Analyzing automated activity and social deception on Twitter during the 2021 Norwegian election

Øystein Aune Sverre

Thesis submitted for the degree of
Master in Applied Computer and Information Technology - ACIT
(Cloud-based Services and Operations)
30 credits

Department of Computer Science
Faculty of Technology, Art and Design

Oslo Metropolitan University — OsloMet

Spring 2022

Analyzing automated activity and social deception on Twitter during the 2021 Norwegian election

Øystein Aune Sverre

Analyzing automated activity and social deception on Twitter during the 2021 Norwegian election

http://www.oslomet.no/

Printed: Oslo Metropolitan University — OsloMet

Abstract

The growing concern of fake news and social bots as threats to democracy leaves society with motivation to investigate and research its presence. In this thesis, we look into social bot research and try to understand the current landscape and its caveats, including its reliance on closed-source tools such as Botometer for bot detection. A Twitter data set is created, consisting of political Tweets made during the 2021 Norwegian election. A variety of techniques, such as manual inspection, plagiarism, exploratory data analysis, and Botometer scores are used to investigate the presence of automated activity and social bots. In the course of this thesis we find no concrete evidence of disguised automated activity or social bot presence, and discover multiple inconsistencies in the Botometer classification results. An argument is made for research to rely less on closed-source tools and resort to more reliable ways to investigate and understand social bots.

Acknowledgments

I would like to thank my girlfriend and my supervisors, Hårek Haugerud and Anis Yazidi, for dealing with me during this ordeal.

Contents

Chapter 1

Introduction

1.1 Motivation

As society adopts new technology, we continuously alter the way we function. This comes with new mediums for consuming entertainment, information, communication, and so on. Along with these new technologies, there is also an easier, more open point of entry, resulting in a big expansion of content available. The Internet for instance has more websites, data, and distractions than one could possibly consume. This leads to sub-cultures and communities that inhabit different parts of the Internet, sharing thoughts and ideas with like-minded people, sort of like ethnic enclaves within big cities consisting of ethnic concentration and characteristic cultural identity.

The rising use and growth of the Internet have made this medium the go-to with functions such as video, music, fiction-writing, audiobooks, shopping, news, and so on. This centralization and ease of entry have made the lines between content categorization blurry. News used to be made exclusively by scrutinized entities and published on reputable channels, but now it can be hard to differentiate. A tweet on Twitter can be an anecdote, a joke, a story, a conversation, news, fiction, or even a paid advertisement. Frequent Internet users are continuously exposed to this stir and are learning to scrutinize and categorize this themselves, but are at fault for human error.

This has in turn led to the rise of the term "fake news". As advertisements can give a profit based on Internet traffic and attention, and the line between news and opinion is blurry, certain institutions thrive on generating web traffic by misleading or lying to their audience. Some also say, considering how susceptible the human mind is to engineering and hive-mind mentality, it also leaves us at risk of being manipulated based on our own Internet enclaves. One could be duped and have their opinions altered and shaped according to someone else's desire, giving fake news a whole different use case than just generating web traffic.

If someone were to automate this behavior on social media to influence our opinions this could be used to alter public perception, much like war propaganda. Ever since discussion and speculation were brought up

regarding the legitimacy of the 2016 United States presidential election, there has been a rising concern about whether this also poses a risk to the modern democracy, as motivated groups could stage a modern version of a coup d'état without anyone ever knowing.

1.2 Problem Statement

To gain knowledge on the subject of fake news and digital deception, and as an investigation into the severity of the problem, an analysis will be completed, exploring social media activity related to the 2021 Norwegian election. The issues will boil down to how one discovers automation, social bots, digital deception, and manipulation. Bots who are attempting to blend in could potentially be tracked by examining behavior, statistics, and trends, effectively separating them from other, regular users. If one could find these patterns, this could be utilized as a tool or technique for bot de-obfuscation, decreasing their power.

By looking at data available on Twitter we will use exploratory data analysis, scripting, Botometer, and manual inspection to try to find any indications or clues of foul play.

Are there suggestions of automated activity or social bots on Twitter during the 2021 Norwegian election?

Chapter 2

Background

2.1 Fake news

Fake news as a topic has always been relevant, as the act of spreading false information, deliberate or unintentional, can have huge consequences on individuals or society as a whole. A clear example of this is war propaganda. For example, a significantly higher share of anti-Semites can be found among women born in the 1920s who were subjected to Nazi propaganda [13]. As humans, we rely heavily on the information we perceive, both conscious and subconsciously [20], and make informed decisions based on our goals and aspirations, not just human instinct. The topic has however seen a major bump in discussion and research since 2016, most notably because of the events and fallout of the 2016 United States presidential election. As can be seen in the figure below, the search interest on the world's largest search engine, Google [32], was basically dormant prior to 2016.

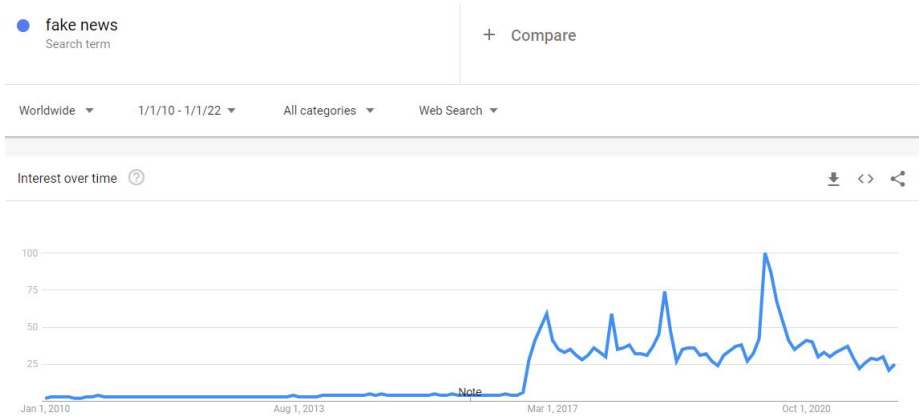

Figure 2.1: Google's service, Google Trends, shows a graph of worldwide search activity on the term "fake news" between 2010, and 2022. The line is flat, until a large uptick in late 2016, which continues with annual upticks and larger withheld interest in between.
[11]

Fake news was already imminent as a threat online however, the phrase "Don't believe everything you read on the Internet" is known as a term regularly given to kids by parents. This phrase was mentioned in articles such as "Technology: Truth or Consequences: Don't believe everything you read on the Internet" as far back as the year 2000 [26]. The author mentioned in this article how a contact had forwarded an email that claimed, with sources, that Tommy Hilfiger had made racist remarks on the Oprah Winfrey show. After further investigation, the author discovered this was false, as Tommy Hilfiger had never even appeared on the show. The author stated: "Hoaxes on the Internet have become a fact of life in the e-mail age." There seems to be a healthy skepticism growing as the ease of spreading information grows.

In the example mentioned above the author received false information by e-mail, which was a much more common way of communicating before the rise of social media in the 2000s. In the figure below you can see the immense growth of social media platforms online after 2002. These platforms have in turn made e-mail more of a work-related tool [4], while social media, with certain exceptions like LinkedIn, is more used for interactions and sharing information between individuals. Social media have quickly become a common part of life, growing so fast that MySpace even competed with Google as the world's largest website in 2006 [7].

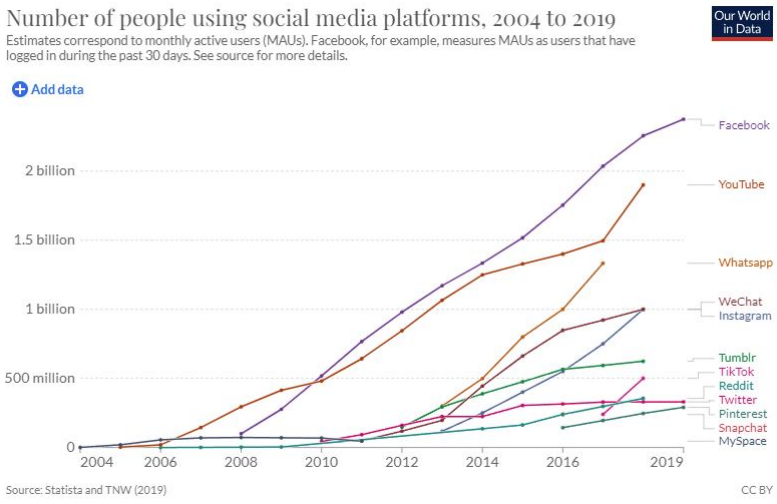

Figure 2.2: A graph published by Statista and The Next Web, using statistics and estimates, shows how social media platforms have grown drastically over the past 20 years.

[23]

Not everything on social media is personal after all, as it allows for following institutions, corporations and personas online, with easy sharing and discussion within your chosen social circle. These, possibly large, pages and accounts could provide the illusion of ethos, as there are fewer personal takes by an author and more of a trustworthy publication without a face, such as a journalistic institution. Being on social media these

publications feel closer and more personal compared to more distanced mediums such as television and newspapers. A scientific article written in 2019 performed an analysis that strongly concluded that people will be more likely to trust a story, and also engage with it if it is shared by someone they trust [33]. This does mean reputable sources have an edge when sharing information on social media, as they have built a foundation of trust for their work, but this is also affected by social engagement. The article mentions how the source of potentially fake news is often overlooked if the person spreading/sharing the news on their own page is trustworthy to the reader. This could result in non-news/journalism-related personas and institutions becoming a source of news and information, without ever needing a journalistic background.

While a page sharing possibly false information must have been manually forwarded to you in the past via e-mail, indirectly vouched for by your contact, this can now appear in a social media feed automatically. The social platform might believe you are interested just because your social circle is engaging with the content, not taking into account whether this information is valid or with ill intent [16]. This means that if someone in your social circle has placed their trust in a less reputable source, this automatically spills into your timeline, potentially spreading even further. This can become more prominent as you have the ability to manage a bigger social circle on social platforms, with potentially less trustworthy contacts or people you don't even know.

Fake news is commonly defined as "Fictitious articles deliberately fabricated to deceive readers" [3]. This does not declare its intent however. The ease of spreading fake news did not only increase with the Internet but also its profitability. A journalistic publication printing a newspaper typically wants to attract attention by crafting deliberately interesting headlines. There is also the concern of returning customers however, so the line between legitimate and fake news headlines had to be carefully monitored, leaving customers with a feeling of wanting to repurchase the newspaper. Be that as it may, online you can generate profit through advertisements, fueled by click rates and page views. This coupled with visitors who are open to exploring websites, as news publications are typically free, means that one can generate money just by disappointing visitors and then finding new visitors.

2.2 Digital deception and manipulation

After the uptick in fake news discussion in the late 2010s, the worry has not only been misinformation but what the intent of that deception is. Typically, online fake news was driven by making a big profit with small effort, as you are not required to do the grunt work often coupled with extensive journalism. Now there is a growing concern that misinformation and fake news, presented as valid information, can be used to deceive and manipulate people, not unlike the war propaganda mentioned earlier. The act of misleading the audience could still have monitory motivations,

as one could exploit someone for profit, but the concern is also other motivations, such as generating dissent within a country's politics. This deception is argued as a threat to modern democracy [17] as individuals could have issues differentiating information and knowing whom to trust.

In "Belief in conspiracy theories and attitudes toward political violence" Federico Vegetti and Levente Littvay stated: "In the last decade, violent political events, often but not always triggered in the context of protests, has been rising in Western democracies. At the same time, there has been a steady increase in the diffusion of conspiracy theories in political communication, a phenomenon that has captured the interest of scholars for its growing political relevance." [39]. One could argue with the increasing awareness of the power of social media, the people are becoming frustrated and conflicted, fearing how different people are trying to deceive them. A study from 2014 said that 55% of respondents in 2011 agreed with at least one of the conspiracy theories presented to them [22]. This research was published 11 years ago and the presence of conspiracy theories as a topic during the 2016 United States presidential election and 2019 COVID pandemic could allude to a growing trend.

A study published in 2017 [31] found quantitative empirical evidence that social bots play a key role in the spread of fake news during the 2016 United States presidential election. Social bots are bots abusing automation tools to generate large amounts of social media activity in order to support, or oppositely attack, political figures, in line with their own interests. Much of the misinformation they tracked was published by a set of relatively few accounts. When comparing one random set of users who shared regular news, and one random set who shared fake news, the bot activity was noticeably higher in the set sharing fake news. These users were given a bot score by the tool Botometer, which analyzes Twitter accounts and provides bot scores based on a series of tests.

The study concludes by stating social bots are an effective method for manipulating social media users and are most likely present on every social media. They propose increasing the use of bot challenges on users, such as CAPTCHAs, could make it much more challenging for bots to automate their behavior. This study is one example of the increased publishing of social bot-related studies which started after the 2016 election, which in turn has led to increased social awareness through news articles related to the subject. You can argue that the concern of social bots online is relatively widespread in 2022.

Chapter 3

Related work

3.1 A similar study

"Spotting political social bots in Twitter: A use case of the 2019 Spanish general election" [24] was published in 2020 and goes in-depth in both quantitative and qualitative examinations related to the 2019 Spanish general election. The presence and behavior of social bots on Twitter were analyzed and their results discovered a non-negligible amount of bots who actively participated in the election, effectively blending into the political landscape.

The authors start their research by defining multiple terminologies, for instance, social bots. This is also later referred to as "astroturfing", which fakes the appearance of real participation while being secretly orchestrated. This activity is purposefully made and fabricated, but its effect on bystanders and participants in echo chambers could be used to skew public perception.

For analysis, the project harvested all Tweets which used one of multiple predetermined hashtags. The harvesting was done for roughly one month, collecting around six million tweets from about one million different users. Then the Tweets were used for analysis and knowledge extraction. One example is their attempt to extract the tone used in a Tweet, ranging from zero (extremely negative) to one (extremely positive). Afterward, the Tweets were cleaned for easier processing, removing special characters and turning emojis into text.

To perform bot classification for the Tweets, the project utilized Botometer [5]. This is a machine learning platform that extracts and analyzes over 1200 features, resulting in a score from 0 (likely human) to 1 (likely a bot). Botometer is also used in four of the seven related works mentioned in the paper. Botometer is described as the "de-facto standard" of bot identification on Twitter and with a high performance, measured using both data sets. The paper does not say which version they are using but insinuates they are using the latest, version 3.

Further, the project created a political affiliation classifier, hoping to be able to sort the bots according to their political behavior. Natural language processing was used to provide sentiment scores for each group of political

party's keywords, essentially allowing the researchers to classify what political party the social bot is rooting for.

18.50% of the users investigated were classified as "uncertain" when obtaining social bot status, which led to them being discarded. 5.12% of the users were classified as social bots, which is 40 098 accounts.

When the data set had classifications for bots and political affliction, multiple investigations were done on different topics. Such as differences between normal and bot behavior, a social bot relation node-network (with political affliction), bot engagement activity, and so on. The paper concludes by making a set of insights and conclusions based on their research, and discussing their findings.

It should be noted that even though the research in this project is extensive and an interesting insight into Twitter data, all the conclusions and results are based on the fact that Botometer gives accurate and meaningful results. The authors also acknowledge this in their discussion, but excuse it by saying Botometer is state of art and widely considered the best option. If Botometer were to be labeled as unreliable it would render all conclusions which are based on differences between the user-sets inconclusive.

A majority of research into social bots seems to be based on Botometer, and this is concerning as we have limited knowledge of the way Botometer functions. If the research community utilized a bigger set of tools this would at the very least give some validity to the growing concerns, but at the moment Botometer seems to be the main tool. The content which follows in this thesis is not a test or comparison experiment for Botometer specifically, but utilizing their tool and acknowledging their presence and place in the research community is important, as they are a big influence on all research in this field.

3.2 The Counterargument

"The Rise and Fall of 'Social Bot' Research" [10] published in 2021 by Florian Gallwitz and Michael Kreil offers a counterargument to the growing concerns raised by social bot research. The authors stated that researchers use crude and questionable heuristics to discriminate between real or automated accounts, or in the majority of cases, rely entirely on the output of automatic bot detection tools. This commonly being Botometer.

The first part of their research establishes multiple other big research projects and critiques their approach by pointing out flaws and misunderstandings when labeling users as bots. One example is a heavily cited study where accounts tweeting 50 times per day were labeled as using "heavy automation", which Gallwitz and Kreil argue is entirely possible for power users while still achieving a healthy amount of sleep. They showcase several examples of human celebrity accounts that exceeded this threshold. The authors also state that "Highly active Twitter users with strong political opinions are commonly dismissed as 'bots or trolls' by users with opposing political views".

Gallwitz and Kreil manually reviewed and contacted several accounts showcased as social bots, and discovered with a high degree of certainty that most bots were just very active passionate political accounts. They continue by saying Botometer generates an enormous amount of false positives, and can therefore not be relied on as a scientific tool. One approach they used to test the false-positive scores was to point Botometer towards users they knew without any doubt were human. Below you can see their results, utilizing a Botometer threshold of 50%:

- 47% of U.S. Congress members present on Twitter were misclassified as bots.
- 10.5% of NASA-related accounts are misclassified as bots.
- 12% of Nobel Prize Laureates are misclassified as bots.
- 14% of female directors are misclassified as bots.
- 17.7% of Reuters journalists are misclassified as bots.
- 21.9% of staff members of UN Women are misclassified as bots.
- 35.9% of the staff of german news agency "DPA" are misclassified as bots.

In their conclusion, Gallwitz and Kreil explain that social bot research is flawed, and didn't manage to find one credible example of a social bot in their investigation. They elaborate by saying future research should have open data sets of accounts, at the very least allowing other researchers to validate the findings. Their counterarguments do have valid criticism of the research community, and if their research is true this could mean the hysteria concerning social bots is significantly overemphasized. This would also mean a lot of research into the subject is non-credible.

"The False positive problem of automatic bot detection in social science research" [25] published in 2020 also investigate this issue. They examined Botometers scores on validated data sets of humans and bots and achieved similar results as Gallwitz and Kreil. A comparison was also done, looking at how Botometer functions when exposed to English and non-English speaking accounts. The results showed that Botometer functioned significantly better at English accounts, even though it still had false positives in both instances.

Both the Spanish social bot study [24] and the one being conducted in this thesis, have data sets with non-English speaking accounts. Based on this, this will drastically increase the false-positive rate when using Botometer.

3.3 Potential ethical issues

Even if the initial research which sparked the debate in section 2.2 was countered in section 3.2, that does not necessarily mean the issues

presented there are non-existent. There are big indicators that foreign agents tried to meddle with US elections using social media advertisements [15]. The issue is how big their presence was and how big of an impact it made. The critique is valid, however, it doesn't completely shove away all their findings.

There are multiple reasons the researchers might not be able to share their complete findings after completing a study. When dealing with external data from social media one has to comply with their uses and more often than not, apply for API access [38]. This means you have elevated access to internal systems instead of scraping data externally from their web page, which is a legal grey area [29]. When applying for this access you sometimes have to define how the data will be used, and if it will be shared. The research which was criticized might not have been able to do any research at all if they had no data, meaning you could argue some restricted findings are better than none. At least if you trust the source.

The issue of using closed source algorithms and tools to find and define social bots is more concerning, but could perhaps be the result of the same issue. It could be the best tools for the job are closed source for a reason, trying to generate some sort of revenue, such as Botometer [5]. Being open-source provides more academic integrity, allowing researchers to scrutinize your code [12], but comes with the addition of being harder to monetize, making developing it harder.

Though these issues are understandable from a data and tools use perspective, this does not mean the research totally solid. In science, one must be able to validate and recreate the approach, results, and conclusions. Therefore the research in section 3.2 has valid concerns, especially concerning the attention the subject received. The topic requires more research with clearer definitions, open data sets, and open-source tools.

Chapter 4

Approach

This section will explain the process of obtaining the data set, and various scripts and techniques which utilizes it. The results will be covered in the "Results" chapter.

4.1 Obtaining a data set

To gain knowledge about the political presence on Twitter related to the Norwegian election, a data set has to be generated with relevant Tweets. Afterward, this data can be utilized to perform analysis, and possibly obtain further data related to the Tweets, such as the account details of the Twitter authors.

The Tweets wanted for this data set should revolve around political discussions, as the point of this investigation is to discover possible foul play trying to skew public perception. The easiest way to obtain relevant Tweets is then to use Twitter's search function, specifying certain keywords and details the Tweets must fulfill to appear in the search results. This is very much like the approach used in the Spanish study [24] mentioned in the "Related work" chapter but involves more than just relevant hashtags.

The data set needs Tweets that were published between 01.01.2021 and 15.09.2021. The analysis requires a specific time frame to look at, and one can assume the most political discourse is during an election year. A multiple-year data set with analysis for political engagement and other relevant trends would be interesting, but such a scope is unrealistic for a project of this size. Filtering the Tweets by date is fairly easy, as this is common metadata and is also integrated into Twitter's advanced search function.

The keywords to select the Tweets from need to be chosen carefully, creating a data set that is as relevant as possible. The selected keywords are from certain categories, fulfilling several purposes. When choosing these keywords, assumptions were made. For instance, if you are interacting with a Twitter account for a political party, your conversation is political in nature. Therefore, every Tweet that tags some political party account, and mentions them by name or acronym, gets collected into the data set. This does mean the occasional Tweet is caught in the crossfire that is not

relevant, but the keywords are carefully selected to minimize this. For instance, the party "Arbeiderpartiet" is an easy keyword to collect, but the acronym "AP" does not work, as it means different things around the world. Therefore that acronym has to be avoided when collecting Tweets. This later must be considered when observing the results as well, as some results could be tainted by imprecise search keywords.

Boolean logical operators can be utilized to do some smart filtering, as the Twitter search function supports this use case. An example is whether or not people mention politicians by their full name or mitigate some part of it, accidentally avoiding being scraped. A logical operator such as this "(Kjell Ingolf Ropstad OR Kjell Ropstad)" ensures mentioning this politician, with or without the middle name, gets scraped.

Even smarter filtering can be done, using the language metadata on Tweets. This is a language classification done on Tweets by Twitter themselves and the result is available to be viewed in the Tweet metadata. This is fairly unreliable unfortunately and not used as a factor in this approach. The similarities to other Scandinavian languages and the confusion which arises with short Tweets and English slang results in this metadata not being useful for picking which Tweets are relevant. The same applies to the geographical location, which is only on Tweets where the author explicitly allowed it, resulting in a significantly smaller data set.

Even with all these considerations, the data set does include occasional Tweets not particularly meant for this data set, such as Swedes discussing EU membership for Sweden, but overall the keywords do a thorough job of creating a relevant data set. Because of the hashtags chosen, such as "#NorwayElection", there are some foreign discussions included, but there is no reason to think social deception or fake news could not affect the Norwegian population through English discussions, so this is not viewed as a problem.

Below you can see a representation of the CSV file containing search strings and the category to which they belong. These search strings will be used in Twitter's search function to pull the relevant Tweets for the data set.

4.1.1 Keywords

	Search	Category
1	Arbeiderpartiet	Party
2	(@Hoyre OR #Høyre)	Party
3	Senterpartiet	Party
4	(Fremskrittspartiet OR frp_no)	Party
5	(Sosialistisk Venstreparti OR Svparti)	Party
6	(@Raudt OR #Rødt)	Party
7	(@Venstre OR #Venstre)	Party
8	(Miljøpartiet De Grønne OR #MDG)	Party
9	((Kristelig Folkeparti OR #KRF) OR KrFNorge)	Party
10	Demokratene i Norge	Party
11	Pensjonistpartiet	Party
12	Partiet De Kristne	Party
13	(Industri og Næringspartiet OR Næringspartiet)	Party
14	Partiet Sentrum	Party
15	Helsepartiet	Party
16	Pasientfokus	Party
17	Liberalistene	Party
18	(Folkeaksjonen OR Nei til mer bompenger)	Party
19	Piratpartiet	Party
20	Norges Kommunistiske Parti	Party
21	Feministisk Initiativ	Party
22	Kystpartiet	Party
23	Generasjonspartiet	Party
24	Redd Naturen	Party
25	Jonas Gahr Støre	Party leader
26	Erna Solberg	Party leader
27	(Trygve Slagsvold Vedum OR Trygve Vedum)	Party leader
28	Sylvi Listhaug	Party leader
29	Bjørnar Moxnes	Party leader
30	Guri Melby	Party leader
31	(Une Aina Bastholm OR Une Bastholm)	Party leader
32	(Kjell Ingolf Ropstad OR Kjell Ropstad)	Party leader
33	Irene Ojala	Party leader
34	(politikk OR politisk)	Term
35	forsvarspolitikk	Term
36	innvandring	Term
37	(abortdebatt OR abortpolitikk)	Term
38	(klimapolitikk OR klimadebatt)	Term
39	skolepolitikk	Term
40	velferd	Term
41	distriktspolitikk	Term
42	(EU-medlemskap OR (EU AND medlem))	Term
43	((politisk OR politikk) AND debatt)	Term
44	rusreform	Term
45	#NorwayElection	Hashtag
46	#valg2021	Hashtag
47	#stortingsvalget2021	Hashtag
48	#stemnytt	Hashtag
49	#stortingsvalget	Hashtag

4.2 Twitter API

The initial plan was to utilize Twitter's own API (Application Programming Interface) through their Developer Platform. This would allow the use of the Twitter search function through programming, automating the process and giving fast standardized results. The process of obtaining access is fairly streamlined, but still created a bit of confusion about how to move forward. After applying you are granted a level of access related to your purpose, which affects what endpoints you can access, rate limits, and so on. Twitter responded with some requests for further elaboration regarding this project, before denying our application. After re-applying, it was suddenly accepted.

The whole ordeal took some time, and even when granted access to the API it was revealed that the access given did not include the Twitter Archive, which was necessary to obtain Tweets from 2021. Without it, the search function would only display recent Tweets. It became fairly obvious that using the Twitter API was not an option if this project should be completed on time with relevant data.

4.3 Twitter scraping

As several people have issues, both using and obtaining access to the Twitter API, there are alternatives. The Tweets are publicly available on Twitter, there is just no automatic way of obtaining them in a readable format without the API. By using an open-source project name "Twint" one can scrape Tweets from Twitter's website and save them, automating an otherwise impossible task. The project allows standardized saving to several formats as if we were using the API, the only difference being we are calling the Twitter front-end, and saving the Tweets from there.

The project itself is a bit outdated and has, because of updates to Twitter's website, become relatively unstable. Luckily other GitHub users have forked the project and created stable builds that are currently functioning. After some experimentation, it was decided that a certain Twint fork[19] was sufficient for this data collection.

A simple Python script was then programmed (A.3), utilizing the Twint Python library for scraping Tweets. The script reads the CSV file containing all the keywords and then conducts a search for each keyword string, finding all Tweets that match within the programmed time frame. This search takes some time because Twitter's front-end is a less efficient way of requesting data, but the scraped Tweets for each keyword are then formatted into a JSON data format, and then stored in their own file. Ultimately we are left with 49 JSON files containing Tweets, which are roughly 302 MB in size.

Another Python script is then used to combine the results from each search (A.2), removing any possible duplicates that may have been collected. This results in a final JSON document, with all collected Tweets and related metadata.

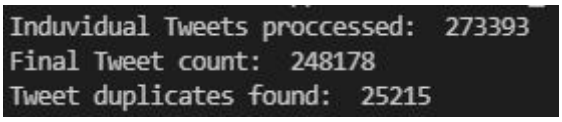

Figure 4.1: The console output from the previous script showcases how the files were combined, removing 25 215 duplicate Tweets.

The final data set is a 277 MB JSON file containing 248 178 Tweets from a nine-month period.

4.4 Python scripting

Several Python scripts were created for processing this data set, fulfilling unique tasks. Here is an example.

Using a Python script (A.4), all the text from the Tweets in the data set was combined into one string, and a word cloud of the words used was created. Each time a word occurred it grew in size. Some words were filtered (adverbs, pronouns, prepositions...) as they are frequently used in all language and obscured the main goal, which was to observe themes, tone, and names. The lists of words to remove were gathered from three sources, a public GitHub repository [1], and two Norwegian language learning platforms [6] [8].

The main goal of the Python scripts is to process big amounts of data and do exploratory data analysis using the Tweets in the data set. Therefore many scripts exploring many different statistics were created. The code can be found in the appendix and on GitHub [35]. Here is a list of scripts used to process data, and a short description:

- Word cloud of words tweeted (A.4)
  Python script aggregating every word used all Tweets, creating a word cloud. Every unique word grows based on each occurrence. Certain words are filtered out based on an external directory. Finally, the word cloud is displayed in the shape and color of a Norwegian flag.

- Most active users by Tweet count in the data set (A.7)
  Python script counting the number of Tweets tweeted by each user, sorting their results, and displaying the top 25 users on a bar chart in descending order.

- The most engaging tweeters in the data set by likes and retweets (A.6)
  Python script calculating the engagement users receive, based on likes and retweet averages per Tweet. The list is sorted based on engagement score and the top 25 users are shown on a chart in descending order.

- The most shared domains from Tweets in the data set (A.10)
  Python script counting the most frequently shared domains in all the

different Tweets. Anything after the URL domain is ignored, and sub-domains are removed. The list is then sorted and the top 25 domains are shown on a chart in descending order.

- Tweet sharing activity by hour in the Norwegian time zone (GMT+1) (A.13)
  Python script converting every Tweet to the GMT+1 time zone, aggregating a count of Tweets per hour in a 24-hour day, and finally displaying a bar chart where each bar represents an hour in a day. The bar grows with each Tweet made within that hour. Essentially showcasing a timeline of the average Twitter activity throughout a day.

- Botometer score averages for Tweet authors in data set (A.9)
  Python script which reads all Botometer replies. If a user received a score above 0.8 (out of 1.0) in any category it is printed to the console for manual investigation. Every Botometer category has an average which is calculated based on all the Botometer replies and displayed at the finish.

- Plagiarism checker of Tweets in data set (A.11)
  Python script which selects Tweets within different categories and compares every Tweet within those categories against each other using a Python library. If the Tweets share a plagiarism ratio above 80 (out of 100) it is saved for manual inspection.

4.5 Botometer

Botometer has faced much criticism for its tools [10], as mentioned in the "Related work" chapter, so a data comparison was desired for perspective. Their API access has a free tier, however, the rate limits slow the usage down significantly. Therefore the premium tier was utilized for this. This thesis utilized the newest version available, version 4.

Every unique Tweet author was filtered out of the data set and saved separately. Then that new data set could be used to send an individual call to the Botometer API for each author. The reply from the Botometer API has several values based on the user's desired result. The first category is "English" and "Universal". The results in "English" utilize all features within Botometer and "Universal" utilizes only language-independent features, which is better for non-English data sets. This is possibly related to the findings in the 2020 study [25] which stated that Botometer is more effective on English-speaking accounts. Because this data set is in Norwegian, "Universal" was utilized.

From there you can select a range category, "Raw", which displays results ranging from 0 through 5, and "Display" which is ranging from 0 through 5. Raw was chosen because it is more in line with programming principles. From there the final sub-categories are defined as:

- fake_follower: bots purchased to increase follower counts.

- self_declared: bots known from botwiki.org.
- astroturf: manually labeled political bots and accounts involved in follow trains that systematically delete content.
- spammer: accounts labeled as spambots from several data sets.
- financial: bots that post using cashtags (Twitter stock tags).
- other: miscellaneous other bots obtained from manual annotation, user feedback, etc.

Good documentation explaining in detail what each category does is scarce, and since the accuracy is debatable, the focus will be on the averages and looking for standouts from the norm. This is with the mindset that a majority of the data set is real people. After a simple Python for-loop pulled data from Botometer, scores were harvested for 37 674 users and failed for 242 users. Based on an investigation into the user IDs of these users, this is because these accounts are "Protected", requiring an accepted follow of the account to read the user data. The scripts saved the results for each user in a separate data set.

After every available user's Botometer scores were retrieved, another Python script was created (A.9) to check for averages. All the different averages were calculated for each Botometer sub-category.

4.6 Plagiarism

There are several techniques to detect bots, but a fair assumption with bots is a level of repetitiveness. To have fluctuations in timing, language, or to avoid patterns in general, this has to be programmed in. Effective bots utilize machine learning and algorithms to emulate human behavior or some degrees of randomness. All of those techniques are hard to perfect on language construction though, as random or generated sentences can be weird or meaningless without a proper message. Therefore looking for plagiarism or repetitiveness within the Tweet's content is a simple way to scout for bots, though social bots are probably more effective at hiding.

By utilizing a Python library named "thefuzz" [30], which is created for string matching, different Tweets can be compared and generate a matching ratio from 0 through 100. These scores can be used to classify and discover accounts that are repeatably spreading certain messages or accounts repeating the same message.

A Python script (A.11) loaded every Tweet from the data set and utilized another library named "itertools" [9]. This library has a function named "combinations" which creates every unique combination of Tweets from the data set, which we then can compare using "thefuzz". The expression for the combinations, where n is our Tweet count and r is the number of Tweets being compared each time, is:

$$\frac{n!}{r!(n-r)!}$$

With our data set of 248 178 Tweets and comparing pairs, r=2, this number is:

$$30\,796\,035\,753$$

This is an extreme amount of combinations, but an attempt was made to compare all Tweets in the data set. After the script had been running for a couple of days it was calculated that with the current progression, it would take an additional 19 days to complete the entire data set. The project didn't have that time available so another approach had to be used instead.

The script was altered to check certain categories of Tweets, using keywords, and do plagiarism checks on Tweets within those categories. The keywords function much the same way as they did in the scraper. If the keyword is mentioned in the Tweet string, it applies. As plagiarism is being investigated within a subset of the data set, instead of the complete version, the keywords are chosen based on their controversy and debate. If someone launched a social bot campaign you could argue they would target a certain discussion or audience, as this discussion would be the place to blend in and cause dissent and chaos. Either way, the scope must be limited for this project to be feasible.

The keywords chosen are:

- innvandring (immigration)
- abort (abortion)
- klima (climate)
- skole (school)
- velferd (welfare)
- rusreform (drug reform)

Each Tweet comparison was initially set to save every combination matching that resulted in a ratio above 65, but this was too sensitive and led to a large amount of similarly built sentences and phrases. This was then changed to 80, which appears to be a better threshold, without requiring 100% identical Tweets. In the figure below you can see the Python script looping through each keyword, comparing every Tweet within that category:

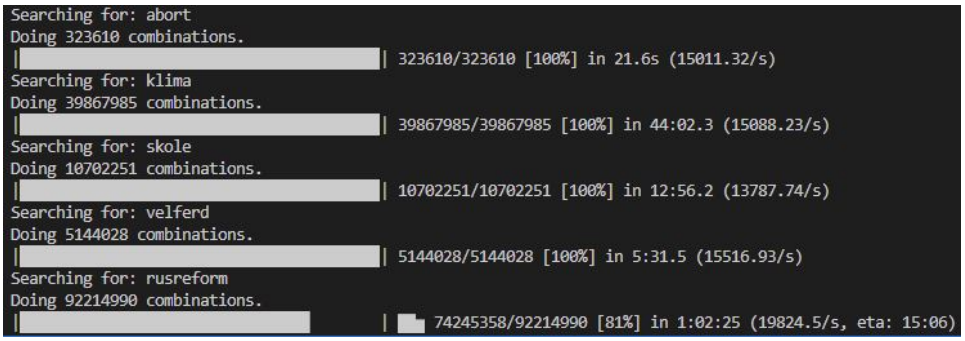

Figure 4.2: A Python script running, displaying the progress of comparing Tweets within each category. When all Tweets are compared, it moves on to the next keyword.

In the end, each comparison is saved as an individual TXT file, containing both compared strings and Twitter usernames. Some additional context was placed in the file name, such as the keyword, Twitter user IDs, and ratio score.

4.7 Twitter account relation network

Much like the network map made in the Spanish study [24], an attempt was done to try to visualize the activity between users in the data set. A Python script was created where each unique user is a node, and each mention of another user created a line between them. This naturally created a network with different clusters representing groups talking, with additional lines between clusters. An example can be seen in the figure below:

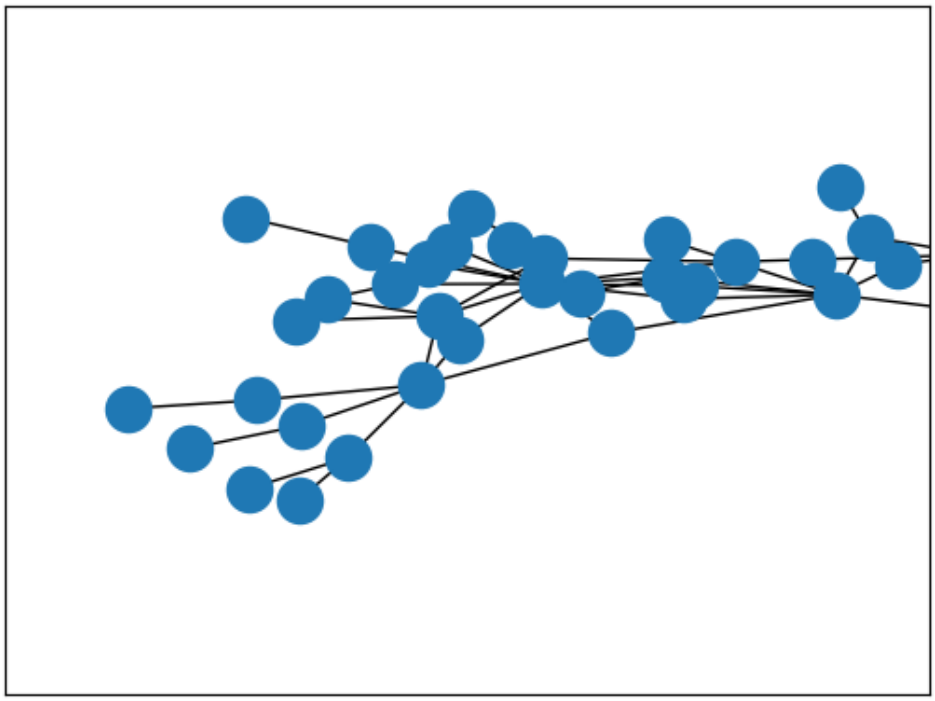

Figure 4.3: A generated node-to-node network representing Tweets from the data set. Each node is a Twitter user, and each time a Tweet mention another user, a line is created between them.

An issue arose however when there was no obvious way of grouping users. The data set is so vast that the network ended up containing more than 37 000 nodes, rendering it unusable without zoom interaction. The Spanish study's approach [24] also utilized a color grouping based on political leaning, which gave more context to the different relationships found in the network.

These issues could be fixed with various workarounds, such as using sub-sets of the data set (like the plagiarism approach), various render changes, and so on, but this could not be completed on time for this project.

Chapter 5

Results

This section will contain the results from the different experiments explained in the "Approach" section. These will be covered more in-depth and discussed in the "Discussion" section.

5.1 Graphs and diagrams

In the first figure below, we see the Norwegian flag, created by the most occurring words from the data set. Certain word groups have been filtered out, but many common words were overlooked. By looking at these words we can get a sense of the most discussed topics and how people engage when being political on Norwegian Twitter. This has to be done with a sense of confirmation bias, as the words typically align very much with the keywords used to create the data set in the first place. Still, there are relevant data points to observe, such as differences between certain keywords. For instance, both "senterpartiet" and "arbeiderpartiet" were search terms, but "arbeiderpartiet" is mentioned more often.

Figure 5.1: A word cloud was created using words from Tweets from within the data set. Certain word categories such as adverbs, pronouns, and prepositions are filtered. The words change in size depending on the occurrence. The words form a Norwegian flag in shape and color.

In the figure below we can see a bar chart displaying the top 25 users from the data set, based on Tweet count from within the data set. It starts relatively high, with a quickly descending top five, before flattening out. One could make the assumption that users that have high or unrealistic activity are automated, but that is not necessarily true. The top user tweeted over 1 400 Tweets which were picked up by the keywords. That is roughly five relevant Tweets a day over the course of nine months. For power users, that is impressive considering these are only the Tweets that were relevant to us, but it is entirely possible. This is a judgment call that is often present when analyzing user data, much like the study from the "Related work" chapter that received critique for labeling active users as "heavily automated".

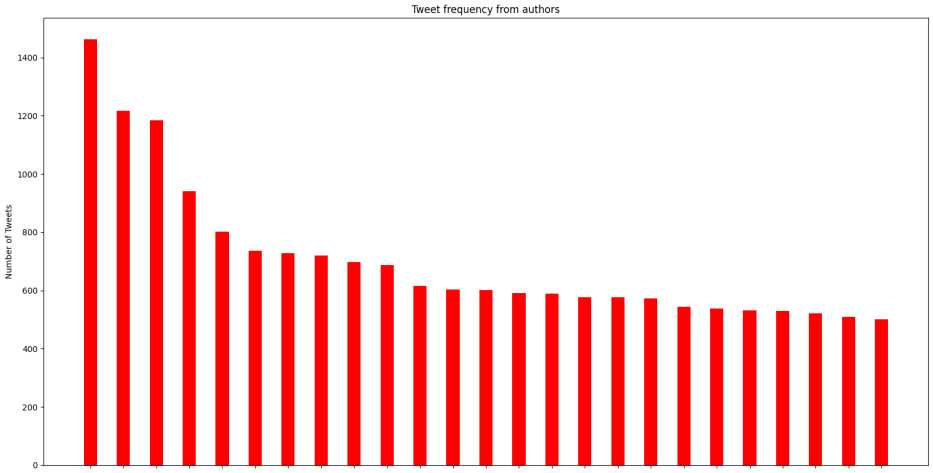

Figure 5.2: The top 25 Twitter users from the data set, are sorted by the number of Tweets tweeted from within the data set. User IDs are censored.

The figure below is an attempt made to try to visualize engagement. The Twitter users found in the data set are ranked based on a formula taking into account Tweet count, Tweet-like-count, and Tweet-retweet-count. If the Tweet count is T, the count of Tweet likes is L, and the count of Tweet retweets is R, the formula for each users engagement score is:

$$Engagement\ score = \frac{L + (2R)}{T}$$

The formula is based on the concept that generating a high Tweet count is easy, but generating high engagement per Tweet is more complex. In the formula, a retweet is valued more than a like, as a retweet is more of a meaningful engagement, effectively spreading a message further by extending it to other timelines.

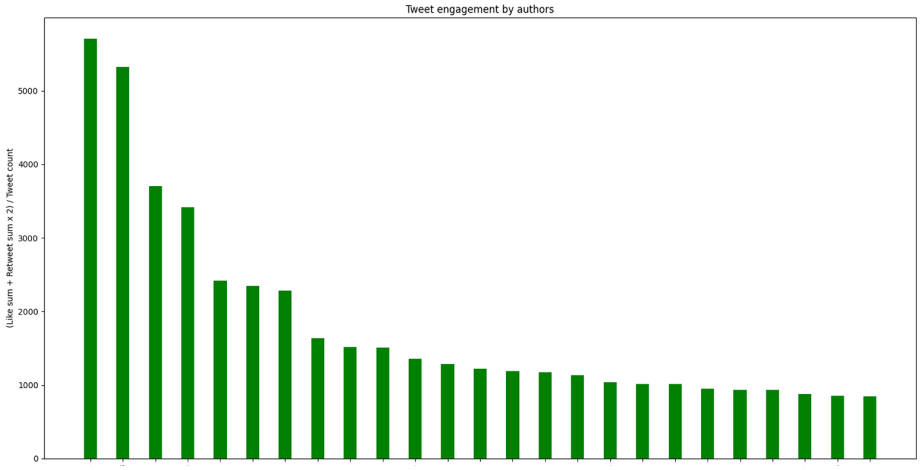

Figure 5.3: The top 25 Twitter users from the data set, sorted by the number of engagements on Tweets from within the data set. User IDs are censored. The formula is the number of likes plus the amount of retweets times two, divided by the number of Tweets.

In the figure below we can see a bar chart displaying the top 25 domains shared in Tweets from the data set. Each URL has been cleared of anything but the domain, then cleared of subdomains (such as "www"), and then the bar is increased at each occurrence. There is a clear dominance of six domains, before a drop where it flattens out at below 500 Tweets per domain.

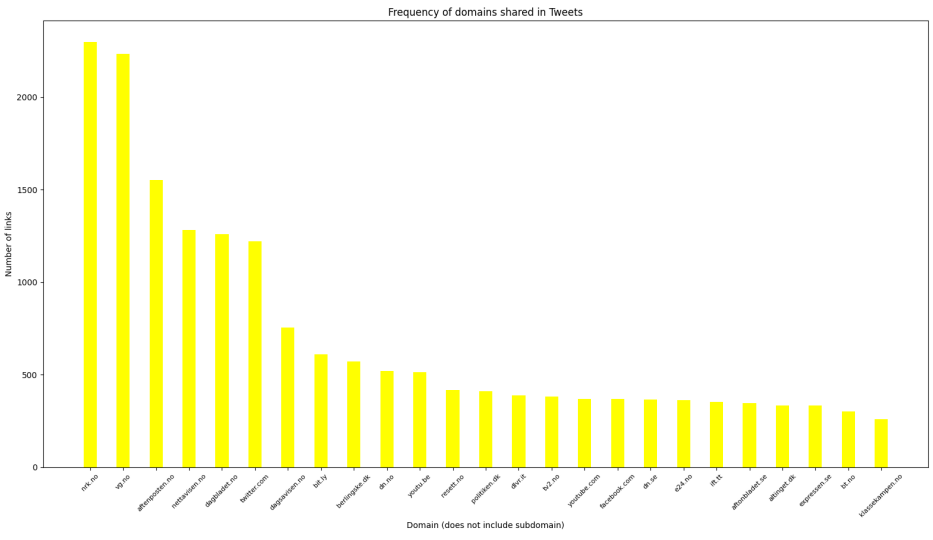

Figure 5.4: The top 25 domains are shared in tweets from the data set, sorted by the number of occurrences. Subdomains such as "www" are ignored.

In the figure below we can see a bar chart displaying each hour in a day. The bar increases each time a Tweet from the data set was tweeted within that hour (adjusted into the Norwegian time zone). This gives an overlook of user activity throughout the data set.

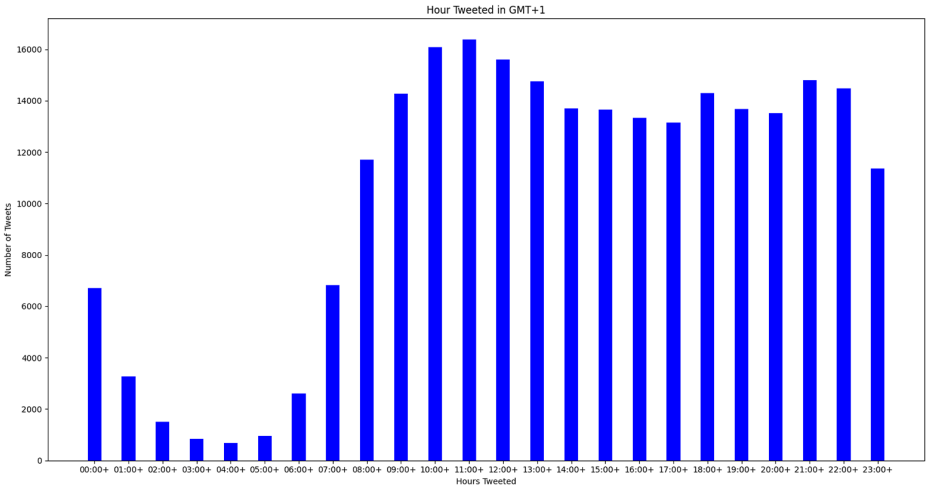

Figure 5.5: Tweet activity in the data set is displayed with one bar for each hour in a day. Time adjusted for the Norwegian time zone (GMT+1).

5.2 Botometer

The figure below displays the console output from the Python script calculating averages from the Botometer reply for each user in the data set. 242 Twitter users were unavailable for Botometer inspection, however, 37 674 users were processed. Each category was explained prior in the "Approach" section.

```
Processed 37674 users.
242 errors.

_____________________
astroturf: 0.09
fake_follower: 0.16
financial: 0.03
other: 0.30
overall: 0.23
self_declared: 0.13
spammer: 0.07
```

Figure 5.6: A screenshot showcasing the average Botometer scores for each sub-category in the universal mode, using raw scores. Trailing decimals are removed.

5.3 Plagiarism

After every Tweet within the keyword categories was compared, it generated 4 276 TXT files, each file representing a Tweet combination with a plagiarism score above 80. 3 321 of these files originate from the drug reform keyword, which seems steep, but this is mostly because the amount of Tweets related to this keyword greatly exceeds the others. This can be seen in the overview below, where the percentage of "rusreform" does not greatly exceed the other keywords.

	Unique Combinations	Amount above 80	Percentage above 80	Keyword
1	92 214 990	3 321	0,36%	rusreform
2	39 867 985	657	0,16%	klima
3	10 702 251	102	0,10%	skole
4	5 144 028	125	0,24%	velferd
5	3 409 966	61	0,18%	innvandring
6	323 610	10	0,31%	abort

Chapter 6

Discussion

This section will contain a discussion based on the results from the experiments in the thesis.

6.1 Word Cloud

Looking at the word cloud (5.1) gives a very general outlook on the types of debates that are done within the data set. We can attempt to measure the popularity of certain people and subjects based on how frequently they are mentioned, thereby scaling the word on the figure. This must be done with the possibility of inaccuracy in mind however, as many caveats make this measurement unreliable. For instance, a subject could be referred to using several different words, leading to several smaller words on the image instead of one larger one. The subject of climate change for example can be seen in the bottom right corner as both "klimapolitikk" and "klima", leading to a smaller presence than if one version of the subject combined their size. This can also be seen with the larger "rusreform" and "rusreformen", which is just the same subject referenced in definite and indefinite grammatical form.

It should also be noted that comparing the popularity of subjects is also dangerous as the scraping keywords define the possibilities. It is theoretically possible that some subject is immensely popular and debated, but was not included as a keyword, or it is commonly spelled another way, and therefore it is not represented. Such a topic would most likely still be picked up by other relevant keywords and be somewhat represented, and thereby noticed, but it is hard to know for certain.

With this in mind, the following lists are the items that appear to be mentioned the most in the word cloud, based on their visual presence. Different nicknames, reference variations, usernames, full names, and so on are merged into a common name or topic, in no particular order.

Topics:

- Drug reform
- Climate change

- Immigration
- Welfare

Parties:

- Arbeiderpartiet
- Senterpartiet
- Fremskrittspartiet
- Kristelig Folkeparti
- Rødt
- Høyre
- Sosialistisk Venstreparti
- Miljøpartiet De Grønne

Names:

- Jonas Gahr Støre
- Guri Melby
- Kenneth Arctander Johansen
- Mímir Kristjánsson
- Sylvi Listhaug
- Eivind Trædal
- Sveinung Rotevatn
- Erna Solberg

Every item listed appears consistent with what one could consider a normal political landscape, with no apparent outliers that could indicate some extremist or polarizing subjects. The topics are somewhat controversial and very relevant, which is typically why they are discussed in this setting. We can't through the word cloud know in what context these topics were discussed, but their presence is expected. Especially considering they were keywords for scraping based on their conversation-sparking controversy. If the word cloud included a lot of threatening, negative or concerning words alongside these subjects it would be a big indicator for a deeper dive into the context, but on the surface, these words seem innocent.

All the parties are big players in the political scene and both their presence and size according to each other seem consistent with the Norwegian political landscape. "Stortinget", the Norwegian parliament, has a list of candidates based on the 2021 election [34] and it coincides roughly with frequency in the data set, even though Twitter activity

shouldn't directly reflect the results of the election. Arbeiderpartiet is absolutely the largest party based on votes, with Høyre right behind. Høyre could be bigger in the word cloud, but they are split into multiple words present ("hoyre" and "høyre") and are hard to track accurately through a word cloud because their name is also a direction used in everyday Norwegian conversations.

All the users and names appearing are politically relevant, with the only person not directly politically affiliated being Kenneth Arctander Johansen. He is however the managing director of RIO [27], a Norwegian organization focusing on professional and user knowledge about drugs so that people with drug-related challenges get the help they need when they need it [28]. He is therefore very engaged in drug reform debates on Twitter and shows up on the word cloud in relation to the keyword. It would be a bigger concern if a polarizing or extremist figure was present on the word cloud, but their presence would still not be automatically concerning without delving into the context of their appearance.

In the word cloud below we filter out any Tweet not made within six weeks of election day, showcasing how the political discussion changes when an election is coming up. The figure displays a wider selection of topics, parties, and people, but still does not explicitly showcase anything that would indicate concern within the political discussion. Most new people appearing are non-politicians but are very loud regarding certain topics for the election, such as drug reform. Several still have their policy dedication listed in their Twitter biography to this day, actively acting as an activist for the topic. The fact that these people would be louder and more engaged with an election coming up is expected.

Figure 6.1: The same word cloud as earlier (5.1), but the only Tweets included are within six weeks of election day.

6.2 Top Activity Users Bar Chart

The activity figure (5.1) displays the top 25 Tweeters within the data set. Each user has been inspected manually, and even though it is hard to make a definite classification, the users appear to be genuine users, not social bots. Almost all of the top 25 are users whose account mainly focuses on political takes, and are Norwegians who are still actively tweeting to this day, changing their focus based on the current political landscape. The different accounts appear to represent multiple different political views and varying levels of political correctness, attracting different kinds of audiences. They are speaking in coherent, context-based sentences, replying to other users, and most are possible to trace back to other social media with real identities. There are no real indicators that should categorize these accounts as automated.

Other than those mentioned, one of the accounts was the Swedish account for the political party "Piratpartiet" and one account was currently suspended and unavailable. That suspended account conveys the impression of being a real human based on the Tweets in the data set, but also fairly explicit and provoking. The type of language used in the Tweets, or possibly some action performed later, most likely led to a suspension based on Twitter's harassment policy.

The data set covers a fairly large time frame, so the top user having over 1 400 Tweets relevant to the keywords ($\approx$ 5 Tweets a day) is not entirely

unrealistic, especially considering the account is very active. The same bar chart, displayed in the figure below, limits the data set to Tweets made within six weeks of election day. The majority of the users present are the same ones as before, and the new ones are within the same categories as before. The one political party from the earlier top 25 is replaced with "Liberalistene" in the new version.

The new top user tweeted on average more than eight Tweets relevant to the data set a day within a time frame of six weeks. Considering the account mainly focuses on politics and the timing is during an election, this is entirely possible and realistic. The uptick from roughly five times a day, from the previous top 25 to eight is also consistent with political relevancy related to the election.

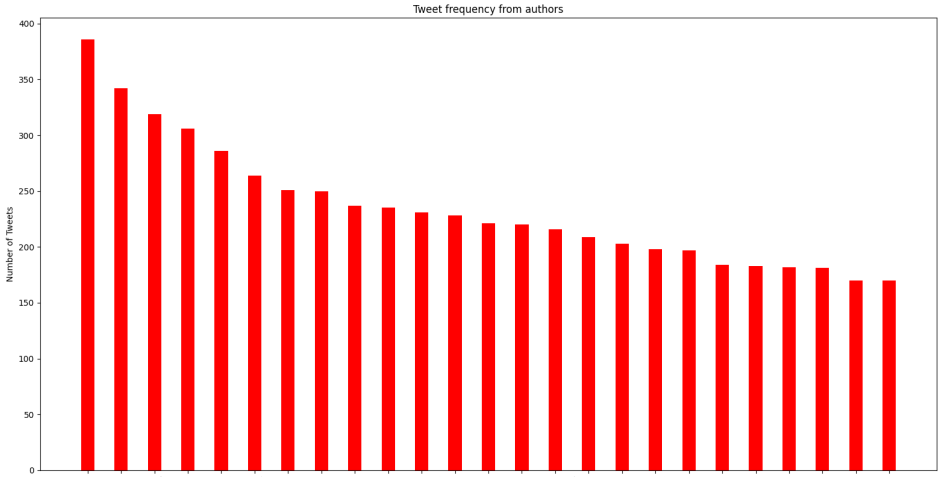

Figure 6.2: The same bar chart as earlier (5.1), but the only Tweets included are within six weeks of election day.

6.3 Top Engagement Users Bar Chart

The top 25 users generating engagement chart (5.1) attempts to visualize the most engaging users, which differ from the most active users. Any account can flood its timeline with Tweets, but generating responses consistently is more challenging. The engagement is however limited to the Tweets which were collected in this data set. If a user had two Tweets relevant to this project with high engagement, but in general receives no traction, it would not show in this graph.

All the users in the top 25 were inspected manually. A large majority were foreign individuals with large followings commenting on the election taking place in Norway. The few Tweets they shared which mentioned the process received so much engagement that they trumped any other Norwegian activity present in the data set. Only two of the 25 accounts were Norwegian, and they were celebrities, which explains their large engagement.

Unfortunately, this showcases how this was a misguided attempt at finding engaging users in the data set. The engagement generated in one or two Tweets from a large account is much bigger than the averages from popular political accounts. These accounts are likely not of celebrity-level notoriety and their longer following of politics over the big time frame severely damages their averages. A summation approach to all engagement traffic might be more interesting, but its accuracy cannot be known without testing, as this also could be trumped by large celebrity accounts.

6.4 Top Domains Bar Chart

The top 25 domains tweeted (5.4) display the most actively shared domains within the data set. The top domains start off high, with an excess of 2 000 shares, before a decline, settling at sub 500. The domains seem to be largely news publications, some social media sites, and one link shortener.

The use of a link shortener might seem nefarious, but there can be several reasons for trying to obfuscate a link before sharing it. Twitter uses a character limit on their Tweets, which often coupled with extensive takes, leaves the user crippled when trying to broadcast their entire message. If you couple this with a potentially long URL, this can severely limit your options. Using a link shortener decreases its size, makes it more visually appealing, and in some cases, allows statistical tracking for the creator.

Of course, like other exploratory data analyses done on this data set, without the context of these Tweets it can be challenging to determine why any of these domains were shared. The display of several big-name news publications in the top five does seem to advocate both good source criticism and topical discussions, which bode well for the conversations in the data set. The domains also include "resett.no" however, which has been critiqued for being a factually wrong and far-right extremist news site [36]. Their presence doesn't seem out of place in spite of their controversial existence. They have a large following, receiving 560 000 visitors in march of 2018 according to themselves [18], and much like with the other publications, we do not know the context of their sharing. It could just as likely be a response to their controversial articles as an endorsement of their writing.

Even if the context of "resett.no" Tweets were validated and showed genuine enthusiasm and trust in their content, this also would not necessarily mean social bots are present. A deeper analysis of the accounts could be done, checking their behavior and comparing to other accounts, but these accounts could just as easily be real accounts that share the viewpoints of the publication. Social bots are known to be polarizing and dubious, but that does not mean political differences and arguments prove they are present. Political controversy has existed long before social media existed.

6.5 Activity Bar Chart

The datasets activity bar chart (5.1) shows fairly consistent Tweet activity throughout the day, starting at about 09:00, and calming down at about 22:00. The peak tweeting hour is at 11:00 with over 16 000 ($\approx$ 6.4%) Tweets being created within that hour. 00:00 through 07:00 is a consistent dip and heavy increase leading to the peak at 09:00. According to an article on "SproutSocial" [14], the best time to post on Twitter is at 09:00, based on engagement figures from Twitter. This is fairly consistent with the peak post time from within the data set. In the heat map below, visualizing their data, one can see the main dip of blue in the afternoon is at 21:00, and the night is fairly quiet, arguably ramping up a bit earlier than the Tweets within the data set.

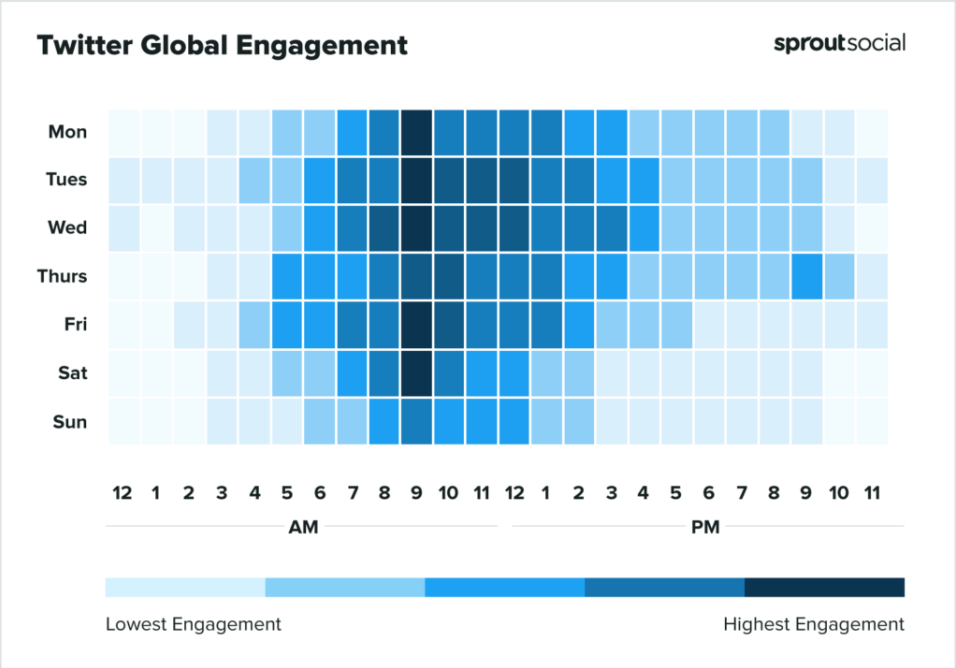

Figure 6.3: Global heat map from SproutSocial visualizing engagement from users on Twitter throughout the week.
[14]

The similarities can be seen as a point toward a well-targeted data set, as a large influx of irrelevant Tweets would most likely deviate the activity from the norm. The exception would of course be noise from countries using the same time zone, such as neighboring countries. Since they also share many common names and words, they are also likely to pollute the data set, so the possibility of their presence can therefore not be ignored.

It also displays how a potential major bot operation would have to be programmed to take time of day into consideration to stay hidden, as current Tweet activity seems fairly consistent with human behavior. This makes the presence of simple bot automation unlikely or negligible.

According to an article from "websiterating.com" [2], "80% of active Twitter users access the site via mobile". This coincides fairly well with the consistent usage during the day as the platform is easy to access, even at normal working hours.

It could be a social bot campaign wanting to alter public perception would be less long term, and be active when the election is closer and actively discussed. Below you can see the same bar chart, but only looking at Tweets created within six weeks of election day. Comparing the two figures can be used to showcase the Twitter posting differences between the year average and the election run-up.

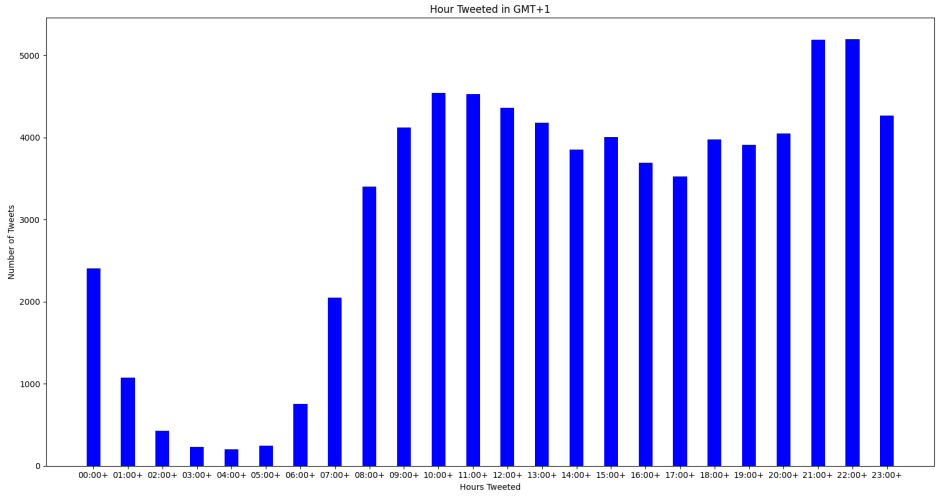

Figure 6.4: The same bar chart as earlier (5.1), but the only Tweets included are within six weeks of election day.

The bar charts are very similar, but six weeks within the election activity peaks at a later time. Most Tweets are definitely posted from 21:00 through 22:00. This coincides well with late evening political television and most people's free time allowing for Twitter discussions. The Norwegian Broadcasting Corporations (NRK) schedule leading up to the election [21] had several political programs starting in this time frame, which would contribute to Twitter discussions. This is especially true as a common Twitter phenomenon is to "live-tweet" [37] during an event for discussions and reactions.

6.6 Botometer

The averages shown in the output (A.9) of the Python script do point to a data set that has a majority of low-scoring accounts in Botometer's API. The biggest average is in the category "other" which is 0.30.

Averages do not show the entire story however, as the point of social bots is to blend into the majority. Therefore inspection of the highest-

ranking Botometer results is needed for a better insight into the service Botometer provides. The manual inspection leads to varying types of accounts, further creating confusion on how to interpret the results given by Botometer. This could be explained by the testing showcased in the "Related works" chapter [25], which showed Botometer's increased struggle with foreign language accounts.

The most valuable result to look at seemed to be "self_declared" as this regularly leads to different types of bots that accidentally were scraped into the data set. This seemed most reliable when the Botometer score in this category was 1.0. These are bots with simple functions who also state their purpose and presence in their Twitter biography. Bots such as random book ideas, automatic headline posting, acronym generators, and so on. Even so, regularly a high score of the "self_declared" category (0.80-0.99) would lead to any random account, with little to no evidence of appearing as a bot, declaring themselves as a bot, or any reference on "botwiki.org". Effectively a false positive, with almost zero indicators of being a bot.

With other categories it hard to determine what Botometer is looking at to reach their conclusion. For instance, one Twitter account had these category results above 0.8:

- fake_follower - 1.0
- other - 0.9488333333333334
- overall - 1.0
- self_declared - 0.84

Investigating the account there are multiple details to take into consideration. The account doesn't have a profile photo, the account was created in 2016, the username is a first name followed by eight seemingly random numbers, the profile name is the first name from the username, and it follows 12 accounts. The account likes two Tweets, both from 2021, and made one reply (from 2021), which is to one of the two Tweets it liked. The 12 accounts followed are news outlets, politicians, and police district accounts.

Based on this it is understandable that Botometer might suspect it is a fake follower account, being used to boost follower counts for, potentially, paying customers. However, there is no overwhelming proof, just many indicators. There are no Tweets, basically no activity, little to no personal information and it follows a couple of related accounts. Those could be considered big indicators. But also, it joined in 2016, and we don't know the "following" timeline of the account. It is theoretically possible this is a very inactive, but real account. Also, it made a fully coherent, related reply to a Tweet, which was most likely done manually. It could be this is an old account that was hacked, wiped, followed a few accounts, did some minimal activity, and then abandoned, but there is no way to know. There is also no information on the account that should lead Botometer to interpret this as a self-declared bot.

Another comparable case had a "fake_follower" score of 1.0 and an account with very similar details. Little activity, old account, follows several accounts. However, this account had a real name and profile photo, which could be traced back to a LinkedIn account that had corresponding details. This is most likely not a "fake follower" bot, just an account with peculiar activity. It does not give faith in the earlier classification when this one seems clearly misguided. If Botometer operates on probability and indications, it seems odd that these accounts should have such different feasibility of being bots.

These examples showcase the inconsistencies between Botometer results and how it is hard to determine what Botometer uses to classify the categories. Botometer most likely has a complicated internal process which determines based on many factors, so there seems to be a level of trust required if it is to be utilized. Especially because the software is closed-source. This is not a major test that utilizes known bots, known humans, and success rates, like the ones mentioned in the "Related work" chapter [25], but the overall impression does not give much faith in their accuracy.

6.7 Plagiarism

A large number of Tweets detected using plagiarism were users utilizing the share button on news sites. This is a button typically available on articles for sharing to social media. Typically this creates a Tweet with the headline as text with the relevant URL at the end. If multiple people share the same article without altering the text this results in multiple identical Tweets, which in turn were detected by the plagiarism script.

Another group of plagiarism Tweets seemed to be a form of political activism utilized by engaged individuals. A group of people specializing within certain political topics will reply to events, articles, and Tweets with a hashtag, or set of hashtags. This occurs frequently within the drug reform keyword. The main goal seems to be to relate the topic of the original content to the political topic. The accounts were inspected and gave no indications of using automation to publish content in general. The basic replies, such as "#rusreform", were sometimes responded to Tweets that could not be associated with the relevant topic without understanding the text or watching a video.

During the phase where the plagiarism script was developed, it was executed for a few days, trying to compare every possible Tweet. This approach was abandoned, but in that time frame, a bot was actually discovered. It was a bot actively replying to varying political Tweets, with the same content every time. The Tweet was a message, urging Sweden to change its Covid-19 strategy involving herd immunity, followed by a series of hashtags related to Covid-19 and European countries. The social deception is minimal however, as the bot itself states in a Tweet that the plan is to "spread the message to make the world a better place".

This is a clue that plagiarism does have the ability to help discover basic bots, even though this one had no level of variation in its message. As with

all other users in the data set, this bot was processed by Botometer, but the results were not investigated until the plagiarism script showcased the account. The Botometer scores are as follows:

- fake_follower: 0.44
- self_declared: 0.4
- astroturf: 0.15
- spammer: 0.19
- financial: 0.15
- other: 0.73

These scores are not extremely high, especially compared to how difficult it was to find real, convincing bots in the pool of users with higher scores than this. The main score which would indicate this account is a bot is "other", which is a very vague category. As this was discovered because of the account's repetitive posting of the same content, it is hard to conceive why this bot wasn't identified using the "spammer" category, as that is essentially what this account is doing. However, that category only has a score of 0.19.

6.8 Overall

Based on all the different charts, diagrams and results, it does not seem to be any convincing evidence that bot activity or social deception occurred during the nine-month period leading up to the election. The word cloud, even though it is a rough overview, gives no clues that indicate suspicious activity. There are no bots posting excessively, spamming more than any normal user can post over a long period of time. There are no polarizing domains towering over other domains in activity rate which differ from expected societal expectations. There is no unusual posting activity indicating large-scale automated processes. There is little to no plagiarism indicating bots, and the one bot found was even upfront, removing any possibility of social deception. The majority of the plagiarism results have logical explanations. Even Botometer gives no overwhelming indicator that bots are haunting the data set.

Saying no deception or automated activity could be present in this data set is naive, however. A majority of this investigation is exploratory data analysis which heavily relies on human intuition and is limited by our ability to conceive the possibility of deception and automation. What is discovered is more up for interpretation and should be considered clues more than evidence. It is absolutely possible that there is a set of heavily automated accounts in the data set, creating coherent, unique Tweets, possibly using machine learning, posting them at acceptable times of day, slowly but surely driving a campaign to skew and alter public perception.

We know there is a motivation for this, as seen in similar attempts, and we know the technology exists to make it possible. It is however, extremely unlikely and based on this analysis and very little evidence for it.

One could argue this thesis did little to actually search for social bots and instead was in pursuit of regular bots or automated activity. This is true to some degree as most of the glaring indicators one would be able to see would expose basic automation activity. However, hunting for social bots does not seem to be a precise recipe at this point, as a majority of research heavily relies on automated tools such as Botometer, which was also utilized here. This would put this in the same category as those studies, except that additional perspectives and tools were taken advantage of here. The presence of social bots was mostly discredited by using manual inspection and this could be labeled as imprecise and imperfect, but at least compared to Botometer this is an open process with clues, indicators, and arguments, whereas Botometer needs to be blindly trusted.

It should be noted that Twitter was chosen as a data source for this investigation for its political presence in culture, open platform, and ease of access to API/Data. It does not represent all political presence and discussion online in Norway. Even if we compare to data from the US, cultural differences could result in different levels of usage, degrees of importance placed on information, and different demographics present on the platform. We also can't know how big of a chunk of the online political discussion in Norway is represented by the data. These results, therefore, act more like a sample and insight into what can be found on Twitter, which then can be used to make estimates and assumptions. Making a factual argument about these results without mentioning this issue would be a misrepresentation.

Social media usage is different around the world, and Twitter has by no means the same place in Norway as it does in the United States. A bigger investigation, into several popular social media sites, would lead to much more accurate and reliable results for Norway as a society.

6.9 Future work

There have been several attempts in this thesis to justify the patterns found and advocate for what potential bots and deception would look like if discovered. Both these perspectives could be delved into more and be tested in a more technical fashion. For instance, by using control groups we can much more reliably discover activities that stand out and cause concern within the data set, without having to rely on human definitions of what is normal behavior.

Multiple overviews were looked at during this investigation, and while interesting, some of them would require more context-based inspections. For example the sharing of different domains. Both credible and less credible domains were present, but how were they shared? Looking for deviations in expected behavior and debating what should and should not be present in statistics does not tell the whole story, and a study with more

time could try to better classify and group different behaviors within a data set such as this.

This thesis has also clearly showcased how social bot research is controversial and is in need of more clear definitions, tools, and data sets. Relying exclusively on tools such as Botometer is unwise and perhaps a more open-source approach needs to be developed for more precise conclusions.

The limited time frame available for this thesis leaves many possibilities regarding the results open. The data set is vast and the opportunities for analysis are huge, therefore it should ideally be experimented more on. Because of privacy concerns, this data set cannot be shared, unfortunately. This is a problem that faces multiple studies, and was discussed heavily in "The Rise and Fall of 'Social Bot' Research" [10]. However, all the code is available to be altered and used further. The data set could be recreated using all the Tweets which are still publicly available and requires no API access by Twitter to be utilized. This lays the foundation for further expansion of this thesis, possibly giving other investigators the tools to do more research in the future.

Chapter 7

Conclusion

Based on this insight into the political landscape present on Twitter in 2021, there does not seem to be any big indicators that any foul play or social deception campaign targeted the 2021 Norwegian election on Twitter. The different breakdowns are a rough perception, but there are little to no clues that would indicate odd or unexpected activity on the platform. This does not mean that none exists, here or some other place, but instead illustrates how misguided social bot research currently is. There doesn't appear to be any agreed-upon approach, and the results are always to some degree debatable, but that could perhaps be applied to all science.

Twitter does in no way represent all discourse, or even a majority of discourse, within Norwegian politics, but does however carry a label as a platform for discussions and personal takes. It's targeting in a bot campaign would likely be based on this and how open content is on the platform, making it easy to both automate processes and fit into the timeline. If an approach can be found which is deemed effective on Twitter, this could in turn be altered and be applied to all social media as a whole.

Some governance and compliance might be necessary for social media developers such as Twitter, as their services today are becoming more and more important in society. If Twitter acts as a public square, enabling democratic discussion, should it be allowed to do whatever it pleases? One study in the "Background" chapter [31] argued platforms could do more, for instance utilize more CAPTCHAs to validate human interaction. This doesn't appear to align with Twitter's own approach, however. The platform has developed several API endpoints and self-declared Twitter bots are quite popular within the community. For now, it doesn't seem like Twitter is attempting to restrict the usability of programming on their platform.

If some authority created a compliance rule-set, this could in turn lead to more reliable platforms and effective investigations into the subject. Much like how GDRP in the EU upholds data protection laws, making companies responsible when non-compliant. This thesis has however, showcased how difficult it is to prove and discover social bots, which in turn makes compliance more complicated. How can we expect Twitter to fight social bots if we have difficulty proving their existence ourselves?

A collaboration between Twitter and a scientific study would still be interesting, anyhow. Twitter does have access to a lot of data and their cooperation could be a big aid in the research of social bots.

The reality that there is technology and motivation to out-wit an analysis such as this, does leave the lack of evidence with multiple degrees of uncertainty. Still, the few amounts of discoveries for deception do imply some comfort, as some automated activity was found. This could also be interpreted as an indicator that social deception already has transcended our ability to detect it, and one no longer can trust the content found online. A conclusion such as this might be a form of belief perseverance however, as finding no proof can be judged as an indicator that something is wrong, and as an indicator that nothing is wrong. This is most likely affected by your confirmation bias before investigating the concept.

The "Background" chapter covered in some detail how the world currently is becoming increasingly aware of the threat of social bots online, yet little concrete evidence can be discovered. This thesis found little to no indications of actual deception, which doesn't appear to align with concerns regarding the current online landscape. These results are however consistent with the study named "The Rise and Fall of 'Social Bot' Research" [10] discussed in "Related work", which acted as a critique of current social bot research. It could perhaps be that the increased awareness makes it more challenging for social bots currently than in the 2016 US election when the concept was relatively unknown. However, it could also be that social bot use is rampant, but this smaller, Norwegian election was not a target. An interesting comparison could be done on other data sets, perhaps proving this. The lack of evidence here does not mean they cannot exist.

Bibliography

[1] 0301. *ordliste*. 2012. URL: https://github.com/0301/ordliste (visited on 22/04/2022).

[2] Matt Ahlgren. *50+ Twitter Statistics Facts For 2022*. 2022. URL: https://www.websiterating.com/research/twitter-statistics/#chapter-2 (visited on 25/04/2022).

[3] Monther Aldwairi and Ali Alwahedi. 'Detecting Fake News in Social Media Networks'. In: *Procedia Computer Science* 141 (2018). The 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops, pp. 215–222. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.10.171. URL: https://www.sciencedirect.com/science/article/pii/S1877050918318210.

[4] Larry Alton. 'Why email remains the top communication tool for businesses'. In: (2017). URL: https://theamericangenius.com/business-news/email-remains-top-communication-tool-businesses/.

[5] 'API'. In: (2022). URL: https://botometer.osome.iu.edu/api.

[6] NORSK FOR DEG! AS. *Nå begynner vi! (A1-A2)*. 2022. URL: https://norskfordeg.no/ressurser/na-begynner-vi/ (visited on 22/04/2022).

[7] Pete Cashmore. *MySpace, America's Number One*. 2006. URL: https://mashable.com/archive/myspace-americas-number-one (visited on 25/02/2022).

[8] Fagbokforlaget. *Grammatikk*. 2022. URL: https://minvei.no/read_container/88b030b3-0925-4bc4-b252-81d295f95e4b (visited on 22/04/2022).

[9] Python Software Foundation. *itertools — Functions creating iterators for efficient looping*. 2022. URL: https://docs.python.org/3/library/itertools.html (visited on 16/05/2022).

[10] Florian Gallwitz and Michael Kreil. In: *The Rise and Fall of 'Social Bot' Research* (Mar. 2021), pp. 1–19. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3814191.

[11] *Google Trends*. 2022. URL: https://trends.google.com/trends/explore?date=2010-01-01%202022-01-01&q=fake%20news (visited on 25/02/2022).

[12] Red Hat. *The open source way*. 2022. URL: https://opensource.com/open-source-way (visited on 16/05/2022).

[13] TOM JACOBS. 'THE LINGERING EFFECT OF NAZI PROPAGANDA'. In: (2015). URL: https://psmag.com/social-justice/the-lingering-effect-of-nazi-propaganda.

[14] Mary Keutelian. *The best times to post on social media in 2022*. 2022. URL: https://sproutsocial.com/insights/best-times-to-post-on-social-media/#twitter-times (visited on 25/04/2022).

[15] Young Mie Kim et al. 'The Stealth Media? Groups and Targets behind Divisive Issue Campaigns on Facebook'. In: *Political Communication* 35.4 (2018), pp. 515–541. DOI: 10.1080/10584609.2018.1476425. eprint: https://doi.org/10.1080/10584609.2018.1476425. URL: https://doi.org/10.1080/10584609.2018.1476425.

[16] Dokyun Lee, Kartik Hosanagar and Harikesh Nair. *Advertising Content and Consumer Engagement on Social Media: Evidence from Facebook (June 5, 2017)*. SSRN, 2017. DOI: http://dx.doi.org/10.2139/ssrn.2290802. URL: https://ssrn.com/abstract=2290802.

[17] Terry Lee. 'The global rise of "fake news" and the threat to democratic elections in the USA'. In: *Public Administration and Policy* 22.1 (Jan. 2019), pp. 15–24. ISSN: 1727-2645. DOI: 10.1108/PAP-04-2019-0008. URL: https://doi.org/10.1108/PAP-04-2019-0008.

[18] Helge Lurås. *Resett når 560 000 unike besøkende i mars 2018*. 2018. URL: https://resett.no/2018/04/02/Resett-nar-560-000-unike-besokende-i-mars-2018/ (visited on 08/05/2022).

[19] manilde. *TWINT - Twitter Intelligence Tool*. 2022. URL: https://github.com/manilde/twint (visited on 22/04/2022).

[20] L.V. Mureyko et al. 'The correlation of neurophysiologic and social mechanisms of the subconscious manipulation in media technology'. In: *International Journal of Civil Engineering and Technology* 9 (Aug. 2018), pp. 1–9.

[21] NRK. *Valg 2021*. 2021. URL: https://tv.nrk.no/serie/valg-2021/202109 (visited on 28/04/2022).

[22] J. Eric Oliver and Thomas J. Wood. 'Conspiracy Theories and the Paranoid Style(s) of Mass Opinion'. In: *American Journal of Political Science* 58.4 (2014), pp. 952–966. DOI: https://doi.org/10.1111/ajps.12084. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/ajps.12084. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/ajps.12084.

[23] Esteban Ortiz-Ospina. *The rise of social media*. 2019. URL: https://ourworldindata.org/rise-of-social-media (visited on 25/02/2022).

[24] Javier Pastor-Galindo et al. 'Spotting Political Social Bots in Twitter: A Use Case of the 2019 Spanish General Election'. In: *IEEE Transactions on Network and Service Management* 17.4 (2020), pp. 2156–2170. DOI: 10.1109/TNSM.2020.3031573.

[25] Adrian Rauchfleisch and Jonas Kaiser. 'The False positive problem of automatic bot detection in social science research'. In: *PLOS ONE* 15.10 (Oct. 2020), pp. 1–20. DOI: 10.1371/journal.pone.0241045. URL: https://doi.org/10.1371/journal.pone.0241045.

[26] Alejandro Reyes. *Technology: Truth or Consequences*. 2000. URL: http://edition.cnn.com/ASIANOW/asiaweek/intelligence/2000/05/26/ (visited on 25/02/2022).

[27] RIO. *Ansatte*. 2022. URL: https://rio.no/om-oss/ansatte/ (visited on 28/04/2022).

[28] RIO. *Om Oss*. 2022. URL: https://rio.no/om-oss/ (visited on 28/04/2022).

[29] Edward Roberts. 'Is Web Scraping Illegal? Depends on What the Meaning of the Word Is'. In: (2018). URL: https://www.imperva.com/blog/is-web-scraping-illegal/.

[30] seatgeek. *thefuzz*. 2022. URL: https://pypi.org/project/thefuzz/ (visited on 16/05/2022).

[31] Chengcheng Shao et al. 'The spread of fake news by social bots'. In: (July 2017).

[32] StatCounter. *Desktop Search Engine Market Share Worldwide*. URL: https://gs.statcounter.com/search-engine-market-share/desktop/worldwide/2021 (visited on 25/02/2022).

[33] David Sterrett et al. 'Who Shared It?: Deciding What News to Trust on Social Media'. In: *Digital Journalism* 7 (June 2019), pp. 1–19. DOI: 10.1080/21670811.2019.1623702.

[34] Stortinget. *Politiske partier på Stortinget*. 2022. URL: https://www.stortinget.no/no/Representanter-og-komiteer/Partiene/Partioversikt/ (visited on 28/04/2022).

[35] Øystein Aune Sverre. *oslomet_thesis*. 2022. URL: https://github.com/aunefyren/oslomet_thesis (visited on 07/05/2022).

[36] Markus Tobiassen. *Bryter med den alternative nettavisen Resett*. 2021. URL: https://www.vg.no/nyheter/innenriks/i/Xqa0pr/bryter-med-den-alternative-nettavisen-resett (visited on 08/05/2022).

[37] Tony Tran. *How to Live Tweet an Event: Tips, Best Practices, and Examples*. 2019. URL: https://blog.hootsuite.com/how-to-live-tweet/ (visited on 28/04/2022).

[38] Twitter. *Developer account support*. URL: https://developer.twitter.com/en/support/twitter-api/developer-account (visited on 16/03/2022).

[39] Federico Vegetti and Levente Littvay. 'Belief in conspiracy theories and attitudes toward political violence'. In: *Italian Political Science Review/Rivista Italiana di Scienza Politica* 52.1 (2022), pp. 18–32. DOI: 10.1017/ipo.2021.17.

List of Figures

Appendix A

Appendix

All the code in this appendix can also be accessed on GitHub [35].

A.1 add_commas.py

When "Twint" generated the JSON files used for the data set, the files were not in the standard JSON format. Each JSON line was a unique Tweet, but there were no brackets wrapping them, and no commas in-between. This Python script added a comma to every line, except the last one. Then brackets are added to each end of the file.

```
# import required module
import os

# Assign variables
directory = 'tweets'
tweets_proccessed = 0

# Iterate over files in tweets directory, checking each file
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # If the path is a file
    if os.path.isfile(f):

        # Open file and read the induvidual lines
        with open(f, 'r', encoding="utf8") as file:
            # read a list of lines into data
            data = file.readlines()

        # For each line, replace the 'newline' with ',newline',
    except if it's the last line
        for i, row in enumerate(data):
            if i != len(data)-1:
                data[i] = row.replace("\n", ",\n")


        # Add bracket to start of file and ending
        data.insert(0, '[')
        data.append(']')


```

```
        # Write everything back to orignal file
        with open(f, 'w', encoding="utf8") as file:
            file.writelines( data )
```

A.2 combine_unique_json.py

When Tweets were collected by "Twint" they were stored in individual files, based on the current keyword in use. This script goes through every JSON file in the folder, collects the Tweets, removes duplicates, and saves the final data set to a new final file.

```
# import required module
import os
import json

# Assign variables
directory = 'tweets'
tweets_proccessed = 0

# Iterate over files in tweets directory, checking each file
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # If the path is a file
    if os.path.isfile(f):

        # Generate list of ID's already in masterfile, tweets.
    json
        id_array = []
        with open('tweets.json', encoding='utf-8') as file:
            tweets = json.load(file)
            for i, row in enumerate(tweets):
                id_array.append(tweets[i]['id'])

        # Go through each Tweet in current file
        with open(f, encoding='utf-8') as file:
            data = json.load(file)
            tweets_proccessed = tweets_proccessed + len(data)

            # Loop through each Tweet in file
            for i, row in enumerate(data):
                found = False

                # Check for the Tweet ID in the already
    established ID array
                for j in id_array:
                    if j == data[i]['id']:
                        found = True
                        break

                # If not found in the ID list, add to master
    file
                if not found:
                    tweets.append(data[i])

        # Write new version of tweets.json
        with open('tweets.json', 'w', encoding="utf8") as file:
            json.dump(tweets, file)
```

```
44
45
46 # Sort final array using created_at
47 tweets.sort(key=lambda x: x["created_at"])
48
49 # Write new sorted version of tweets.json
50 with open('tweets.json', 'w', encoding="utf8") as file:
51     json.dump(tweets, file)
52
53
54 # Print some stats of Tweets
55 print('Induvidual Tweets proccessed: ', tweets_proccessed)
56 print('Final Tweet count: ', len(tweets))
57 print('Tweet duplicates found: ', (tweets_proccessed-len(tweets
    )))
```

A.3 download_tweets.py

This Python script utilizes "Twint" to scrape Tweets from Twitter. The script loads the CSV file containing search strings (keyword), collects every Tweet it applies to within the correct time frame and saves all the results to a file.

```
1  import pandas as pd
2  import twint
3  import datetime
4  start_date = datetime.date(2021, 1, 1)
5  end_date = datetime.date(2021, 9, 15)
6
7  # Read the CSV file containing search strings, and loop through
       each one
8  df = pd.read_csv('search_terms.csv', encoding = 'utf8', sep=';'
     )
9  for index, row in df.iterrows():
10     print('Searching for: ', row['search'])
11
12     c = twint.Config()
13
14     c.Store_json = True
15     c.Count = True
16     c.Stats = True
17     c.Store_object = True
18     c.User_full = True
19
20     # Define search to use current string and limit the search
       to within the time frame
21     c.Search = row['search']
22     c.Since = start_date.strftime('%Y-%m-%d') + ' 00:00:00'
23     c.Until = end_date.strftime('%Y-%m-%d') + ' 00:00:00'
24
25     # Perform a search and save the results in a unique file
26     c.Output = "tweets/tweets_term_" + str(index) + ".json"
27     twint.run.Search(c)
```

A.4 generate_word_cloud.py

As an attempt to visualize the conversation, this Python script collects every word in every Tweet in the data set. Some words from predefined lists are filtered out. Then a visual word cloud is generated, where each occurrence of a word grows its size. The word cloud takes the form of a Norwegian flag and is then displayed.

```
# Get modules
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_gradient_magnitude
import pandas as pd
import json
import numpy as np
from PIL import Image
from os import path
import os
from datetime import date

# Define variables
comment_words = ''
stopwords = set(STOPWORDS)
directory = 'words'
last_six_weeks = True
election_date = date.fromisoformat('2021-09-13')

# Import PNG of Norwegian flag for mask and coloring
d = path.dirname(__file__) if "__file__" in locals() else os.
    getcwd()
norwegian_flag = np.array(Image.open(path.join(d, "
    norwegian_flag.png")).convert('RGB'))

# create mask  white is "masked out"
norwegian_flag_mask = norwegian_flag.copy()
norwegian_flag_mask[norwegian_flag_mask.sum(axis=2) == 0] = 255

# some finesse: we enforce boundaries between colors so they
    get less washed out.
# For that we do some edge detection in the image
edges = np.mean([gaussian_gradient_magnitude(norwegian_flag[:,
    :, i] / 255., 2) for i in range(3)], axis=0)
norwegian_flag_mask[edges > .08] = 255

# Iterate over files in words directory, checking each file and
     adding words to stopwords
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # If the path is a file
    if os.path.isfile(f):
        text = open(f, encoding="utf-8").read()
        stopword_array = text.split()
        for word in stopword_array:
            stopwords.add(word.lower())


# Generate variable with words found in Tweets within tweets.
    json
```

```
with open('tweets.json', encoding='utf-8') as file:
    tweets = json.load(file)
    for i, row in enumerate(tweets):
        if not tweets[i]['retweet']:

            if last_six_weeks:
                split_tweet_time = tweets[i]['created_at'].
    split()
                tweet_time = date.fromisoformat(
    split_tweet_time[0])
                duration = election_date - tweet_time
                duration_in_s = duration.total_seconds()
                days = duration.days
                if days > 42:
                    print("Skipped date: " + tweets[i]['
    created_at'])
                    continue
                else:
                    print("Didn't skip date: " + tweets[i]['
    created_at'])


            user_id = tweets[i]['user_id']

            # split the value
            tokens = tweets[i]['tweet'].split()

            # Converts each token into lowercase
            for i in range(len(tokens)):
                tokens[i] = tokens[i].lower()

            comment_words += " ".join(tokens)+" "


# Create wordcloud, 2000x2000, grey background color, no
    collocations, none of the stopwords
wordcloud = WordCloud(width = 2000, height = 2000,
        background_color ='white',
                mask = norwegian_flag_mask,
        stopwords = stopwords,
                collocations = False,
        min_font_size = 10).generate(comment_words)

# Create coloring from image
image_colors = ImageColorGenerator(norwegian_flag)

# Create and show grapich, color using the coloring created
    earlier
plt.figure()
plt.imshow(wordcloud.recolor(color_func = image_colors),
    interpolation="bilinear")
plt.axis("off")
plt.show()
```

A.5 tweet_author_botometer.py

This script collects every unique Twitter user ID from the data set, retrieves their Botometer scores using their API, and then saves the result to a file. It also checks the saving-directory for Botometer scores before attempting to retrieve scores from Botometer, skipping additional downloads in case of multiple script executions. If Botometer already gave a score, but the result was an error, it retries to fetch the scores.

```
import botometer
import matplotlib.pyplot as plt
from datetime import datetime
from dateutil import tz
import pytz
import json
import os

rapidapi_key = "XXX"
twitter_app_auth = {
    'consumer_key': 'XXX',
    'consumer_secret': 'XXX',
    'access_token': 'XXX',
    'access_token_secret': 'XXX',
  }
bom = botometer.Botometer(wait_on_ratelimit=True,
                          rapidapi_key=rapidapi_key,
                          **twitter_app_auth)


data = []

# Generate array with unique Twitter authors
with open('tweets.json', encoding='utf-8') as file:
    tweets = json.load(file)
    for i, row in enumerate(tweets):
        # Create Datetime object using the syntax Twitter uses, set timezone to UTC
        if not tweets[i]['retweet']:
            user_id = tweets[i]['user_id']
            if user_id not in data:
                data.append(user_id)


# Print the amount of authors found in dataset
print("Proccessed " + str(len(data)) + " authors.")

path = 'botometer'

# Iterate over files in 'botometer replies' directory, checking each file with results
for filename in os.listdir(path):
    f = os.path.join(path, filename)
    # If the path is a file
    if os.path.isfile(f):

        # Go through each user id in current file. If Botometer already has downloaded the data, remove it from the list
    of accounts to request.
```

```
        with open(f, encoding='utf-8') as file:
            file_data = json.load(file)
            if "error" in file_data:
                print("Twitter user ID had error in it.")
                continue
            else:
                if int(file_data["user"]["user_data"]["id_str"
    ]) in data:
                    data.remove(int(file_data["user"]["
    user_data"]["id_str"]))
                    print("Skipped Twitter user ID: " +
    file_data["user"]["user_data"]["id_str"])


# For each unique account in the list, call Botometer API and
    save the result in a file using the author Twitter ID
for screen_name, result in bom.check_accounts_in(data):
    # Write new version of tweets.json
    with open(path + '/' + str(screen_name) + '.json', 'w',
    encoding="utf8") as file:
        json.dump(result, file)
```

A.6 tweet_author_engagement.py

As an attempt to visualize engagement, each user in the data set has an engagement score calculated. The script retrieves every Tweet in the data set and saves certain data for each unique Twitter user. The amount of Tweets, likes received, and retweets received. This generates a score for the user where the sum of likes and the sum of retweets times two are divided by the sum of Tweets. The users are then sorted by most engagement and the top 25 users are displayed in descending order.

```
import matplotlib.pyplot as plt
from datetime import datetime
from dateutil import tz
import pytz
import json

# Create class for creating Twitter account objects
class Author(object):
    pass

# Define empty array of accounts
authors = []

# Generate variables with Tweet-time found in Tweets within
    tweets.json
with open('tweets.json', encoding='utf-8') as file:
    tweets = json.load(file)
    for i, row in enumerate(tweets):
        # Check each Tweet
        if not tweets[i]['retweet']:

            # Define variables from Tweet
            user_id = str(tweets[i]['user_id'])
            likes_count = tweets[i]['likes_count']
```

```
            retweets_count = tweets[i]['retweets_count'] + 1

            # Look for account in array
            author_index = False
            for index,author in enumerate(authors):
                if authors[index].user_id == user_id:
                    author_index = index

            # If not found, create new Author object with variables
            if author_index is False:
                author = Author()
                author.user_id = user_id
                author.likes = likes_count
                author.retweets = retweets_count
                author.tweets = 1
                authors.append(author)
            else:
                # If author exists, apply variables to the existing object
                authors[author_index].likes = likes_count + authors[author_index].likes
                authors[author_index].retweets = retweets_count + authors[author_index].retweets
                authors[author_index].tweets = authors[author_index].tweets + 1


# Utilize the formula for engagement on each object in account array
data = {}
for author in authors:
    data[author.user_id] = (author.likes + author.retweets * 2) / author.tweets


# Sort array and limit to the top 25
while len(data) > 25:
    key_min = min(data.keys(), key=(lambda k: data[k]))
    data.pop(key_min)


data = {k: v for k, v in sorted(data.items(), key=lambda item: item[1], reverse=True)}

print(data.keys())

authors = list(data.keys())
tweet_count = list(data.values())

# Create figure size
fig = plt.figure(figsize = (25, 15))

# Add values
plt.bar(authors, tweet_count, color ='green',
        width = 0.4)

# Label and show figure
plt.xlabel("Tweet author user ID")
```

```
plt.xticks(rotation=45, fontsize=5)
plt.ylabel("(Like sum + Retweet sum x 2) / Tweet count")
plt.title("Tweet engagement by authors")
plt.show()
```

A.7 tweet_author_frequency.py

As an attempt to visualize activity, each user is ranked by Tweet activity. The script retrieves every Tweet in the data set and gives every user points for each Tweet published. The users are then sorted by most Tweets and the top 25 users are displayed in descending order.

```
import matplotlib.pyplot as plt
from datetime import datetime
from dateutil import tz
import pytz
import json
import itertools
from datetime import date

last_six_weeks = True
election_date = date.fromisoformat('2021-09-13')
data = {}

# Go through each Tweet in data-set
with open('tweets.json', encoding='utf-8') as file:
    tweets = json.load(file)
    for i, row in enumerate(tweets):
        # Check if retweet
        if not tweets[i]['retweet']:

            # If last_six_weeks is true, skip any Tweet not made within six week of election day
            if last_six_weeks:
                split_tweet_time = tweets[i]['created_at'].split()
                tweet_time = date.fromisoformat(split_tweet_time[0])
                duration = election_date - tweet_time
                duration_in_s = duration.total_seconds()
                days = duration.days
                if days > 42:
                    print("Skipped date: " + tweets[i]['created_at'])
                    continue
                else:
                    print("Didn't skip date: " + tweets[i]['created_at'])

            # Define userID as variable
            user_id = str(tweets[i]['user_id'])

            # Add occurence of userID to dict
            if user_id not in data:
                data[user_id] = 1
            else:
                data[user_id] = data[user_id] + 1
```

```
41
42
43 print("Proccessed " + str(len(data)) + " authors.")
44
45 # Sort dict and limit to top 25
46 data = {k: v for k, v in sorted(data.items(), key=lambda item:
       item[1], reverse=True)}
47 data = dict(itertools.islice(data.items(),25))
48
49 authors = list(data.keys())
50 tweet_count = list(data.values())
51
52 print(data.keys())
53
54 # Create figure size
55 fig = plt.figure(figsize = (25, 15))
56
57 # Add values
58 plt.bar(authors, tweet_count, color ='red',
59         width = 0.4)
60
61 # Label and show figure
62 plt.xlabel("Tweet author user ID")
63 plt.xticks(rotation=45, fontsize=6)
64 plt.ylabel("Number of Tweets")
65 plt.title("Tweet frequency from authors")
66 plt.show()
```

A.8 tweet_author_relation.py

As an attempt to visualize relations, each user is displayed on a network figure using nodes, and each mention of another user (node), draws a line between them. The script retrieves every Tweet in the data set, creating a node for each user. Then every Tweet is examined again, drawing lines from Tweet authors to other nodes using the "mention" metadata tag in the Tweet. The final figure is then displayed using "networkx". This was unfortunately never finished.

```
1  import matplotlib.pyplot as plt
2  from datetime import datetime
3  from dateutil import tz
4  import pytz
5  import json
6  import pandas as pd
7  import networkx as nx
8  import scipy
9
10 users = []
11 tweet_list = []
12 G = nx.Graph()
13
14 # Go through each Tweet in data-set
15 with open('tweets.json', encoding='utf-8') as file:
16     tweets = json.load(file)
17     for i, row in enumerate(tweets):
```

```
        # Check if not retweet and stop a sub 1000 for testing purposes
        if not tweets[i]['retweet'] and i < 1000:

            user_id = str(tweets[i]['user_id'])

            if user_id not in users:
                users.append(user_id)

            for mentioned in tweets[i]['mentions']:
                if mentioned['id'] not in users:
                    users.append(mentioned['id'])

            tweet_list.append(tweets[i])


for tweet in tweet_list:
    for mentioned in tweet['mentions']:
        G.add_edge(tweet['user_id'], mentioned['id'], weight=10)


nx.draw_networkx(G, pos=None, arrows=None, with_labels=False)

plt.show()
```

A.9 tweet_botometer_average.py

As an attempt to visualize Botometer scores, every Botometer reply is calculated into averages. The script retrieves every Botometer reply in the data set, adds every category score to the global category variable, while also increasing the increment for said category. Every time a score is above 0.8 it is also printed to the console for manual inspection. The averages are then calculated and printed to the console.

```
import matplotlib.pyplot as plt
from datetime import datetime
from dateutil import tz
import pytz
import json
import os
import itertools
from datetime import date

last_six_weeks = True
election_date = date.fromisoformat('2021-09-13')
last_six_weeks = True
election_date = date.fromisoformat('2021-09-13')
directory = 'botometer'
number = 0
error = 0
astroturf = 0.0
fake_follower = 0.0
financial = 0.0
other = 0.0
overall = 0.0
```

```
self_declared = 0.0
spammer = 0.0

valid_user_ids = []

# If last_six_weeks is true, validate all unique user ID from the data-set who Tweeted within six weeks of the election day
if last_six_weeks:
    with open('tweets.json', encoding='utf-8') as file2:
        tweets = json.load(file2)
        for i, row in enumerate(tweets):
            # Create Datetime object using the syntax Twitter uses, set timezone to UTC
            if not tweets[i]['retweet']:
                split_tweet_time = tweets[i]['created_at'].split()
                tweet_time = date.fromisoformat(split_tweet_time[0])
                duration = election_date - tweet_time
                duration_in_s = duration.total_seconds()
                days = duration.days
                if days > 42:
                    print("Skipped date: " + tweets[i]['created_at'])
                    continue
                else:
                    print('Found Tweet within time frame.')
                    user_id = str(tweets[i]['user_id'])
                    if user_id not in valid_user_ids:
                        valid_user_ids.append(user_id)


# Iterate over files in the Botometer reply directory, checking each file
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # If the path is a file
    if os.path.isfile(f):

        with open(f, encoding='utf-8') as file:
            file_data = json.load(file)

            # Get user ID from the file name
            file_name = file.name.split('.')
            file_name2 = file.name.split('\\')
            file_name3 = file_name2[1].split('.')

            # Validate the user ID if last_six_weeks is true
            if last_six_weeks and file_name3[0] not in valid_user_ids:
                continue

            # If the file contains 'error', ignore it and log the occurence
            if "error" in file_data:
                print("Twitter user ID had error in it.")
                error = error + 1
                continue
```

```
72            else:
73                # Add the different scores from Botometer to
    the averages
74                astroturf = astroturf + file_data['raw_scores'
    ]['universal']['astroturf']
75                fake_follower = fake_follower + file_data['
    raw_scores']['universal']['fake_follower']
76                financial = financial + file_data['raw_scores'
    ]['universal']['financial']
77                other = other + file_data['raw_scores']['
    universal']['other']
78                overall = overall + file_data['raw_scores']['
    universal']['overall']
79                self_declared = self_declared + file_data['
    raw_scores']['universal']['self_declared']
80                spammer = spammer + file_data['raw_scores']['
    universal']['spammer']
81                number = number + 1
82
83                any_takes = False
84
85                # if any category is above 0.8, output it to
    the console
86                if file_data['raw_scores']['universal']['
    astroturf'] > 0.8:
87                    print(str(file_name3[0]) + ": astroturf - "
     + str(file_data['raw_scores']['universal']['astroturf']))
88                    any_takes = True
89
90                if file_data['raw_scores']['universal']['
    fake_follower'] > 0.8:
91                    print(str(file_name3[0]) + ": fake_follower
     - " + str(file_data['raw_scores']['universal']['
    fake_follower']))
92                    any_takes = True
93
94                if file_data['raw_scores']['universal']['
    financial'] > 0.8:
95                    print(str(file_name3[0]) + ": financial - "
     + str(file_data['raw_scores']['universal']['financial']))
96                    any_takes = True
97
98                if file_data['raw_scores']['universal']['other'
    ] > 0.8:
99                    print(str(file_name3[0]) + ": other - " +
    str(file_data['raw_scores']['universal']['other']))
100                   any_takes = True
101
102               if file_data['raw_scores']['universal']['
    overall'] > 0.8:
103                   print(str(file_name3[0]) + ": overall - " +
     str(file_data['raw_scores']['universal']['overall']))
104                   any_takes = True
105
106               if file_data['raw_scores']['universal']['
    self_declared'] > 0.8:
107                   print(str(file_name3[0]) + ": self_declared
     - " + str(file_data['raw_scores']['universal']['
    self_declared']))
```

```
                    any_takes = True

                if file_data['raw_scores']['universal']['spammer'] > 0.8:
                    print(str(file_name3[0]) + ": spammer - " + str(file_data['raw_scores']['universal']['spammer']))
                    any_takes = True

                if any_takes:
                    print('_________________________')


# Print the results
print('Processed ' + str(number) + ' users.')
print(str(error) + ' errors.')
print('_________________________')
print('astroturf: ' + str(format(astroturf / number, '.2f')))
print('fake_follower: ' + str(format(fake_follower / number, '.2f')))
print('financial: ' + str(format(financial / number, '.2f')))
print('other: ' + str(format(other / number, '.2f')))
print('overall: ' + str(format(overall / number, '.2f')))
print('self_declared: ' + str(format(self_declared / number, '.2f')))
print('spammer: ' + str(format(spammer / number, '.2f')))
```

A.10 tweet_domain_frequency.py

As an attempt to visualize sharing activity, each domain is ranked by Tweet occurrence. The script retrieves every Tweet in the data set, and finds every domain from the "urls" metadata tag. Each URL is cleaned using "urlparse" and saved. Each time it occurs it gains one point. The domains are then sorted by most occurrences and the top 25 domains are displayed in descending order.

```
import matplotlib.pyplot as plt
from datetime import datetime
from dateutil import tz
import pytz
import json
from urllib.parse import urlparse

data = {}

# Iterate through each Tweet in data-set
with open('tweets.json', encoding='utf-8') as file:
    tweets = json.load(file)
    for i, row in enumerate(tweets):
        # Check if retweet
        if not tweets[i]['retweet']:
            # Find each URL, clean it, and save it with occurence
            urls = tweets[i]['urls']
            for url in urls:
                domain = urlparse(str(url)).netloc
                if 'www.' in domain:
```

```
                    domain = domain.replace('www.','')
                if domain not in data:
                    data[domain] = 1
                else:
                    data[domain] = data[domain] + 1


# Sort dict and limit to top 25
while len(data) > 25:
    key_min = min(data.keys(), key=(lambda k: data[k]))
    data.pop(key_min)


data = {k: v for k, v in sorted(data.items(), key=lambda item:
    item[1], reverse=True)}

authors = list(data.keys())
tweet_count = list(data.values())

# Create figure size
fig = plt.figure(figsize = (25, 15))

# Add values
plt.bar(authors, tweet_count, color ='yellow',
        width = 0.4)

# Label and show figure
plt.xlabel("Domain (does not include subdomain)")
plt.xticks(rotation=45, fontsize=8)
plt.ylabel("Number of links")
plt.title("Frequency of domains shared in Tweets")
plt.show()
```

A.11 tweet_plagiarism.py

In an attempt to investigate the automated activity, Tweets within certain categories are checked against each other for plagiarism. The script retrieves every Tweet in the data set which matches the current keyword and uses "itertools" to create every unique combination of the Tweets. Each Tweet is then compared using "thefuzz", providing a match ratio between 0 and 100. If the match is above 80, the Tweets are saved with relevant metadata for manual inspection.

```
import os
import json
from thefuzz import fuzz
import itertools
import math
from alive_progress import alive_bar

master_count = 0
count = 0
keywords = ['innvandring', 'abort', 'klima', 'skole', 'velferd'
    , 'rusreform']

```

```
# Loop through each keyword, looking for topics to check for plagiarism within
for keyword in keywords:
    print('Searching for: ' + keyword)
    filtered_tweets = []

    # Loop through every Tweet in data-set and look for Tweets which contains our keyword
    with open('tweets.json', encoding='utf-8') as file:
        tweets = json.load(file)
        for i, row in enumerate(tweets):
            if tweets[i]['retweet']:
                continue

            if keyword not in tweets[i]['tweet'].lower():
                continue

            filtered_tweets.append(tweets[i])


    # Calculate the number of combinations of Tweets to check
    combinations = math.factorial(len(filtered_tweets)) // math.factorial(2) // math.factorial(len(filtered_tweets)-2)
    print('Doing ' + str(combinations) + ' combinations.')

    # Generate progress bar
    with alive_bar(combinations) as bar:
        # Go through every unique combinations of Tweets to check
        for tweet_a, tweet_b in itertools.combinations(filtered_tweets, 2):

            count = count + 1

            # Check the plagiarism ratio
            ratio = fuzz.ratio(tweet_a['tweet'], tweet_b['tweet'])

            # Save if ratio is above 80
            if ratio > 80:
                #print('Found Tweets with ratio: ' + str(ratio))
                with open('compared_tweets/' + keyword + '_' + str(tweet_a['id']) + '_' + str(tweet_b['id']) + '_' + str(ratio) + '.txt', 'w', encoding="utf8") as new_file:
                    new_file.write(tweet_a['username'] + ': ' + tweet_a['tweet'] + "\n" + tweet_b['username'] + ': ' + tweet_b['tweet'])

            bar()


print('Complete.')
```

A.12 tweet_search.py

A some points certain items within the data set had to be inspected, but the text files were too big to be managed through a text editor. This simple Python script searched for certain predetermined variables in the data set and printed them to the console for inspection.

```
import matplotlib.pyplot as plt
from datetime import datetime
from dateutil import tz
import pytz
import json
from urllib.parse import urlparse

# Select Twitter user ID to seacrh for
search_term_id = 0000000

# Go through each Tweet in data-set
with open('tweets.json', encoding='utf-8') as file:
    tweets = json.load(file)
    for i, row in enumerate(tweets):
        # Check if Tweet is retweet
        if not tweets[i]['retweet']:
            if tweets[i]['user_id'] == search_term_id:
                # Print Tweets made by the user ID
                print(tweets[i]['username'] + ": " + tweets[i][
    'tweet'])
                print("_________________________")
                continue
```

A.13 tweet_time_bar.py

As an attempt to visualize user activity, each Tweet is mapped into a bar chart showing each hour in a day. The script retrieves every Tweet in the data set and converts the time from the metadata into the Norwegian time zone. The hour that the Tweet was published is then extracted and one point is given to the corresponding bar in the bar chart. The hours are then displayed in the bar chart, showing how much activity occurs in each hour.

```
import matplotlib.pyplot as plt
from datetime import datetime
from dateutil import tz
from datetime import date
import pytz
import json

hours = ['00:00+', '01:00+', '02:00+', '03:00+', '04:00+', '
    05:00+', '06:00+', '07:00+', '08:00+', '09:00+', '10:00+',
    '11:00+', '12:00+', '13:00+', '14:00+', '15:00+', '16:00+',
     '17:00+', '18:00+', '19:00+', '20:00+', '21:00+', '22:00+'
    , '23:00+']
tweet_hours = [0] * 24
from_zone = tz.gettz('UTC')
to_zone = tz.gettz('Europe/Oslo')
last_six_weeks = True
```

```
13 election_date = date.fromisoformat('2021-09-13')
14
15 # Generate variables with Tweet-time found in Tweets within
       tweets.json
16 with open('tweets.json', encoding='utf-8') as file:
17     tweets = json.load(file)
18     for i, row in enumerate(tweets):
19         if not tweets[i]['retweet']:
20
21             # If enabled, skip Tweets not made within six weeks
        of election day
22             if last_six_weeks:
23                 split_tweet_time = tweets[i]['created_at'].
       split()
24                 tweet_time = date.fromisoformat(
       split_tweet_time[0])
25                 duration = election_date - tweet_time
26                 duration_in_s = duration.total_seconds()
27                 days = duration.days
28                 if days > 42:
29                     print("Skipped date: " + tweets[i]['
       created_at'])
30                     continue
31                 else:
32                     print("Didn't skip date: " + tweets[i]['
       created_at'])
33
34
35             # Create Datetime object using the syntax Twitter
       uses, set timezone to UTC
36             created_at = datetime.strptime(tweets[i]['
       created_at'], "%Y-%m-%d %H:%M:%S %Z")
37             created_at = created_at.replace(tzinfo=from_zone)
38
39             # Convert to GMT+1
40             created_at = created_at.astimezone(to_zone)
41
42             # Add one Tweet to the related hour
43             tweet_hours[int(created_at.hour)] = tweet_hours[int
       (created_at.hour)] + 1
44
45
46 print(tweet_hours)
47
48 # Create figure size
49 fig = plt.figure(figsize = (25, 5))
50
51 # Add values
52 plt.bar(hours, tweet_hours, color ='blue',
53         width = 0.4)
54
55 # Label and show figure
56 plt.xlabel("Hours Tweeted")
57 plt.ylabel("Number of Tweets")
58 plt.title("Hour Tweeted in GMT+1")
59 plt.show()
```

A.14 tweet_to_text.py

This script ended up not being needed, but it was created for a plagiarism approach. It iterated through every Tweet in the data set and saved the Tweet text to a unique file named after the Tweet ID.

```
import json

count = 0

# Go through every Tweet in the data set
with open('tweets.json', encoding='utf-8') as file:
    tweets = json.load(file)
    for i, row in enumerate(tweets):
        # Skip retweets
        if not tweets[i]['retweet']:
            tweet_id = str(tweets[i]['id'])
            tweet_body = tweets[i]['tweet']
            # Save the Tweet to a TXT file
            with open('tweet_text/' + tweet_id + '.txt', 'w')
    as f:
                f.write('tweet_body')
                count = count + 1


print('Finished writing ' + str(count) + ' tweets.')
```