	807	A.2, Run 3
original	3	0.05%	9	A.2, Run 4
original, base	24	0.05%	0	-
original, base	3	0.05%	0	-
original, base	168	0.05%	7	A.2, Run 7
original, base	24	0.02%	1370	A.2, Run 8
original, base	3	0.02%	4	A.2, Run 9
all	12	0.05%	466	A.2, Run 10
original, base	12	0.02%	54	A.2, Run 11

Selector: all -> no exclusions; original -> no retweets; base -> no extra media handle

this are the following:

a) Topic Discovery

One problem is the simple topic discovery method based on hashtags, resulting in 44.075 different topics for the observed timeframe. Having a large number of different topics helps to generate different topic chains without restrictive time deltas, but also impairs the meaning of these chains, as tweets related to a topic might be left out because they do not feature the exact same hashtag. It also favors the predominance of retweet-based topic chains, as in these cases the hashtags are directly copied and thus the same.

More importantly, 72% (1.231.933 out of 1.708.994) of the tweets do not have a topic assigned as they do not feature any hashtags. This effectively reduces the sample data to almost one quarter of the original size, leaving only 477.061 tweets usable for topic-based analysis.

While the methodology of this thesis and the implementation of the corresponding script is specifically designed in a way that makes it easy to improve topic discovery, an effective way that works on short messages has yet to be found. This issue is discussed in more detail in the Discussion section (5.1).

b) Data composition and density

In addition to the problems pointed out in the Data section (2.4), there is another caveat when it comes to association rule mining. Within the timeframe of investigation, there are 1.708.994 coming from 221.526 different author accounts. Ignoring tweets without an assigned topic, these numbers are reduced 477.061 to and 188.511. The problem here is that there are only 2.5 tweets per author on average. Topic Chains are assembled with tweets and need at least two of them to be meaningful, and although tweets can possibly have multiple topics assigned and therefore be part of multiple Topic Chains, this is not common. That leaves a very small number of transactions relative to the number of items, resulting in a more difficult association rule mining.

5. Discussion

This section covers possible improvements, including reasons why they have not been incorporated and ideas on how to approach them.

5.1. Topic Discovery

The major point of possible improvement in my methodology is the assignment of meaningful topics to tweets. Hashtags are a solid base for that, since they are purposefully

used by authors to indicate that their tweet is about a certain topic. This is much more precise and reliable than an algorithm could be for short messages. However, as mentioned earlier, many tweets do not contain hashtags and therefore lack any option to get a topic assigned based on them. In addition, several popular hashtags are not expressive when it comes to pinpoint a topic, which is discussed in detail in the Data section (2.3).

There has been a lot of research on topic discovery for texts, but general statistical topic models do not work well on short messages as seen in Twitter. For this reason, I have looked into multiple research papers specifically aimed at short texts in prospect to find a suitable method for advanced topic discovery within the database of political tweets.

The processing of the *TweetsKB* project by Fafalios et al [20] stores tweets in an RDF database and uses DBpedia to classify semantics. When considering the use of this method, two major problems occur. First, the data processed in this study is mostly German, and while there is a German version of DBpedia, it is not on the same level as the English one. Considering we are dealing with local politicians and up-to-date topics, some information is expected to be missing. Second, the effort for entity linking and triple generation exceed the scope of a master thesis, as for the creation of TweetsKB, a "Hadoop cluster [...] of 40 computer nodes" [20] has been used.

Other approaches for topic discovery in Twitter have been elaborated by Zhao et al [21] and Ramage et al [22]. Both of these are, as I concluded, too complex for my desired results, as they focus on advanced things like multi-category labeling or content summary. While these features might definitely be interesting for further investigations, my goal was a simple topic classification.

More promising was the work of Rosa et al [23]. While they do not provide a direct instruction or algorithm, they explain the methods they used and the quality of their results. They also tackle issues that are relevant for my work, such as temporal topic drift and linked documents. However, important was in particular "fine grained (hashtag) classification" [23], which they claim is not performing as well as the "coarse" one using just a few general categories. In the last part, they also summarize clusters into stories, which would fit my purpose very well. The problem here was that they defined stories manually as they were "not interested in the task of cluster decomposition" [23], and they hired workers on Amazon Mechanical Turk for the relevance judgments. Both of these steps were too extensive and not feasible for my thesis.

Lastly, Xun et al [24] discuss topic discovery via *Word Embeddings*. Briefly summarized, they employ vector space models, but instead of traditional word types, they use word embeddings, which are vector representations themselves. They use Wikipedia to learn these word embeddings before applying them to the words in messages. Additionally, they strive to identify "background" words with no relevant topic semantics. An important constraint of this method is that it is only able to assign one single topic per

message. This restriction by itself would be, although not favorable, acceptable for my research.

More of a problem is the fact that the clustering process is hardly compatible with my topic assignments. The algorithm needs a fixed number of topics to assign messages to, but without providing the option to set what topics are to be excepted. This might be sufficient when it comes to the simple sorting of messages, but here, topics not only need to be aligned with the hashtag-based topics, but are also used as a basis for rule mining. Additionally, the document support for training German language word embeddings is not as sophisticated as that for English ones, especially considering that many of the expected topic buzzwords are novel or trending terms. While these word embeddings would have been the most promising approach for topic discovery if hashtags or similar mechanisms did not exist, the expected improvement of employing them as an additional indicator for topics was not worth the effort.

Concluding this section, the field of topic discovery emerges as vastly difficult and imprecise, due to the length and nature of the observed tweets. Therefore, based on the above aspects, hashtag based topics seem to be most efficient in order to focus on the core subject of this thesis.

5.2. Filtering

One way to improve the validity of the results is to clean up the dataset itself. Obvious spam messages have been found significantly less in samples of the effective 2019 dataset, compared to those of the old dataset from the German Federal Election 2017. However, flooding (see 2.4) is still an issue, as it causes messages with low meaning in terms of content and popularity to have a relatively large impact on analysis techniques.

A straight-forward way to tackle this is to find tweets that are very similar and posted by the same author, and mark them as "clones". Other methods can be employed directly during the analysis. When looking which authors have posted how many tweets about a certain topics, a threshold (e. g. 50%) can be fixed for the tweets fraction from a single author. If this threshold is exceeded, it is then very likely that the topic is not as relevant for discussion as raw numbers suggest, because a large part of the related tweets come from a single person.

However, while both methods would be capable of eliminating certain cases of observed flooding, they require additional processing and might produce false positives and false negatives, depending on their parameters. In addition, they only work for single account sources, which impact just the direct tweet-based association rules for topics.

5.3. Communication Structure

Twitter as a platform features three possible options how a tweet can be directly related to another tweet. A *Response* connects a tweet to the one it refers to, creating a discussion thread. A *Retweet* copies the exact content of another tweet while also linking to the original. A *Retweet with Comment* or *Quote* is basically a compound of response and retweet. Accompanied by a new message, the content of the original tweet is linked and displayed, but technically not part of the new tweet.

All information about this structure is carried over during data processing (see 3.2), storing the ID of the original tweet if applicable. It is used mainly to retrieve tweets that are referred but not found in the base dataset, and to filter out retweets for certain applications. Furthermore, another primary reason for this data to be preserved is its meaningfulness for communication analysis. This thesis focuses on finding implicit connections via assigned topics and association rules, but these can be enriched or matched with the explicit links in the Twitter communication structure.

The explicit connections are not incorporated into the analysis however, since they could possibly overshadow or distort insights gained by topic- and time-based association rule mining. The most viable way to include explicit connections would be supplementing topic chains with authors who took part in a discussion without their tweets being recognized as regarding the topic. For this, finding a well balanced way to establish such hybrid topic chains is an extensive, but essential task.

6. Conclusion

The final section summarizes the research in this thesis.

6.1. Answers to Research Questions

At this point, it can be evaluated which of the guiding research questions have been answered and where.

a) Which politicians have the highest influence on online debates?
The most applicable measurement to answer this questions is the inspiration ranking discussed in 4.3, according to which Sven Giegold has the highest influence.

b) Which politicians are able to cause response in common media?
To answer this question, an association rules table that includes media accounts can be consulted. Interestingly, there are a lot of association rules with media accounts in the consequent, but they almost exclusively contain other media accounts in the antecedent. When reviewing rules from a table without tweets from the additional media handle (4.2 and Table 3, Run 8), media tweets remain only if they have interacted with those of the

observed politicians. *@ZDFheute* in particular is a consequent for multiple politicians listed in the inspiration rankings (Table 2), amplifying its validity.

c) Do certain media sources preferably react to specific politicians or parties?
As mentioned above, media accounts mostly react to other media accounts, as they are presumably all animated by outside events. Within the generated association rule tables, this question can unfortunately not be answered.

d) Which topics are addressed and discussed by which political parties?
The research focus of this thesis goes towards connections between either different authors or different topics. General statements about favorite topics of an author can be confirmed by database queries, however authors in the underlying source data are not technically assigned to a party.

e) What are the effects of upcoming elections on Twitter interaction, regarding frequency and order of tweets?
Daily activity increases close to the election date and decreases significantly after that, as established in 2.2. This is also the main reason why the research is focused on the timeframe before the elections. In addition, many tweets are tagged with references to the then upcoming elections, resulting in multiple trending hashtags, which are closer described in 2.3.

f) Are there media outlets that work tightly together, e. g. influence each other?
When mining association rules with excluded retweets and 0.05% minimum support, connections between media accounts are the only rules remaining (4.2 and Table 3, Run 3). This indeed suggest that they post about the same topics while also using the same hashtags. In the third column of the full inspiration power rankings (A.3), the most influential media accounts are led by *@faznet*, *@tagesschau*, *@zeitonline* and *@ZDFheute*. Unlike the two others, this ranking does not resemble a Zipf distribution, but an almost linear one, indicating that there is no direct inspiration phenomenon, but simply a consensus on what topics - including their hashtags - are relevant.

6.2. Summary

Although not all concerns could be resolved, the hereby concluded research brought profound insight into the German political twittersphere beforehand the European Parliament Election 2019. Findings are of contentual, structural and technical nature.

Contentual findings include everything learned about topics and authors in this particular timeframe of political discussion. The prevalence of the copyright debate runs like a common thread through all research steps, from temporal peaks in tweet numbers and trending hashtags to well supported topic association rules. Influential politicians

emerged from both author association rule tables and inspiration power rankings.

Structural findings teach about the general setup of a political environment on Twitter. Similarities between media and their distinction from other accounts have been observed, as well as rates of retweets and the usage of hashtags. Potential problems like flooding and the importance of considering the general communication structure, specifically the impact of retweets, came to attention.

Technical findings consist in the application of different methods and their outcomes. Although efforts have been made, automated topic discovery could not be refined to an adequate level. Existing models are overall evaluated as not precise enough to use them for the sensitive clustering of very short texts into transactions. The generation of topic chains and the application of association rules to both topics and authors worked overall as intended, but there is still potential for improvement.

Acknowledgements

At this point I would like to thank my supervisors - first, **Prof. Dr. Steffen Staab** who entrusted me with this interesting topic and thus made this master thesis possible in the first place. Special thanks also go to **Dr. Sarah de Nigris** and **Dr. Oul Han** for their reviews and suggestions, as well as their time to discuss progress and issues related to my master thesis in numerous meetings. In addition I want to express my gratitude to the University Koblenz-Landau and the Institute for Web Science and Technologies; to the teachers and tutors I learned from, and to the fellow students I worked with in this exciting degree course. Last but not least, thank you to my parents for continuously supporting me throughout my studying and my work on the thesis.

References

[1] Karolina Koc-Michalska, Darren G Lilleker, Alison Smith, and Daniel Weissmann. The normalization of online campaigning in the web. 2.0 era. *European journal of communication*, 31(3):331–350, 2016.

[2] Maurice Vergeer. Politics, elections and online campaigning: Past, present... and a peek into the future. *New Media & Society*, 15(1):9–17, 2013.

[3] *Twitter API Documentation*, 2018. Last accessed: 2019-12-06.

[4] Andreas Jungherr. The logic of political coverage on twitter: Temporal dynamics and content. *Journal of communication*, 64(2):239–259, 2014.

[5] Nigel Jackson and Darren Lilleker. Microblogging, constituency service and impression management: Uk mps and the use of twitter. *The journal of legislative studies*, 17(1):86–105, 2011.

[6] Erin Rossiter. *Measuring Agenda-Setting Power in Political Discourse*. PhD thesis, Washington University in St. Louis, Department of Political Science, 2019.

[7] Sebastian Stier, Arnim Bleier, Malte Bonart, Fabian Mörsheim, Mahdi Bohlouli, Margarita Nizhegorodov, Lisa Posch, Jürgen Maier, Tobias Rothmund, and Steffen Staab. *Systematically monitoring social media: The case of the German federal election 2017*, volume 2018/04. 2018.

[8] Sebastian Stier, Arnim Bleier, Malte Bonart, Fabian Mörsheim, Mahdi Bohlouli, Margarita Nizhegorodov, Lisa Posch, Jürgen Maier, Tobias Rothmund, and Steffen Staab. Social media monitoring for the german federal election 2017 (dataset), 2018.

[9] John Schranz. European parliament approves new copyright rules for the internet. `https://www.europarl.europa.eu/news/en/press-room/20190321IPR32110/european-parliament-approves-new-copyright-rules-for-the-internet`, 2019. Last accessed: 2019-12-06.

[10] Markus Reuter. Demos gegen uploadfilter: Alle zahlen, alle städte. `https://netzpolitik.org/2019/demos-gegen-uploadfilter-alle-zahlen-alle-staedte/`, 2019. Last accessed: 2019-12-06.

[11] Sotiris Kotsiantis and Dimitris Kanellopoulos. Association rules mining: A recent overview. *GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006*, pages 71–82, 2006.

[12] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *KDD*, volume 98, pages 16–22, 1998.

[13] Damianos P Melidis, Myra Spiliopoulou, and Eirini Ntoutsi. Learning under feature drifts in textual streams. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 527–536. ACM, 2018.

[14] Aliza Rosen and Ikuhiro Ihara. Giving you more characters to express yourself. `https://blog.twitter.com/official/en_us/topics/product/2017/Giving-you-more-characters-to-express-yourself.html`, 2017. Last accessed: 2019-12-06.

[15] Chris P Rainsford and John F Roddick. Adding temporal semantics to association rules. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 504–509. Springer, 1999.

[16] Edi Winarko and John F Roddick. Armada–an algorithm for discovering richer relative temporal association rules from interval-based data. *Data & Knowledge Engineering*, 63(1):76–90, 2007.

[17] Philippe Fournier-Viger, Usef Faghihi, Roger Nkambou, and Engelbert Mephu Nguifo. Cmrules: Mining sequential rules common to several sequences. *Knowledge-Based Systems*, 25(1):63–76, 2012.

[18] Philippe Fournier-Viger. Spmf - an open-source data mining library v2.40. `http://www.philippe-fournier-viger.com/spmf/`, 2019. Last accessed: 2019-12-06.

[19] Philippe Fournier-Viger, Roger Nkambou, and Vincent Shin-Mu Tseng. Rulegrowth: mining sequential rules common to several sequences by pattern-growth. In *Proceedings of the 2011 ACM symposium on applied computing*, pages 956–961. ACM, 2011.

[20] Pavlos Fafalios, Vasileios Iosifidis, Eirini Ntoutsi, and Stefan Dietze. Tweetskb: A public and large-scale rdf corpus of annotated tweets. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 177–190, 2018.

[21] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. Topical keyphrase extraction from twitter. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 379–388, 2011.

[22] Daniel Ramage, Susan Dumais, and Dan Liebling. Characterizing microblogs with topic models. In *Fourth international AAAI conference on weblogs and social media*, 2010.

[23] Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. Topical clustering of tweets. *Proceedings of the ACM SIGIR: SWSM*, 63, 2011.

[24] Guangxu Xun, Vishrawas Gopalakrishnan, Fenglong Ma, Yaliang Li, Jing Gao, and Aidong Zhang. Topic discovery for short texts using word embeddings. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1299–1304. IEEE, 2016.

A. Association Rule Tables

A.1. Association Rules for Topics

Based on tweets:

```
{('#Cannabissteuer',): (('#cannabis', '#jugendschutz', '#
    legalisieren'), 1.0),
('#ca3mrdEuro',): (('#Cannabissteuer', '#cannabis', '#
    jugendschutz', '#legalisieren'), 1.0),
('#jugendschutz',): (('#cannabis', '#legalisieren'),
    0.9826086956521739),
('#legalisieren',): (('#cannabis',), 0.9743589743589743),
('#cannabis',): (('#legalisieren',), 0.912),
('#Cannabissteuer', '#ca3mrdEuro'): (('#cannabis', '#
    jugendschutz', '#legalisieren'), 1.0),
('#Cannabissteuer', '#jugendschutz'): (('#cannabis', '#
    legalisieren'), 1.0),
('#ca3mrdEuro', '#jugendschutz'): (('#Cannabissteuer', '#
    cannabis', '#legalisieren'), 1.0),
('#Cannabissteuer', '#legalisieren'): (('#cannabis', '#
    jugendschutz'), 1.0),
('#ca3mrdEuro', '#legalisieren'): (('#Cannabissteuer', '#
    cannabis', '#jugendschutz'), 1.0),
('#Cannabissteuer', '#cannabis'): (('#jugendschutz', '#
    legalisieren'), 1.0),
('#ca3mrdEuro', '#cannabis'): (('#Cannabissteuer', '#
    jugendschutz', '#legalisieren'), 1.0),
('#jugendschutz', '#legalisieren'): (('#cannabis',), 1.0),
('#cannabis', '#jugendschutz'): (('#legalisieren',), 1.0),
('#cannabis', '#legalisieren'): (('#jugendschutz',),
    0.9912280701754386),
('#Cannabissteuer', '#ca3mrdEuro', '#jugendschutz'): (('#
    cannabis', '#legalisieren'), 1.0),
('#Cannabissteuer', '#ca3mrdEuro', '#legalisieren'): (('#
    cannabis', '#jugendschutz'), 1.0),
('#Cannabissteuer', '#jugendschutz', '#legalisieren'): (('#
    cannabis',), 1.0),
('#ca3mrdEuro', '#jugendschutz', '#legalisieren'): (('#
    Cannabissteuer', '#cannabis'), 1.0),
('#Cannabissteuer', '#ca3mrdEuro', '#cannabis'): (('#
    jugendschutz', '#legalisieren'), 1.0),
('#Cannabissteuer', '#cannabis', '#jugendschutz'): (('#
    legalisieren',), 1.0),
```

```
('#ca3mrdEuro', '#cannabis', '#jugendschutz'): (('#
    Cannabissteuer', '#legalisieren'), 1.0),
('#Cannabissteuer', '#cannabis', '#legalisieren'): (('#
    jugendschutz',), 1.0),
('#ca3mrdEuro', '#cannabis', '#legalisieren'): (('#
    Cannabissteuer', '#jugendschutz'), 1.0),
('#cannabis', '#jugendschutz', '#legalisieren'): (('#
    Cannabissteuer',), 1.0),
('#Cannabissteuer', '#ca3mrdEuro', '#cannabis', '#jugendschutz')
    : (('#legalisieren',), 1.0),
('#Cannabissteuer', '#ca3mrdEuro', '#cannabis', '#legalisieren')
    : (('#jugendschutz',), 1.0),
('#Cannabissteuer', '#ca3mrdEuro', '#jugendschutz', '#
    legalisieren'): (('#cannabis',), 1.0),
('#Cannabissteuer', '#cannabis', '#jugendschutz', '#legalisieren
    '): (('#ca3mrdEuro',), 0.9734513274336283),
('#ca3mrdEuro', '#cannabis', '#jugendschutz', '#legalisieren'):
    (('#Cannabissteuer',), 1.0),
('#SaveYourInternet', '#StopACTA2'): (('#Uploadfilter',), 0.81),
('#SaveYourInternet', '#Terrorfilter'): (('#Uploadfilter',),
    0.988950276243094),
('#StopACTA2', '#Terrorfilter'): (('#SaveYourInternet', '#
    Uploadfilter'), 1.0),
('#StopACTA2', '#Uploadfilter'): (('#SaveYourInternet',),
    0.9878048780487805),
('#Terrorfilter', '#Uploadfilter'): (('#SaveYourInternet',),
    0.9728260869565217),
('#SaveYourInternet', '#StopACTA2', '#Terrorfilter'): (('#
    Uploadfilter',), 1.0),
('#SaveYourInternet', '#StopACTA2', '#Uploadfilter'): (('#
    Terrorfilter',), 0.9814814814814815),
('#SaveYourInternet', '#Terrorfilter', '#Uploadfilter'): (('#
    StopACTA2',), 0.888268156424581),
('#StopACTA2', '#Terrorfilter', '#Uploadfilter'): (('#
    SaveYourInternet',), 1.0),
('#26MaiCDU',): (('#unserEuropa',), 0.6206896551724138),
('#Innenpolitik',): (('#Weltpolitik',), 0.9962962962962963),
('#Weltpolitik',): (('#Innenpolitik',), 0.9962962962962963),
('#May',): (('#Brexit',), 0.5198675496688742),
('#Artikel11', '#Artikel12'): (('#Artikel13',),
    0.7422222222222222),
('#Artikel12', '#Artikel13'): (('#Artikel11',),
    0.8608247422680413),
```

```
('#NieMehrSPD', '#NiemalsAfD'): (('#NieMehrCDU',),
    0.7819548872180451),
('#NieMehrCSU', '#NiemalsAfD'): (('#NieMehrCDU',),
    0.9343065693430657),
('#Artikel11',): (('#Artikel13',), 0.7150837988826816),
('#GehtWählen', '#NieMehrCSU'): (('#NieMehrCDU',),
    0.9590163934426229),
('#Artikel13Demo', '#SaveYourInternet'): (('#Artikel13',),
    0.5363636363636364),
('#Artikel17', '#SaveYourInternet'): (('#Artikel13',),
    0.8689655172413793),
('#Artikel17', '#Uploadfilter'): (('#Artikel13',),
    0.812206572769953),
('#CSU', '#SPD'): (('#CDU',), 0.8731343283582089),
('#Artikel13', '#NieMehrCSU'): (('#NieMehrCDU',),
    0.838150289017341),
('#NieMehrCSU', '#NieMehrSPD'): (('#NieMehrCDU',),
    0.9379310344827586),
('#SaveYourInternet', '#Uploadfilter'): (('#Artikel13',),
    0.6268041237113402),
('#NieMehrCDU', '#Uploadfilter'): (('#Artikel13',),
    0.5906735751295337)}
```

Based on authors:

```
('#AfD', '#Artikel13', '#Europawahl2019'): (('#EU',),
    0.8071135430916553),
('#AfD', '#Brexit', '#Europawahl2019'): (('#EU',),
    0.8647260273972602),
('#AfD', '#CDU', '#EU', '#Europawahl'): (('#Europawahl2019',),
    0.8144927536231884),
('#AfD', '#CDU', '#Europawahl', '#Europawahl2019'): (('#EU',),
    0.8554033485540334),
('#AfD', '#CDU', '#Europawahl', '#SPD'): (('#EU',),
    0.8457792207792207),
('#AfD', '#CDU', '#Europawahl2019', '#SPD'): (('#EU',),
    0.8709175738724728),
('#AfD', '#CSU', '#Europa'): (('#EU',), 0.8596774193548387),
('#AfD', '#CSU', '#Europawahl2019'): (('#EU',),
    0.8288288288288288),
('#AfD', '#CSU', '#Weber'): (('#EU',), 0.8771626297577855),
('#AfD', '#Deutschland'): (('#EU',), 0.8006379585326954),
('#AfD', '#EP2019', '#Europawahl2019'): (('#EU',),
    0.8367346938775511),
('#AfD', '#EU', '#Europa', '#Europawahl'): (('#Europawahl2019
```

```
    ',), 0.8227665706051873),
('#AfD', '#Europa', '#Europawahl', '#Europawahl2019'): (('#EU
    ',), 0.8704268292682927),
('#AfD', '#Europa', '#Europawahl', '#SPD'): (('#EU',),
    0.8811188811188811),
('#AfD', '#Europa', '#Europawahl2019'): (('#EU',),
    0.8254665203073546),
('#AfD', '#Europa', '#Europawahl2019', '#SPD'): (('#EU',),
    0.8986013986013986),
('#AfD', '#Europawahl', '#Europawahl2019', '#SPD'): (('#EU',),
    0.8599397590361446),
('#AfD', '#Europawahl2019', '#Merkel'): (('#EU',),
    0.8646362098138748),
('#AfD', '#Europawahl2019', '#Sachsen'): (('#EU',),
    0.8386023294509152),
('#AfD', '#Weber'): (('#EU',), 0.8258766626360339),
('#Antifa', '#EU'): (('#AfD',), 0.8368495077355836),
('#Antifa', '#Europawahl2019'): (('#AfD',), 0.8777589134125636)
    ,
('#Article13', '#Artikel13Demo'): (('#Artikel13',),
    0.8456692913385827),
('#Article13', '#Artikel17'): (('#Artikel13',),
    0.847972972972973),
('#Article13', '#EP'): (('#Artikel13',), 0.8509803921568627),
('#Article13', '#NieMehrCDU'): (('#Artikel13',),
    0.8031591737545565),
('#Article13', '#Uploadfilter'): (('#Artikel13',),
    0.8113948919449901),
('#Artikel13', '#Artikel13Demo', '#CopyrightDirective'): (('#EP
    ',), 0.8180645161290323),
('#Artikel13', '#Artikel13Demo', '#Demosold'): (('#EP',),
    0.8832731648616126),
('#Artikel13', '#Artikel13Demo', '#uploadfilters'): (('#EP',),
    0.9365671641791045),
('#Artikel13', '#Artikel17', '#Demosold'): (('#EP',),
    0.9128856624319419),
('#Artikel13', '#Artikel17', '#EU'): (('#Uploadfilter',),
    0.8436657681940701),
('#Artikel13', '#Brüssel'): (('#Berlin',), 0.8782051282051282),
('#Artikel13', '#Brüssel', '#EP'): (('#Berlin',),
    0.9014778325123153),
('#Artikel13', '#Brüssel', '#Uploadfilter'): (('#Berlin',),
    0.8510971786833855),
```

```
('#Artikel13', '#CSU', '#SPD'): (('#CDU',), 0.802731411229135),
('#Artikel13', '#CopyrightDirective'): (('#EP',),
    0.8374384236453202),
('#Artikel13', '#CopyrightDirective', '#Demosold'): (('#EP',),
    0.9490196078431372),
('#Artikel13', '#CopyrightDirective', '#Demosold', '#
    uploadfilters'): (('#EP',), 0.9923371647509579),
('#Artikel13', '#CopyrightDirective', '#EP'): (('#uploadfilters
    ',), 0.9006920415224914),
('#Artikel13', '#CopyrightDirective', '#Uploadfilter', '#
    uploadfilters'): (('#EP',), 0.9689608636977058),
('#Artikel13', '#CopyrightDirective', '#uploadfillter'): (('#EP
    ',), 0.8573692551505546),
('#Artikel13', '#CopyrightDirective', '#uploadfilters'): (('#EP
    ',), 0.9871065604854001),
('#Artikel13', '#Demosold'): (('#EP',), 0.855475763016158),
('#Artikel13', '#Demosold', '#EP', '#uploadfilters'): (('#
    CopyrightDirective',), 0.9736842105263158),
('#Artikel13', '#Demosold', '#NieMehrCDU'): (('#EP',),
    0.8761776581426649),
('#Artikel13', '#Demosold', '#Uploadfilter'): (('#EP',),
    0.8649350649350649),
('#Artikel13', '#Demosold', '#uploadfillter'): (('#EP',),
    0.8664688427299704),
('#Artikel13', '#EP', '#Uploadfilter', '#uploadfilters'): (('#
    CopyrightDirective',), 0.9702702702702702),
('#Artikel13', '#EP', '#uploadfilters'): (('#CopyrightDirective
    ',), 0.9874810318664643),
('#Artikel13', '#Europa', '#Europawahl2019'): (('#EU',),
    0.8297546012269938),
('#Artikel13', '#NieMehrCDU', '#bots'): (('#Uploadfilter',),
    0.8295589988081049),
('#Artikel13', '#NieMehrCSU'): (('#NieMehrCDU',),
    0.847041847041847),
('#Artikel13', '#NieWiederCDU', '#WirSindKeineBots'): (('#
    gehtwaehlen',), 0.9944320712694877),
('#Artikel13', '#NieWiederCDU', '#WirSindKeineBots', '#bots'):
    (('#gehtwaehlen',), 0.9987437185929648),
('#Artikel13', '#NieWiederCDU', '#WirSindKeineBots', '#
    gehtwaehlen'): (('#bots',), 0.8902575587905935),
('#Artikel13', '#NieWiederCDU', '#bots'): (('#gehtwaehlen',),
    0.944640753828033),
('#Artikel13', '#NieWiederCDU', '#bots', '#gehtwaehlen'): (('#
```

```
    WirSindKeineBots',), 0.9912718204488778),
('#Artikel13', '#NieWiederCDU', '#gehtwaehlen'): (('#bots',),
    0.8595927116827439),
('#Artikel13', '#WirSindKeineBots'): (('#gehtwaehlen',),
    0.9205702647657841),
('#Artikel13', '#WirSindKeineBots', '#bots', '#gehtwaehlen'):
    (('#NieWiederCDU',), 0.9875776397515528),
('#Artikel13', '#WirSindKeineBots', '#gehtwaehlen'): (('#
    NieWiederCDU',), 0.9878318584070797),
('#Artikel13', '#bots', '#gehtwaehlen'): (('#NieWiederCDU',),
    0.9744835965978129),
('#Artikel13', '#gehtwaehlen'): (('#NieWiederCDU',),
    0.8662952646239555),
('#Artikel13Demo', '#Artikel17', '#Uploadfilter'): (('#
    Artikel13',), 0.8078740157480315),
('#Artikel13Demo', '#Berlin'): (('#Artikel13',),
    0.8059701492537313),
('#Artikel13Demo', '#CopyrightDirective', '#EP'): (('#Artikel13
    ',), 0.8720770288858322),
('#Artikel13Demo', '#CopyrightDirective', '#uploadfilters'):
    (('#EP',), 0.975),
('#Artikel13Demo', '#EP', '#NieMehrCDU'): (('#Artikel13',),
    0.8573825503355704),
('#Artikel13Demo', '#EP', '#Uploadfilter'): (('#Artikel13',),
    0.8197879858657244),
('#Artikel13Demo', '#EP', '#uploadfilters'): (('#
    CopyrightDirective',), 0.9663716814159292),
('#Artikel13Demo', '#NieMehrCDU', '#Uploadfilter'): (('#
    Artikel13',), 0.8346560846560847),
('#Artikel13Demo', '#SaveYourInternet', '#Uploadfilter'): (('#
    Artikel13',), 0.9048223350253807),
('#Artikel13Demo', '#Urheberrechtsreform'): (('#Artikel13',),
    0.8302945301542777),
('#Artikel13Demo', '#uploadfillter'): (('#Artikel13',),
    0.9231905465288035),
('#Artikel17', '#CDU'): (('#Artikel13',), 0.864957264957265),
('#Artikel17', '#EU', '#Uploadfilter'): (('#Artikel13',),
    0.9112081513828238),
('#Artikel17', '#NieMehrCDU', '#Uploadfilter'): (('#Artikel13
    ',), 0.8426470588235294),
('#Artikel17', '#SaveYourInternet'): (('#Artikel13',),
    0.8842443729903537),
('#Artikel17', '#Urheberrechtsreform'): (('#Artikel13',),
```

```
    0.8811410459587956),
('#Artikel17', '#copyright'): (('#Artikel13',),
    0.9170305676855895),
('#Attac',): (('#Campact',), 0.9803328290468987),
('#Berlin', '#Brüssel', '#EP'): (('#Artikel13',),
    0.8243243243243243),
('#Berlin', '#NieMehrCDU'): (('#Artikel13',),
    0.8289902280130294),
('#Brexit', '#Europa', '#Europawahl'): (('#EU',),
    0.8650927487352446),
('#Brexit', '#Europa', '#Europawahl2019'): (('#EU',),
    0.879045996592845),
('#Brexit', '#Europawahl', '#Europawahl2019'): (('#EU',),
    0.8528481012658228),
('#CDU', '#CSU', '#Europa'): (('#EU',), 0.8185975609756098),
('#CDU', '#CopyrightDirective'): (('#Artikel13',),
    0.8327814569536424),
('#CDU', '#Demosold'): (('#EP',), 0.8539898132427843),
('#CDU', '#EP', '#Uploadfilter'): (('#Artikel13',),
    0.8038277511961722),
('#CDU', '#EU', '#Europa', '#Europawahl'): (('#Europawahl2019
    ',), 0.8281733746130031),
('#CDU', '#EU', '#Europa', '#Europawahl2019'): (('#Europawahl
    ',), 0.8057228915662651),
('#CDU', '#Europa', '#Europawahl', '#Europawahl2019'): (('#EU
    ',), 0.8656957928802589),
('#CDU', '#Europa', '#Europawahl2019'): (('#EU',),
    0.8310387984981227),
('#CDU', '#Europa', '#Europawahl2019', '#SPD'): (('#EU',),
    0.8771929824561403),
('#CDU', '#Europawahl', '#Europawahl2019'): (('#EU',),
    0.8032967032967033),
('#CDU', '#Europawahl', '#Europawahl2019', '#SPD'): (('#EU',),
    0.8560371517027864),
('#CDU', '#Meuthen'): (('#AfD',), 0.873015873015873),
('#CDU', '#NieMehrCDU', '#Uploadfilter'): (('#Artikel13',),
    0.8368),
('#CDU', '#Sachsen'): (('#AfD',), 0.8748019017432647),
('#CDU', '#Weber'): (('#EU',), 0.8402457757296466),
('#CSU', '#EU', '#Uploadfilter'): (('#CDU',),
    0.8166144200626959),
('#CSU', '#Europa', '#Europawahl'): (('#EU',),
    0.8660130718954249),
```

```
('#CSU', '#Europa', '#Europawahl2019'): (('#EU',),
    0.8626543209876543),
('#CSU', '#Europa', '#SPD'): (('#EU',), 0.848780487804878),
('#CSU', '#SPD', '#Uploadfilter'): (('#CDU',),
    0.8276972624798712),
('#Campact',): (('#Attac',), 0.9908256880733946),
('#CopyrightDirective', '#Demosold', '#EP'): (('#Artikel13',),
    0.8789346246973365),
('#CopyrightDirective', '#Demosold', '#EP', '#uploadfilters'):
    (('#Artikel13',), 0.9071803852889667),
('#CopyrightDirective', '#Demosold', '#uploadfilters'): (('#EP
    ',), 0.9930434782608696),
('#CopyrightDirective', '#EP'): (('#Artikel13',),
    0.9328599096191091),
('#CopyrightDirective', '#EP', '#NieMehrCDU'): (('#Artikel13',)
    , 0.8875739644970414),
('#CopyrightDirective', '#EP', '#Uploadfilter'): (('#Artikel13
    ',), 0.8694418164616841),
('#CopyrightDirective', '#EP', '#Uploadfilter', '#uploadfilters
    '): (('#Artikel13',), 0.8820638820638821),
('#CopyrightDirective', '#EP', '#uploadfillter'): (('#Artikel13
    ',), 0.9327586206896552),
('#CopyrightDirective', '#EP', '#uploadfilters'): (('#Artikel13
    ',), 0.9517367458866545),
('#CopyrightDirective', '#NieMehrCDU', '#Uploadfilter'): (('#
    Artikel13',), 0.8639344262295082),
('#CopyrightDirective', '#Uploadfilter', '#uploadfilters'):
    (('#EP',), 0.961038961038961),
('#CopyrightDirective', '#Urheberrechtsreform'): (('#Artikel13
    ',), 0.8509984639016898),
('#CopyrightDirective', '#uploadfilters'): (('#EP',),
    0.9802867383512545),
('#DSGVO',): (('#eprivacy',), 0.8),
('#Demosold', '#EP', '#NieMehrCDU'): (('#Artikel13',),
    0.8188679245283019),
('#Demosold', '#EP', '#uploadfillter'): (('#Artikel13',),
    0.9211356466876972),
('#Demosold', '#EP', '#uploadfilters'): (('#CopyrightDirective
    ',), 0.9710884353741497),
('#Demosold', '#Urheberrechtsreform'): (('#EP',),
    0.8719723183391004),
('#EP', '#GehtWählen'): (('#Artikel13',), 0.8644338118022329),
('#EP', '#NieMehrCDU', '#Uploadfilter'): (('#Artikel13',),
```

```
    0.8264840182648402),
('#EP', '#Uploadfilter', '#bots'): (('#Artikel13',),
    0.8209677419354838),
('#EP', '#Uploadfilter', '#uploadfilters'): (('#
    CopyrightDirective',), 0.9678953626634959),
('#EP', '#Urheberechtsreform'): (('#Artikel13',),
    0.8477157360406091),
('#EP', '#copyright'): (('#Artikel13',), 0.8644628099173554),
('#EP', '#uploadfilters'): (('#CopyrightDirective',),
    0.9859408795962509),
('#EP2019', '#EU', '#Europa'): (('#Europawahl2019',),
    0.817741935483871),
('#EP2019', '#EU', '#Europawahl'): (('#Europawahl2019',),
    0.8367346938775511),
('#EP2019', '#Europa', '#Europawahl2019'): (('#EU',),
    0.8190630048465266),
('#EP2019', '#Europawahl', '#Europawahl2019'): (('#EU',),
    0.8289269051321928),
('#EU', '#EUWahl'): (('#AfD',), 0.8349633251833741),
('#EU', '#Europa', '#Europawahl', '#SPD'): (('#Europawahl2019
    ',), 0.8045977011494253),
('#EU', '#Europawahl2019', '#Merkel'): (('#AfD',),
    0.8111111111111111),
('#EU', '#Europawahl2019', '#Meuthen'): (('#AfD',),
    0.9190140845070423),
('#EU', '#Europawahl2019', '#Sachsen'): (('#AfD',),
    0.9081081081081082),
('#EU', '#Greta'): (('#AfD',), 0.8476190476190476),
('#EU', '#Meuthen'): (('#AfD',), 0.8739977090492554),
('#EU', '#Sachsen'): (('#AfD',), 0.880722891566265),
('#Europa', '#Europawahl', '#Europawahl2019', '#SPD'): (('#EU
    ',), 0.8628659476117103),
('#Europa', '#Europawahl2019', '#SPD'): (('#EU',),
    0.8236658932714617),
('#Europa', '#Merkel'): (('#EU',), 0.8118518518518518),
('#Europa', '#Meuthen'): (('#AfD',), 0.8774509803921569),
('#Europa', '#Weber'): (('#EU',), 0.859375),
('#FCKArt13',): (('#Uploadfilter',), 0.9501385041551247),
('#GehtWählen', '#Uploadfilter'): (('#Artikel13',),
    0.8041853512705530),
('#Germany',): (('#Nazi',), 0.805247225025227),
('#Germany', '#Nazi'): (('#war',), 0.993734335839599),
('#Germany', '#Nazi', '#terror'): (('#war',),
```

```
    0.9987341772151899),
('#Germany', '#Nazi', '#war'): (('#terror',),
    0.9949558638083228),
('#Germany', '#terror'): (('#Nazi', '#war'),
    0.9987341772151899),
('#Germany', '#terror', '#war'): (('#Nazi',), 1.0),
('#Germany', '#war'): (('#Nazi',), 1.0),
('#Meinungsfreiheit', '#Uploadfilter'): (('#Artikel13',),
    0.8255451713395638),
('#Meuthen', '#SPD'): (('#AfD',), 0.8643533123028391),
('#Nazi',): (('#Germany',), 0.8571428571428571),
('#Nazi', '#terror'): (('#Germany', '#war'),
    0.9987341772151899),
('#Nazi', '#terror', '#war'): (('#Germany',), 1.0),
('#Nazi', '#war'): (('#Germany',), 1.0),
('#NieMehrCDU', '#SaveYourInternet', '#Uploadfilter'): (('#
    Artikel13',), 0.8654485049833887),
('#NieMehrCDU', '#uploadfillter'): (('#Artikel13',),
    0.9237057220708447),
('#NieMehrCDU', '#uploadfilters'): (('#EP',),
    0.9473684210526315),
('#NieMehrSPD', '#Uploadfilter'): (('#Artikel13',), 0.81),
('#NieWiederCDU', '#Uploadfilter'): (('#Artikel13',),
    0.8421787709497207),
('#NieWiederCDU', '#WirSindKeineBots', '#bots'): (('#
    gehtwaehlen',), 0.998812351543943),
('#NieWiederCDU', '#WirSindKeineBots', '#bots', '#gehtwaehlen')
    : (('#Artikel13',), 0.9453032104637337),
('#NieWiederCDU', '#WirSindKeineBots', '#gehtwaehlen'): (('#
    Artikel13',), 0.9390115667718192),
('#NieWiederCDU', '#bots'): (('#Artikel13',),
    0.9370860927152318),
('#NieWiederCDU', '#bots', '#gehtwaehlen'): (('#Artikel13',),
    0.944640753828033),
('#NieWiederCDU', '#gehtwaehlen'): (('#Artikel13',),
    0.9302093718843469),
('#SaveYourInternet', '#Uploadfilter'): (('#Artikel13',),
    0.8353741496598639),
('#SaveYourInternet', '#Urheberrechtsreform'): (('#Artikel13',)
    , 0.8474320241691843),
('#Uploadfilter', '#axelsurft'): (('#Artikel13',),
    0.8778280542986425),
('#Uploadfilter', '#copyright'): (('#Artikel13',),
```

```
    0.8586251621271076),
('#Uploadfilter', '#uploadfillter'): (('#Artikel13',),
    0.9078189300411522),
('#WirSindKeineBots', '#bots', '#gehtwaehlen'): (('#
    NieWiederCDU',), 0.9859320046893317),
('#WirSindKeineBots', '#gehtwaehlen'): (('#Artikel13',),
    0.9348500517063082),
('#bota',): (('#GehtWählen',), 0.9791666666666666),
('#ep2019',): (('#GreenWave',), 0.8600891861761427),
('#eprivacy',): (('#DSGVO',), 0.8704663212435233),
('#terror',): (('#Germany', '#Nazi', '#war'),
    0.9962121212121212),
('#terror', '#war'): (('#Germany', '#Nazi'), 1.0),
('#war',): (('#Germany', '#Nazi'), 0.993734335839599)
```

A.2. Association Rules for Authors

These tables are too big to be directly appended here. They can be found in a separate spreadsheet file, *RulesTables.ods* or *RulesTables.xls*. The files contain 9 tabs, summarized in Table 3.

A.3. Inspiration Power Rankings

Table 4: Inspiration Power Rankings

original		***all***		***original with media***	
sven_giegold	23	AfD	20	faznet	7
Lars9596	20	Joerg_Meuthen	13	tagesschau	7
ZDFheute	20	ZDFheute	12	zeitonline	7
OggoleMurx	16	EchoPRN	11	ZDFheute	7
drjdvalentin	16	SilviaHerrmann_	11	zeitonline_pol	7
woelken	13	zeitonline	10	FAZ_NET	7
JfVec1	12	Gruene_Austria	9	BR24	6
KrahMax	12	tagesschau	9	phoenix_de	6
BytePirat	11	IN_ROL_0815	9	weserkurier	6
PhilHackemann	10	HansLak	8	HAZ	6
FritzFizz	10	roman51110	8	teleherzog	6
JuttaPaulusRLP	10	zeitonline_pol	8	MDRAktuell	5
Typo87	8	WKogler	8	RT_Deutsch	5
dneuerer	8	AfDKompakt	7	DLFNachrichten	4
schdrahlemann	8	BR24	7	rbbinforadio	4
jan_buehlbecker	8	faznet	7	rpo_politik	4
michabl	8	peterroger17	7	SPIEGELONLINE	4

guidoV4	7
nicolabeerfdp	7
AfD	7
watch_union	6
Gruene_Austria	6
Tagesspiegel	6
2GRIMREAPER3	5
Langhaar_Andy	5
nichtvermietbar	5
RicoTV7	5
SCHIEDER	5
faznet	5
bueti	5
Buffynator	4
spdde	4
WeissAfD	4
ComputerHeim	4
tagesschau	3
asozialpolitik	3
Blackgrey20	3
Piratenpartei	2
Jarlath_eu	2
SPOE_at	2
enavigo	2
ChWaldheim	2
Ladre15830	2
FCM_Johnny	2
Angriffsmacht	2
jjordan1964	2
wenig_worte	2
MariaNoichl	2
wolfhardt	2
SvenRecker	2
SapientemS	2
tweets_nika	2
AndreaL72801912	2
NicolasReinmann	2
MichaelRueter	2
grnstreifen	2
gruenewien	2
yes2copyright	2
ediFanoB	1

FAZ_NET	7
ndaktuell	7
teleherzog	7
its_b1nd	6
POSchenker	6
dieLinke	6
HAZ	6
rbbinforadio	6
weserkurier	6
YetiFloridsdorf	6
AfD_HD	5
PostNormalEra	5
PapayR	5
rpo_politik	5
DLFNachrichten	5
MDRAktuell	5
phoenix_de	5
RT_Deutsch	5
WDR	5
BirnstinglausRo	4
bonsai68hh	4
Scorp122	4
sven_giegold	4
ExitOILExitOEL	4
Piratenpartei	4
KarenLennartz	4
artep21	4
mrstone0856	4
SPass10541209	4
CaptainXtra	4
CDU	4
nicolangecdu	4
DLF	4
SPIEGELONLINE	4
tazgezwitscher	4
goetageblatt	4
STOPPStaatsfunk	3
ausdiesemGrund	3
generaljohnny56	3
drjdvalentin	3
FritzLiberal	3
Hessinnen	3

tazgezwitscher	4
ndaktuell	4
goetageblatt	4
SZ	3
Achgut_com	3
FAZ_Politik	3
SWRAktuell	3
DLF	3
gabonn	3
SZ_Politik	2
SZ_TopNews	2
berlinerzeitung	2
derfreitag	2
kleinezeitung	2
kn_online	2
aachenerzeitung	1
shz_de	1

Alex_S_privat	1	greenstorm2030	3
fdp	1	GrueneSt	3
Act4AnimalsEU	1	gruenewien	3
emil_goldberg	1	krippmarie	3
acti_bot	1	Achgut_com	3
Murxer	1	DeutschZuerst	3
Joerg_Meuthen	1	Volksverherzer	3
FDPamHackeschen	1	andyalt1	3
phoenix_de	1	ocelothe2k1	3
SomeXT	1	derfreitag	3
chris_pyak	1	FAZ_Politik	3
dieLinke	1	rponline	3
Disfosgen	1	SWRAktuell	3
alexandra_geese	1	jaquearnoux	3
__Alex__S__	1	ChHuegel	3
fa77648c17654f8	1	CSU	3
REWAKA2412	1	SilviaHerrmann5	3
dispotech	1	kleinezeitung	3
ISIIsidor1	1	gabonn	3
TheSile44058350	1	kn_online	3
BKVBI	1	Tinimaus1110	3
Little_Lucrezia	1	EuropeanBerlin	2
annamakoll	1	OezlemADemirel	2
SalibaJes	1	schirdewan	2
ARD_BaB	1	FixIt_EU	2
lWhiteStar	1	ManfredWeber	2
Vergwohlrumpler	1	Schnauzevoll61	2
Group4513	1	diemitdemZopf	2
CDU	1	eleration	2
WKogler	1	ZuvieleRechte	2
GrueneSt	1	mymission2030	2
katarinabarley	1	JustNobody2018	2
nudlegg	1	IrmiSalzer	2
derfieselthimo	1	katarinabarley	2
WahlkampfSPD	1	piratengetweete	2
michaelhorak	1	woelken	2
Isarjunge1	1	magna_est	2
JUAchernSasbach	1	nanniag	2
moellerdav	1	OlafBrandenbur2	2
vorwaerts	1	pkristoefel	2
d_schwarzendahl	1	TDamson	2
nicolangecdu	1	SZ_Politik	2

TorstenFischer_	1
TEnigma23	1
JenerFloh	1
TWOSIXSIXSIX	1
VaniF0X	1
KeyBe72	1
OezlemADemirel	1
DerSteuerzahler	1
Junge_Union	1

SZ_TopNews	2
SZ	2
AfDwirkt	2
Katerlady13	2
SUnterhaltung	2
AndreasAndy3131	2
Hellinvernel	2
BeaArep	2
berlinerzeitung	2
ZDF	2
wolf_kamp	2
LizzysNewsDude	2
ruslandzhebr	2
SCHIEDER	2
aachenerzeitung	1
Alpinwetter	1
AngelaStoytchev	1
anitaalhelm	1
APVogt	1
ArneLietz	1
ArnoldSchiller	1
BrauerHagen	1
BytePirat	1
danikhan326	1
martinekdahl	1
MedDivine	1
MirceaKitsune	1
EPP	1
chronocon	1
Critzebiakoffsk	1
CySpace_	1
daPeda_da	1
digitalfueralle	1
engelderliebe80	1
FDP4ever5	1
fdpbt	1
FuerOpferschutz	1
genauhinsehen	1
PFLab	1
Grafiklust	1
Hartmut73	1
Sydnai_	1

herbert_mentzer	1
heutejournal	1
heuteplus	1
IchBinRene_	1
k_edtstadler	1
MMArmbruster	1
mysteryjes	1
LandauDaniel	1
NN_Online	1
proBorderNation	1
saltedges	1
THOR200375	1
WatchPolitik	1
UdoHemmelgarn	1
ZelleTTY	1
1210gruen	1
AlternativeNRW	1
Rumsucher	1
GuidoReil	1
Schneider_AfD	1
hackenstad	1
GBAlph4	1
Raidoncosplay	1
EVP_DE	1
fr	1
shz_de	1
Leonardodawien	1
guidoV4	1
spdde	1
KpropcRoy	1
raphstar	1
SPOE_at	1
Glesi6	1

B. Source Code

B.1. dbconnect.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 13 15:30:36 2018
```

```
@author: Tobias Thesing

based on the E-Democracy connection script by Omar K. Aly
"""


from pymongo import MongoClient
import sshtunnel
import mysql.connector
import bgtunnel

#import tweepy as tp
#import json
#import subprocess
#import MySQLdb
#import datetime
#import time


'''
connects to database via SSH (requires keyfile) and returns a
MongoDB database link to the eDemocracy project
'''
def connectsource():

    print("Trying to connect to source database...")

    server = sshtunnel.SSHTunnelForwarder(
        '141.26.208.33',
        ssh_username="ubuntu",
        ssh_pkey="new_serverkey",
        remote_bind_address=('127.0.0.1', 27017))

    server.start()

    client = MongoClient('127.0.0.1', server.local_bind_port)

    db = client.twtest001

    print("Connected!")

    return db
```

```
'''
As connectsource(), but can run in background if started by
    remote access.
'''
def connectsourcebg():

    server = bgtunnel.open(
        ssh_address='141.26.208.33',
        ssh_user='ubuntu',
        identity_file='new_serverkey',
        host_address='127.0.0.1',
        host_port=27017,
        strict_host_key_checking=False)

    client = MongoClient('127.0.0.1', server.bind_port)

    db = client.twtest001

    print("Connected!")

    return db


"""
Connects to the mySQL database.
"""
def connecttarget():

    print("Trying to connect to target database...")

    tunnel = sshtunnel.SSHTunnelForwarder(

        '141.26.208.88',
        ssh_username="tobias",
        ssh_pkey="new_serverkey",
        remote_bind_address=('127.0.0.1', 3306)
        #local_bind_address=('0.0.0.0', 52875)

    )

    tunnel.start()

    print("Using port "+str(tunnel.local_bind_port))
```

```
    mydb = mysql.connector.connect(
        host="localhost",
        user="python",
        passwd="ba2014",
        database="thtweets",
        port=tunnel.local_bind_port

        )

    print("Connected!")

    return mydb


"""
Connects to local mySQL database.
"""
def connectlocaltarget():

    mydb = mysql.connector.connect(
        host="localhost",
        user="python",
        passwd="ba2014",
        database="thtweets",
        port=3306
    )

    print("Connected to local database.")

    return mydb
```

B.2. datahandler.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jan 28 02:48:07 2019

@author: Tobias Thesing
"""

import dbconnect
import time
```

```
import pandas
import re
#import pymongo


"""
Creates tables in a local mySQL database to be filled with tweet
    data.
"""
def setupTables(target):
    cursor = target.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS Tweets (id BIGINT
        PRIMARY KEY, time DATETIME(0), author VARCHAR(255),
        text TEXT(300), retweet BIGINT, reply BIGINT, quote
        BIGINT, flags INT);")
    cursor.execute("CREATE TABLE IF NOT EXISTS Topics (
        descriptor VARCHAR(255) PRIMARY KEY, type VARCHAR(255));
        ")
    cursor.execute("CREATE TABLE IF NOT EXISTS Links (tweetid
        BIGINT, topicdescriptor VARCHAR(255), FOREIGN KEY (
        tweetid) REFERENCES Tweets(id) ON DELETE CASCADE,
        FOREIGN KEY (topicdescriptor) REFERENCES Topics(
        descriptor) ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT pk PRIMARY KEY (tweetid, topicdescriptor));")
    cursor.execute("ALTER TABLE Tweets MODIFY COLUMN text text
        CHARACTER SET utf16 COLLATE utf16_general_ci NULL;")

    print("Tables created: Tweets, Topics, Links")
    return


"""
Replaces key characters in a string to insert it into a database
    without trouble.
"""
def charchop(string):

    cstring = string.replace("\\", "\\b")
    cstring = cstring.replace("\'", "\\\'")
    cstring = cstring.replace("%", "\\%")
    cstring = cstring.replace("_", "\\_")

    return cstring
```

```
"""
Extracts all Hashtags used in a text and returns them as a list.
"""
def extractHashtags(text):

    cleantext = re.sub('[^A-Za-z0-9#_äöüßÄÖÜ]+', ' ', text)
       #check
    words = cleantext.split()

    htags = []

    for word in words:

        if word[0] == '#':

            htags.append(word)

    return htags


"""
Finds out if a text has been truncated.
If so, a link to the original text is returned.
If it's truncated as a retweet, the ID of the original tweet is
   returned.
If it's not truncated, the string "COMPLETE" is returned.
"""
def analyzeTruncation(text, truncated, retweet):

    words = text.split()

    if (truncated):
        link = words[-1]
        return link

    elif (words[0] == "RT") and "..." in text:
        return "https://twitter.com/i/web/status/"+str(retweet)

    elif "..." not in text:
        return "COMPLETE"
```

```
    else:
        return "ERROR"


"""
Transfers a Tweet selected into a local mySQL database.
Tweet entry has to be in a compatible MongoDB or Twitter API
    format.

Hashtag-based topics are generated and assigned for all
    transferred tweets.

A flag number can be assigned to the saved Tweet.
"""
def transferTweet(entry, target, flags):

    t_id = entry["id"]
    t_time = time.strftime('%Y-%m-%d %H:%M:%S', time.strptime(
        entry["created_at"], '%a %b %d %H:%M:%S +0000 %Y'))
    t_author = entry["user"]["screen_name"]

    t_truncated = entry["truncated"]

    if t_truncated:
        t_text = entry["extended_tweet"]["full_text"].replace('\
            n', ' ').replace('\r', '')
    else:
        t_text = entry["text"].replace('\n', ' ').replace('\r',
            '')

    if "retweeted_status" in entry:
        t_retweetid = entry["retweeted_status"]["id"]
    else:
        t_retweetid = 0

    t_replyid = entry["in_reply_to_status_id"]
    if not isinstance(t_replyid, int):
        t_replyid = 0

    if "quoted_status_id" in entry:
        t_quoteid = entry["quoted_status_id"]
    else:
        t_quoteid = 0
```

```
    if t_retweetid == 0:
        t_hashtags = extractHashtags(entry["text"])
    else:
        t_hashtags = []

    cursor = target.cursor()

    e_query = "INSERT IGNORE INTO Tweets VALUES ("+str(t_id)+",
        '"+str(t_time)+"', '"+t_author+"', '"+charchop(t_text)+"
        ', "+str(t_retweetid)+", "+str(t_replyid)+", "+str(
        t_quoteid)+", "+str(flags)+");"
    #print("Executing Query: "+e_query)
    print("Tweet Transfer: "+str(t_id))
    cursor.execute(e_query)

    for hashtag in t_hashtags:

        e_query = "INSERT IGNORE INTO Topics VALUES ('"+charchop
            (hashtag)+"', 'hashtag');"
        #print("Executing Query: "+e_query)
        cursor.execute(e_query)

        e_query = "INSERT IGNORE INTO Links VALUES ("+str(t_id)+
            ", '"+charchop(hashtag)+"');"
        #print("Executing Query: "+e_query)
        cursor.execute(e_query)

    target.commit()

    return


"""
Transfers all Tweets from a MongoDB collection into a local
    mySQL database.
Requires a source collection (MongoDB format) and a target
    database (mySQL).

All Tweets are transferred without flags.
"""
def transferAllTweets(sourcedb, target):
```

```
    for entry in sourcedb.find():

        if all(values in entry for values in ["id", "created_at"
            , "user"]):
            transferTweet(entry, target, 0)

        else:
            print("Invalid entry skipped!")

    return


"""
Searches the mySQL database for retweet and reply references
    where the original
tweet can not be found. Locates these tweets in a MongoDB
    collection and adds
them to the mySQL database.

Tweets collected this way are flagged as 1.
"""
def addReferenceTweets(sourcedb, target):

    cursor = target.cursor()

    cursor.execute("SELECT retweet, reply FROM Tweets WHERE
        flags = 0;")
    resultset = cursor.fetchall()

    idlist = set([i[0] for i in resultset] + [j[1] for j in
        resultset])
    idlist.remove(0)

    print("List of referenced Tweets generated.")

    cursor.execute("SELECT id FROM Tweets;")
    resultset = cursor.fetchall()

    tweetlist = set([i[0] for i in resultset])

    print("List of available Tweets loaded.")

    fetchlist = idlist - tweetlist
```

```
    print("List of "+str(len(fetchlist))+" Tweets to fetch
        generated.")
    #print(fetchlist)


    z = 0

    for entry in sourcedb.find():

        if "id" in entry:

            tid = entry["id"]

            if tid in fetchlist:

                print(str(z)+" irrelevant Tweets skipped.")
                print("Referenced Tweet found: "+str(tid))
                transferTweet(entry, target, 1)
                z = 0
                fetchlist.remove(tid)

            else:

                z += 1

        else:

            print("Invalid entry skipped!")


    print(str(len(fetchlist))+" Tweets could not be found in
        source database.")
    print(fetchlist)


    return


"""
Reads a list of entries from a file.
"""
def readListFromFile(filename):
```

```
    table = pandas.read_csv(filename)

    l = table['screenname'].tolist()

    #print(l)

    return l


"""
Transfers Tweets from a MongoDB collection to the target mySQL
    database.
Tweets are selected by a list of authors given by the 'handles'
    argument.

Tweets collected this way are flagged with 2.
"""
def addTweets(sourcedb, target, handles):

    z = 0
    n = 0

    print("Searching for Tweets...")

    for entry in sourcedb.find():

        if "user" in entry:
            if "screen_name" in entry["user"]:

                author = "@"+entry["user"]["screen_name"]

                if author in handles:

                    print(str(z)+" irrelevant Tweets skipped.")
                    print("Tweet found.")
                    transferTweet(entry, target, 2)
                    z = 0
                    n += 1

                else:

                    z += 1
```

```
            else:
                print("Invalid entry skipped!")

        else:
            print("Invalid entry skipped!")

    print(str(n)+" Tweets found and transferred.")

    return


"""
Makes retweets 'inherit' the topics of the original tweets, if
    they're found
in the database.
"""
def inheritTopics(target):

    cursor = target.cursor()

    cursor.execute("SELECT id, retweet FROM Tweets WHERE retweet
        <> 0;")
    resultset = cursor.fetchall()

    for entry in resultset:

        cursor.execute("SELECT topicdescriptor FROM Links WHERE
            tweetid ="+str(entry[1])+";")
        topicset = cursor.fetchall()

        for topic in topicset:
            cursor.execute("INSERT IGNORE INTO Links VALUES ("+
                str(entry[0])+", '"+topic[0]+"');")  #problem?
            print("Added topic "+topic[0]+" to Tweet ID "+str(
                entry[0]))
            target.commit()

    return


sourcedb = dbconnect.connectsource()
pol = sourcedb["deatpols"]
```

```
full = sourcedb["german"]

targetdb = dbconnect.connectlocaltarget()


def fullTransfer():

    setupTables(targetdb)

    transferAllTweets(pol,targetdb)
    print("Transfer of original politician Tweets completed.")

    addReferenceTweets(full, targetdb)
    print("Transfer of reference Tweets completed.")

    addTweets(full, targetdb, readListFromFile("organizations.
        txt"))
    print("Transfer of media/organization Tweets completed.")

    inheritTopics(targetdb)
    print("Topics copied to retweets.")

    print("- End of Script -")

    return


fullTransfer()


# Code Sources:
# https://stackoverflow.com/questions/7703865/going-from-twitter
    -date-to-python-datetime-date
# https://github.com/jmagnusson/bgtunnel
```

B.3. analysis.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jul 19 17:34:21 2019

@author: Tobias Thesing
"""
```

```
import dbconnect
import pyfpgrowth
import pickle
import datetime
from requests.structures import CaseInsensitiveDict
import subprocess
from tabulate import tabulate
import pprint

#import pandas
#import numpy
#import datahandler


db = dbconnect.connecttarget()

"""
Searches for all topics talked about by a given author, returns
    for each topic
the number of mentions by that author.
"""
def getTopicsperAuthor(author):

    cursor = db.cursor()

    new = "AND t.time < '2019-06-03 00:00:00' "

    query = "SELECT l.topicdescriptor, COUNT(l.tweetid) c FROM
        Tweets t, Links l WHERE t.id = l.tweetid AND t.author
        LIKE '"+author+"' "+new+"GROUP BY l.topicdescriptor
        ORDER BY c DESC;"

    cursor.execute(query)
    resultset = cursor.fetchall()

    return resultset

"""
Searches for all authors talking about a given topic, returns
    author name and
timestamp for each entry. Execution can be modified with the
    mode variable.
```

```
0: standard
1: include text (01)
2: only original tweets (10)
3: include text, only original tweets (11)
4: exclude media accounts (100)
6: exclude media accounts, only original tweets (110)

"""
def getAuthorsperTopic(topic, mode):

    cursor = db.cursor()

    if (mode == 1) or (mode == 3):
        qs1 = ", t.text"
    else:
        qs1 = ""

    if (mode == 2) or (mode == 3) or (mode == 6):
        qs2 = "AND t.retweet = 0 "
    else:
        qs2 = ""

    if (mode == 4) or (mode == 6):
        qs3 = "AND t.flags <= 1 "
    else:
        qs3 = ""

    new = "AND t.time < '2019-06-03 00:00:00' "

    query = "SELECT t.author, t.time"+qs1+" FROM Tweets t, Links
        l WHERE t.id = l.tweetid AND l.topicdescriptor LIKE '"+
        topic+"' "+qs2+qs3+new+"ORDER BY t.time ASC;"

    cursor.execute(query)
    resultset = cursor.fetchall()

    return resultset


"""
Uses the getAuthorsperTopic function to provide a list of
    sequences representing
```

```
    authors discussing a topic.

    The argument delta defines how many hours have to pass without
        any registered tweet
    on the topic to consider the next tweet a seperate sequence.
    """
    def getAuthorSequencesperTopic(topic, mode, delta):

        tweetlist = getAuthorsperTopic(topic, mode)

        sequencelist = []
        last = datetime.datetime(2019, 3, 1)

        for entry in tweetlist:

            if last + datetime.timedelta(hours=delta) < entry[1]:
                sequencelist.append([entry[0]])

            else:
                sequencelist[-1].append(entry[0])

            last = entry[1]

        return sequencelist


    """
    Creates sequential transactions for all topics, split by a time
        delta (see
    getAuthorSequencesperTopic) and saves them as a list.
    """
    def createSequentialTransactions(mode, delta):

        transactions = []

        cursor = db.cursor()
        cursor.execute("SELECT descriptor FROM Topics;")
        topics = [entry[0] for entry in cursor.fetchall()]

        for topic in topics:

            newseq = getAuthorSequencesperTopic(topic, mode, delta)
            transactions = transactions + newseq
```

```
            print(topic + " added with " + str(len(newseq)) + "
                sequences.")
            print("Total: "+str(len(transactions)))
            #print(transactions)

        with open('transactionsS.pickle', 'wb') as f:
            pickle.dump(transactions, f)

        return transactions


"""
Creates a file with author sequences that is readable by the
    SPMF tool.
"""
def createSequenceDBFile(filename, sequences=[]):

        if (len(sequences) == 0):
            with open('transactionsS.pickle', 'rb') as f1:
                sequences = pickle.load(f1)

        with open('authormap1.pickle', 'rb') as f2:
            authormap = pickle.load(f2)

        csvfile = open(filename, "a+")


        for sequence in sequences:

            line = ""

            for item in sequence:

                line = line + str(authormap[item]) + ","

            csvfile.write(line[:-1]+"\r")

        return


"""
Function for processing tool results, might be obsolete.
```

```
"""
def replaceIDs(source, target):

    with open('authormap2.pickle', 'rb') as f1:
        id_to_author = pickle.load(f1)

    with open(source, "rt") as sourcefile:
        with open(target, "wt") as targetfile:

            idlist = []

            for authid in id_to_author:
                idlist.append(authid)

            idlist.sort(reverse=True)

            for line in sourcefile:
                for authid in idlist:
                    line = line.replace(str(authid),
                       id_to_author[authid])

                targetfile.write(line)

    return


"""
Calls the SPMF rule miner to use the CMRules algorithm for
    sequential rule mining.
Needs a properly formatted source file (CSV, only positive
    integers), a target file,
and minimal support and confidence in percent.
"""
def callRuleMiner(source, target, minsup, minconf, algorithm="
    CMRules"):

    lines = sum(1 for line in open(source))

    subprocess.call(["java", "-jar", "spmf.jar", "run", "
        Convert_a_sequence_database_to_SPMF_format", source, "
        tempoutput.txt", "CSV_INTEGER", str(lines)])

    subprocess.call(["java", "-jar", "spmf.jar", "run",
```

```
        algorithm, "tempoutput.txt", target, str(minsup)+"%",
        str(minconf)+"%"])

    return


"""
Reads the sequential association rules from an output file.
Returns a list of rules where each rule is a list containing
antecedent, consequent, support and confidence in that order.
"""
def parseRules(source):

    with open('authormap2.pickle', 'rb') as f1:
        id_to_author = pickle.load(f1)

    rules = []

    with open(source, "rt") as sourcefile:

        for line in sourcefile:

            fragline = line.split("␣")
            ante_id = fragline[0].split(",")
            cons_id = fragline[2].split(",")
            sup = int(fragline[4])
            conf = float(fragline[6].replace("\n", ""))

            ante = []
            cons = []

            for entry in ante_id:
                ante.append(id_to_author[int(entry)])

            for entry in cons_id:
                cons.append(id_to_author[int(entry)])


            rules.append([ante, cons, sup, conf])

    return rules
```

```
    """
    Full process.

    selector_mode: defines tweet filter (see getAuthorsperTopic())
    time_delta:    the maximum time interval tweets can have to be
       part of the same sequence
    minsup:        minimum support for calculating rules
    minconf:       minimum confidence for calculating rules
    filename:      naming of the produced files
    algo:          can change the used algorithm for SPMF
    """
    def runSequenceMining(selector_mode, time_delta, minsup, minconf
        , filename="default", algo="CMRules"):

        sequences = createSequentialTransactions(selector_mode,
            time_delta)

        dbfile = filename+"_db.csv"
        rulesfile = filename+"_rules.txt"

        createSequenceDBFile(dbfile, sequences)

        callRuleMiner(dbfile, rulesfile, minsup, minconf, algorithm=
            algo)

        rules = parseRules(rulesfile)

        return rules


    """
    Partial Process, skips time-consuming generation of sequences by
        reading from
    a prepared database file.
    """
    def continueSequenceMining(filename, minsup, minconf, algo="
        CMRules"):

        dbfile = filename+"_db.csv"
        rulesfile = filename+"_rules.txt"

        callRuleMiner(dbfile, rulesfile, minsup, minconf, algorithm=
            algo)
```

```
    rules = parseRules(rulesfile)

    return rules


"""
Uses the getAuthorsperTopic function to provide a dictionary
    with all authors
discussing a topic and for each one a list of timestamps they
    posted.
"""
def getDiscussionActivityperTopic(topic):

    tweetlist = getAuthorsperTopic(topic, 0)

    topicdict = {}

    for entry in tweetlist:

        if entry[0] in topicdict:

            topicdict[entry[0]].append(entry[1])

        else:
            topicdict[entry[0]] = [entry[1]]

    #print(topicdict)

    return topicdict


"""
Returns a list of all distinct authors that have posted Tweets
    in the database.
"""
def getListofAuthors():

    cursor = db.cursor()

    query = "SELECT DISTINCT author FROM Tweets;"

    cursor.execute(query)
```

```
    resultset = cursor.fetchall()
    authorlist = [i[0] for i in resultset]

    return authorlist


"""
Creates dictionaries that assign an integer ID to each author
   and saves them.
"""
def constructAuthorMappings():

    list = getListofAuthors()

    author_to_id = CaseInsensitiveDict()
    id_to_author = {}

    current_id = 0

    for author in list:
        current_id += 1
        author_to_id[author] = current_id
        id_to_author[current_id] = author
        #print(author + " --- " + str(current_id))

    with open('authormap1.pickle', 'wb') as f1:
        pickle.dump(author_to_id, f1)

    with open('authormap2.pickle', 'wb') as f2:
        pickle.dump(id_to_author, f2)


    return


"""
Generates a list of "Transactions" to analyze for Association
   Rules. Each Tweet
represents a Transaction and each Topic represents an Item.
   Empty Transactions
(Tweets with no assigned Topics) are skipped.

The list of transactions is also stored in a file for quick
```

```
    access.
"""
def createTopicTransactionTable():

    cursor = db.cursor()
    new = " AND time < '2019-06-03 00:00:00'"

    query = "SELECT id FROM Tweets WHERE retweet = 0"+new+";"

    cursor.execute(query)
    resultset = cursor.fetchall()

    idlist = [i[0] for i in resultset]

    transactions = []

    for tweetid in idlist:

        query = "SELECT topicdescriptor FROM Links WHERE tweetid
            = "+str(tweetid)+";";
        cursor.execute(query)
        resultset = cursor.fetchall()

        # author filter?

        if resultset:

            topics = [i[0] for i in resultset]
            transactions.append(topics)

            print(tweetid)


    print("Generated List of Transactions.")

    with open('transactions.pickle', 'wb') as f:
        pickle.dump(transactions, f)

    return transactions


"""
Generates a list of "Transactions" to analyze for Association
```

```
    Rules. Each Topic
represents a Transaction and each Author who posted about it
    represents an Item.

The list of transactions is also stored in a file for quick
    access.
"""
def createAuthorTransactionTable():

    cursor = db.cursor()

    query = "SELECT DISTINCT descriptor FROM Topics;"

    cursor.execute(query)
    resultset = cursor.fetchall()

    topiclist = set([i[0] for i in resultset])

    transactions = []

    for topic in topiclist:

        rawauthors = getAuthorsperTopic(topic, 4) #excludes
            media
        authors = set([i[0] for i in rawauthors])

        if authors:
            print("Authors for "+topic+":")
            print(authors)

            transactions.append(authors);

        else:
            print("No authors for "+topic+" registered.")


    print("Generated List of Transactions.")

    with open('transactionsA.pickle', 'wb') as f:
        pickle.dump(transactions, f)


    return transactions
```

```
"""
Loads a stored transactions file.
"""
def restoreTransactions(filename):

    with open(filename, 'rb') as f:
        transactions = pickle.load(f)

    return transactions


"""
Preliminary function to calculate association rules.
"""
def calculateAssociationRules(transactions, minsup, minconf):

    patterns = pyfpgrowth.find_frequent_patterns(transactions,
        minsup)
    rules = pyfpgrowth.generate_association_rules(patterns,
        minconf)

    print(rules)

    return rules


"""
Creates a transaction table with authors as transactions and
    topics as items.
"""
def calculateTopicRelations():

    transactions = []

    authors = getListofAuthors()

    for author in authors:

        rawtopics = getTopicsperAuthor(author)

        if rawtopics:
```

```
            topics = [i[0] for i in rawtopics]
            print(author)
            print(topics)
            print("----")
            transactions.append(topics)

    with open('transactionsT.pickle', 'wb') as f:
        pickle.dump(transactions, f)

    return transactions


"""
Internal generator function.
"""
def daterange(start_date, end_date):
    for n in range(int ((end_date - start_date).days)):
        yield start_date + datetime.timedelta(n)


"""
Counts the number of tweets in the database per day in the given
    interval.
"""
def calculateDailyActivity(start, end):

    activity = {}
    cursor = db.cursor()

    for day in daterange(start, end):

        datestr1 = day.strftime("%Y-%m-%d")
        datestr2 = (day + datetime.timedelta(days=1)).strftime("
            %Y-%m-%d")

        query = "SELECT COUNT(*) FROM Tweets WHERE time >= '"+
            datestr1+" 00:00:00' AND time < '"+datestr2+"
            00:00:00'"

        cursor.execute(query)
        result = cursor.fetchone()[0]

        print(result)
```

```
        activity[day] = result

    return activity


"""
Returns a list of all active accounts of the organizations
    handle.
"""
def getOrganizations():

    cursor = db.cursor()
    cursor.execute("SELECT DISTINCT author FROM Tweets WHERE
        flags = 2;")
    resultset = [entry[0] for entry in cursor.fetchall()]

    return resultset


"""
Experimental
"""
def filterforOrganizations(rules):

    orglist = getOrganizations()

    #with open(rulesfile+'.pickle', 'rb') as f1:
    #    rules = pickle.load(f1)

    newrules = []

    for rule in rules:

        if rule[1][0] in orglist:

            newrules.append(rule)

    return newrules


"""
Calculates popularity for each author bei looking how many
```

```
    different others
talk about the same topics. Takes a list of rules as generated
    by the
parseRules() function.
"""
def getFollowings(rules):

    poplist = {}

    for entry in rules:

        for ante in entry[0]:

            if ante in poplist:

                poplist[ante] = set(list(poplist[ante]) + entry
                    [1])

            else:

                poplist[ante] = set(entry[1])


    return poplist


"""
Runs getFollowings() for a saved file with rules and prints a
    console output
of all inspired accounts per author and their total number.
"""
def printFollowings(file):

    with open(file+'.pickle', 'rb') as f1:
        rules = pickle.load(f1)
        follow = getFollowings(rules)

        for key in follow:
            print(key)
            print(follow[key])
            print("---")

        for key in follow:
```

```
            print(key +": "+str(len(follow[key])))

    return


# Guides:
# https://medium.com/@pcm1312/implementing-fp-growth-in-python
    -170f3dc64d78
# https://www.jessicayung.com/how-to-use-pickle-to-save-and-load
    -variables-in-python/
# https://stackoverflow.com/questions/1060279/iterating-through-
    a-range-of-dates-in-python
# https://github.com/psf/requests/blob/v1.2.3/requests/
    structures.py#L37
# https://www.tablesgenerator.com/
```

C. Database Queries

For statements in this thesis that are directly based on the dataset, the following SQL queries have been used.

```
SELECT COUNT(*) FROM Tweets WHERE time < '2019-06-03 00:00:00';
SELECT COUNT(*) FROM Tweets WHERE retweet <> 0 AND time <
    '2019-06-03 00:00:00';

SELECT COUNT(*) FROM Tweets;
SELECT COUNT(*) FROM Tweets WHERE retweet <> 0;

SELECT DISTINCT t.author FROM Tweets t WHERE t.time <
    '2019-06-03 00:00:00';
SELECT DISTINCT l.topicdescriptor FROM Tweets t, Links l WHERE t
    .id = l.tweetid AND t.time < '2019-06-03 00:00:00';
SELECT DISTINCT t.id FROM Tweets t WHERE NOT EXISTS (SELECT *
    FROM Links l WHERE t.id = l.tweetid) AND t.time <
    '2019-06-03 00:00:00';
SELECT DISTINCT t.id FROM Tweets t, Links l WHERE t.id = l.
    tweetid AND t.time < '2019-06-03 00:00:00';
SELECT DISTINCT t.author FROM Tweets t WHERE t.time <
    '2019-06-03 00:00:00' AND NOT EXISTS (SELECT * FROM Links l
    WHERE t.id = l.tweetid);

SELECT t.author, COUNT(t.id) c FROM Tweets t, Links l WHERE t.id
    = l.tweetid AND l.topicdescriptor LIKE '[party hashtag]'
```

```
    AND t.retweet = 0 AND time < '2019-06-03 00:00:00' GROUP BY
    t.author ORDER BY c DESC;
SELECT t.author, t.time, t.text FROM Tweets t, Links l WHERE t.
    id = l.tweetid AND l.topicdescriptor LIKE '#FarmVille' ORDER
     BY t.time ASC;
SELECT * FROM Tweets WHERE author LIKE 'Crypto_Schurke' AND t.
    time < '2019-06-03 00:00:00';
SELECT * FROM Tweets WHERE author LIKE 'JfVec1' AND time <
    '2019-06-03 00:00:00';
SELECT * FROM Tweets WHERE author LIKE 'sven_giegold' AND time <
     '2019-06-03 00:00:00';
```

Additionally, these queries can be used to gather specific information and are applied within the Python classes:

Most used hashtags:
SELECT topicdescriptor, COUNT(tweetid)AS ntw FROM Links GROUP BY topicdescriptor ORDER BY ntw DESC;

Topics used by specified Politician:
SELECT l.topicdescriptor, COUNT(l.tweetid)AS c FROM Tweets t, Links l WHERE t.id = l.tweetid AND t.author LIKE '[author]' GROUP BY l.topicdescriptor ORDER BY c DESC;

Politicians talking about specified topic (sequence):
SELECT t.author, t.time, t.text FROM Tweets t, Links l WHERE t.id = l.tweetid AND l.topicdescriptor LIKE '[hashtag]' ORDER BY t.time ASC;

Politicians talking about specified topic (numbers):
SELECT t.author, COUNT(t.id)c FROM Tweets t, Links l WHERE t.id = l.tweetid AND l.topicdescriptor LIKE '[hashtag]' GROUP BY t.author ORDER BY c DESC;

D. Original Dataset Issues

While working on the clustering of tweets, this is what I found out about the old eDemocracy database (2017) issues.

- the "isRetweet"-flag is not used, it is "false" by default for all checked data samples
- retweets are instead marked with "RT @username:" in the text box, followed by the actual content
- this seems to be a secondary source of tweets being chopped off
- the "truncated" flag is used in some, but not all of the truncated tweets
- IF the flag is used, there is a link in the end of the text box that leads to the tweet, so it can be reconstructed if it has not been deleted meanwhile

- if the flag is NOT used, there is either no link or the link itself is incomplete and therefore unusable

It seems like the truncated flag is used for original tweets that are truncated due to the increase of the character limit. Tweets that are retweets do not have the truncated flag even if they are incomplete. As mentioned before, this might be because the in-text retweet flag uses up character space.

Other interesting findings, gathered by using my truncation analysis function on a sample:

- 59% of the tweets are flagged as "COMPLETE" and are not truncated at all
- another 29% are truncated retweets, leaving only about 12% of the tweets that are truncated originals
- however, 60% are retweets anyways

E. Iterations

The first completely transferred dataset contains 1.450.448 tweets, in which 37.717 hashtag-based topics and 741.981 links have been found. This results in an average of 20 uses for each unprocessed hashtag. This is an increase to sample runs with the old and partially truncated dataset, where a sample of 10.000 tweets only had an average of 4 uses per hashtag.

For the first run of adding references, 45.878 entries remained. In total, 22.122 tweets from the fetch list have not been found in the source collection, which corresponds to roughly half of the entries.

This results in a total count of 1.715.704 tweets to work with in the first iteration.

The second full run has cleaner processing, which was still experimental in the first one, and slightly optimized hastag parsing.

In the third run, so called "quoted" tweets are marked as a distinct mechanic, in addition to retweets and replies. Hashtags parsing is refined further, and retweets do not have their hashtags directly parsed any more due to truncation issues, but instead get them assigned from their original tweets, as long as they can be found in the database.

Numbers for all iterations:

1. 2019-03-13 through 2019-06-03, 1.450.448 tweets, 37.717 topics, 741.981 links
2. 2019-03-13 through 2019-06-03, 1.712.324 tweets, 65.674 topics, 915.385 links
3. 2019-03-13 through 2019-06-03, 1.922.163 tweets, 52.013 topics, 900.030 links - 455.964 of the tweets (number includes duplicates) up to 2019-08-22 taken from the media handle

4. 2019-03-13 through 2019-08-22, 2.297.593 tweets, 60.246 topics, 1.016.124 links