Taming bots on Twitter

Using machine learning to improve Twitter monitoring for political journalism – an explorative study

**By Haluka Maier-Borst**

**Student no. 1662137 Maier-Borst, Haluka**

**Declaration**

**This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.**

****

Signed Haluka Maier-Borst, 3rd of November 2017

**This dissertation is being submitted in partial fulfilment of the requirements for the degree of MSc.**

****

Signed Haluka Maier-Borst, 3rd of November 2017

**This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A Bibliography is appended.**

****

Signed Haluka Maier-Borst, 3rd of November 2017

**I confirm that the electronic copy is identical to the bound copy of the dissertation.**

****

Signed Haluka Maier-Borst, 3rd of November 2017

**I hereby give consent for my dissertation, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.**

****

Signed Haluka Maier-Borst, 3rd of November 2017

# Acknowledgments

*一期一会*

*(Ichi-go ichi-e, lit. translation "one time, one meeting") is a cultural concept in Japan and also translated as "Treasure every encounter”.*

I want to thank each and every lecturer, colleague, fellow student and friend that I have met during this year in Wales and England. I have learned so much from all of you and yet, I understand how little I know and how much I still have to learn. This dissertation would not have been possible without your advice, your encouragement and the pints we had together.

Table of Contents

Introduction 1

Review of current work 4

Using Twitter as a tool for predicting elections 4

The spread of bots on Twitter and how to detect them 7

Methodology 9

Original research part I: problems, fixes and lessons learned 12

Project management 12

Configuration of the Twitter scraper and hosting on AWS 13

Agreement/disagreement analysis and its poor performance 15

Bot detection based on counts and machine learning 20

Hosting on Amazon Web Services 22

Frontend website 23

Story telling and narrative 25

Problems with high number of tweets, possible memory leaks 26

Assessing which criteria of Gayo-Avello checklist are met 27

Original research part II: Does filtering for bots improve the election predictions? 29

The case with the AfD included 29

The case with the AfD excluded 32

Impact of the AfD on Twitter predictions 33

What do the bot-statistics show? 34

The case for a bot-filtering approach 35

Conclusion 35

Simple Twitter election predictions based on mentions do not work 35

Bot filtering can improve predictions but only on a small scale 35

The far-right seems to have been heavily supported by bots 36

Sentiment analysis of text is much more complicated than expected 36

Agile concepts and modular organization of code are helpful 36

Amazon instances and S3 buckets are a lean and efficient hosting solution 36

Discussion 37

Link to the Github Repository with the code: 40

Literature 41

# 

# Introduction

“Caring is sharing”, ”picture or it didn't happen” and “well, that escalated quickly” – phrases emerged from the social web have become part of everyday lingo. But it is not only our use of language that shows how the digital and the analogue world have become more interlinked. We share our vacation pictures with friends via Instagram, we retweet pointed statements on Twitter and we use Facebook to stay connected with friends from all around the world. To state the obvious: social networks have become a regular part of life for most people living in the 21st century. Especially, for the generation between 18 to 24 years the importance of social networks is at unprecedented heights, making it the main source for news for this audience (Newman et al., 2017, p.10).

Unsurprisingly, the political landscape has adjusted to this new media reality. Political campaign managers understand that creating a good meme or engaging with the audience in an AMA (Ask Me Anything) can be as important as well-designed advertisements in TV, radio or print. But this also means that long-held believes about campaigning resources have lost their value. While in the past, a party needed numerous supporters on the ground and a large budget to spend to spread its political message; the situation is different in the digital world.

A few “trolls”, that do not even have to be based in your country, can flood social networks with their messages and derail a political conversation (Borger, 2017). Furthermore, with the rise of automation, it is even possible for a single person to spread a message cheaply using a multitude of automatically steered accounts, so called “bots” (Wakabayashi and Shane, 2017).

At the same time, we are also experiencing a crisis in the business of predictions. Within a year, we saw how the United Kingdom unexpectedly voted in favor of leaving the European Union, we were surprised by an election that made Donald Trump the unlikely president of the United States of America and just a few months ago we also witnessed how Theresa May, who was set for winning a General Election by a landslide, ended up with a hung-parliament.

Given these new and challenging disruptions that are changing how politics work, I have decided to dedicate my master’s thesis to an alternative way of monitoring political conversations on the web: polling based on tweets. The approach itself has been around for some time. Tumasjan et al. (2010) started the idea of Twitter as a proxy for election outcomes by predicting the results of the German election based on tweets. More recently, Celli et al. (2016) used Twitter to predict the vote for Brexit in the EU referendum.

However, none of the examples that I have found has tried to adapt to the recent challenge of bots and trolls on social media. Therefore, the aim of this thesis is to create a Twitter-poll for the German General Election on 24th of September 2017 that tracks the political debate by gathering tweets and tries to filter out bots. The filtering technique will be based on machine learning, a technology that has only been used sporadically in journalism so far (Aldhous, 2017 and Ceccon, 2015).

In this dissertation, I will firstly review the existing work about using Twitter as a tool for election prediction, the increase of bot generated content and different techniques used to detect such content. After this brief introduction, I will explain my own Twitter poll tracker “hashtagswahl” that incorporates different bot filters. Finally, I will evaluate the performance of my tool and discuss future pathways for similar approaches.

# Review of current work

Single tweets have become news events themselves as Donald Trump’s tweets show frequently. However, with 500 millions tweets a day (Twitter Usage Statistics, 2017), Twitter is more than just a message board for unorthodox presidents. It is a constantly updating stream of information that reflects what a large group of people talk about. Hence, Twitter has been used for numerous analytic purposes such as predicting the outbreak of swine flu (Lampos&Cristianini, 2010), the changes in stock market (Bollen, Mao & Zeng, 2010) or the outcome of elections (Tumasjan et al., 2010).

## Using Twitter as a tool for predicting elections

The first take on using Twitter for predicting election outcomes is to my knowledge the one by Tumasjan et al. (2010), who used it for the German elections in 2009 and claimed “the mere number of tweets mentioning a political party can be considered a plausible reflection of the vote share and its predictive power even comes close to traditional election polls.” (Tumasjan et al., 2010, p.183)

Tumasjan’s study was criticized for being arbitrary regarding the parties included and the time frame chosen for the retrieval of the tweets (Jungherr et al., 2012). But while these counter arguments are partly specific to the German election system and were countered again in detail by Tumasjan et al. (2012) later, Gayo-Avello (2012) pointed out bigger concerns about the general idea of using Twitter as a tool for polling. In fact, his criticism became a checklist for future research. The points that he brought up are briefly summed up here:

* To be a true prediction, it has to be made public in advance of the election.
* Chance is not a valid baseline because incumbency tends to play a major role in most elections and hence, has to be addressed.
* There has to be a definition of what a “vote” is (tweets, users) and also what the result of the “Twitter polling” has to be compared against (polling results, election results, parties’ vote share or ranking order).
* If sentiment analysis is used, it has to be tested, validated and not simply be used as a black box.
* Demographics have to be addressed as Twitter is not representative
* It has to take into account that not everyone tweets about politics and not every message is trustworthy.
* The chosen method should be checked also against past elections.

(Gayo-Avello, 2012, pp. 92)

Partly as a response to this criticism and partly due to other motivations, researchers have made adjustments in several Twitter based prediction models since then.

Choy et al. (2011) adjusted their analysis of the Singaporean presidential election by using a specific lexicon that classified certain words as positive or negative. This allowed the application to estimate the sentiment of tweets and to look at whether a candidate’s name was mentioned in a positive or negative context. The idea was to cover specific cases such as scandals when the name of a politician might be mentioned more frequently on Twitter but this does not necessarily translate to a rise in popularity. Also, Choy et al. weighted their Twitter result in respect to population data from census to adjust for the fact that Twitter users are younger than the average. Still, their prediction was only able to name the two leading candidates out of four but predicted the wrong candidate to become president. Furthermore, the margin of error between result and prediction was for three out of the four candidates larger than five percent.

Sang & Bos (2012) tried several approaches for the Dutch Senate election and came closest to the actual result when they calibrated their method with data from actual polls. This led to a relatively good result that was indeed quite comparable to traditional polls. Still, one might argue that this closeness to poll predictions is not surprising after all, given that the Twitter measures of their own method were calibrated and heavily adjusted according to polls in the beginning. Also, this is of course a possibly problematic method in a time, when polls have made wrong predictions as noted earlier.

Ceron et al. (2013) used an approach similar to Choy et al. that was based on a manually coded corpus of words to assess sentiment in tweets. However, their approach fared far better than the one by Choy et al. Ceron et al. could show that their method was reflecting accurately the popularity of Italian politicians as measured in polls but also that they could correctly forecast the outcome of the French presidential ballot with a small margin of error.

Especially in the context of the U.S. and UK election processes, numerous approaches have been tested as well. Shi et al.(2012) created state-based predictions for the Republican primaries by filtering out promotional accounts and estimating the geo location of the Twitter user through a self-developed algorithm. The results in terms of prediction of the winner were all correct. However, some of the predicted outcomes were off by a margin of seven per cent points, which would be concerning in tighter races.

In contrast, Wang et al.(2012) focused on the constant inaccuracy around sentiment analysis. They created an interface that randomly picked out a small sample of tweets from the entire collection that they had stored and asked users to manually rate these as positive or negative. This created a hybrid model of sentiment analysis, partly based on algorithms and partly based on user-ratings.

By using this user feedback, the classification of a small number of specific tweets was corrected, but more importantly a machine learning algorithm was trained with this data set, which improved the accuracy of the algorithm to 59 per cent. This exceeded the baseline of categorizing all tweets as negative, which was the most common sentiment in the sample dataset (56 per cent). However, Wang only used their system as a tracker of online conversation but never made predictions based on this, which is why it is difficult to assess its value for election forecasts.

For recent UK voting processes, Twitter has not been a good predictor in most of the study cases. Even relatively sophisticated approaches including geo-location of tweets, sentiment analysis, and topic modeling have not been accurate in predicting the outcome (Burckhard et al. 2016, Burnap et al. 2015).

But there is one exception. Celli et al.(2016) used two forms of sentiment or agreement/disagreement analysis for their EU referendum prediction. One was based on a lexicon like Choy et al.’s approach, but the other used a stylometric method that included machine learning. This meant that for each tweet, its level of agreement or disagreement was calculated by looking at features such as length of tweet, average length of words etc.

Both of these approaches were extremely accurate when it came to predicting the win for the Leave campaign in the EU referendum. In fact, they even outperformed most regular polls with error margins of 0.11 to 0.64 per cent points.

This taken together with the outcomes in the U.S. election process might hint to the fact that in a first-pass-post system, Twitter as an election predictor struggles in the same way as traditional pollsters do. In such systems, regional effects are amplified and furthermore, winning by small margins in certain constituencies or states and losing by a large margin in others can lead to a great imbalance between share of seats and shae of votes.

In the case of the last two UK General Election it has to be noted that Labour and the Conservatives, both won a bigger share of the vote than in the previous election. Still in the 2015 election Labour lost 26 seats (Election, 2015) and in the 2017 snap election the Conservatives lost 13 seats (Election, 2017) despite gaining in vote share. In the 2016 U.S. elections, the discrepancies was even bigger as we witnessed a candidate becoming president who clearly lost in terms of the popular vote (Presidential Results, 2016).

## The spread of bots on Twitter and how to detect them

However, there is another issue with Twitter as a polling tool that has already been addressed in Gayo-Avello’s critical analysis. Not every message on Twitter is trustworthy and “Twitter is plagued with rumors, propaganda, and misleading information that are processed as valid political opinion” (Gayo-Avello, 2012, p. 93). Gayo-Avello concludes that such false information has to be excluded from analysis per se because of its falsity. This statement by itself is problematic.

Propaganda, misconceptions and prejudices have always been part of democracies and also influenced election outcomes and political decisions. Classic examples are totalitarian regimes such as under Lenin and Hitler that used propaganda for their purposes (Brendon, 2017). But also in modern day’s history, we have seen false claims such as the one about weapons of mass destruction in Iraq leading to a war.

More importantly, one might argue that recent political campaigns such as the one by Donald Trump or the one in favor of leaving the European Union were largely based on lies (Konnikova et al., 2017 & Kirk, 2017). Excluding them from an analysis would be an idealistic but wrong way of looking at political conversations as lies and misconceptions are crucial to understand debates.

However, what is true is that understanding political discussions on social media can become impossible, if propaganda drowns other arguments due to its sheer volume and noise.

This is especially true in the case of bots that spread propaganda messages in vast quantity, which can create the impression that a certain information “regardless of its accuracy, is highly popular and endorsed by many”(Ferrara, 2015, p.2). In the context of the EU-referendum, it was estimated that a third of all tweets that included non-neutral hashtags was produced by accounts that use heavy automation (Howard et al., 2016).

Furthermore, by using a whole network of bot accounts rather than a couple of heavily tweeting accounts the phenomenon of “astroturfing” can be created (Ratkiewicz et al., 2011, p.298). This means that because of the large numbers of likes and retweets, it looks like a candidate has gained widespread support through a grass root movement when in reality his support comes from a network of automated accounts.

The response to this development was that several approaches for detecting bots have been suggested. Lee et al. (2011) created 60 bots themselves and used a “honeypot” approach, in which they defined bot accounts as accounts that followed their accounts that tweeted nonsense. However, it has been argued that such an approach would not be as effective as back then, as social bots have become more sophisticated in their behavior (Ferrara, 2015). This is inline with recent observations that defined twelve more elaborated characteristics that have to be taken into account in order to spot a bot (DFRLab, 2017).

Others have used algorithms that look at characteristics of the relationships between different accounts to examine whether an account is a bot or not. Still, it was acknowledged that this approach leads to a high false-positive and negative rate that can only be adjusted for by using a human in the loop (Cao et al. 2012).

This, however, implies that humans are per se good at detecting bots, which is an assumption that is partly questioned by Wang et al. (2013), who reported that accuracy for human examiners rapidly dropped over time.

Recently, some researchers (Neudert el al., 2017) used the mere number of tweets per day from an account including a certain group of hashtags as the defining criteria to determine whether an account is a bot nor not. This means that it is assumed that any account that tweets more than 50 times, for example, with the hashtag “#Labour” within 24 hours is a bot.

It is argued that this method “has the advantage of capturing the content most likely to be about this important political issue” (Howard et al. ,2016, pp.2). The other upside of this method is that it is easily understandable for non-experts and hence can be used in journalistic stories about the influence of social bots on elections (Blood, 2017).

Lastly, BotOrNot, developed by researchers from Indiana University (Davis et al., 2016) looks at a wide range of behavioral features of accounts to determine whether they are bots or not. These include temporal patterns, friend lists, content and sentiment of the content that distinguishes bots accounts from regular accounts. The accuracy yielded with this approach of distinction was at 95 per cent. In its core, this approach is a machine learning algorithm that is trained on the afore-mentioned behavioral features in order to detect bots.

# Methodology

The main idea for my application was to connect these two fields of web-based social science, the Twitter based predictions and the bot detection, in order to monitor the political discussion on Twitter before the German elections and to create a prediction model.

The distinction between bot and non-bot tweets was done with two methods. One was based on counts, so any account tweeting more than 24 times about the German elections within a day was counted as a bot. The other approach was based on a machine learning algorithm that flags accounts as bots by looking at stylometric features of their tweets.

For this purpose, I created an application that consists of four core modules. The distribution of code into several packages was done as it allowed easier error handling but also meant that tasks could be distributed onto several servers and hence, one part of the code failing did not mean that the entire application failed. The four modules that were triggered every 15 minutes were:

1. A Twitter scraper that connects with the Twitter Streaming API and gathers all tweets that either mention the name of a party or the name of candidates of a party that was projected to make it into the next German parliament. The scraper creates a JSON-file, saves it and then continues the scraping.
2. A JSON to CSV converter that creates two CSVs out of the JSON files. One CSV includes all tweets from the last 24 hours and one includes all tweets that were posted on the day before, so between 48 to 24 hours ago.
3. A sample creator that goes through the latter CSV (tweets from 48 to 24 hours ago) and:
4. creates a stylometric profile for each tweet. This means it counts, for example, the number of dots, commas, exclamation marks but also the number of characters, words, unique words, lower and upper case words.
5. creates two samples from the collection of tweets: One with tweets from accounts that only tweeted once within this timeframe about the monitored topics and saves it as a sample for non-bot tweets. And one with tweets from accounts that tweeted more than 24 times about these topics within this timeframe, the sample for bot-tweets.
6. uses these two samples to train a machine learning algorithm called “random forest classifier”, then calculate the accuracy of this algorithm and lastly saves the classifier model.
7. creates a list of all account-ids that tweeted 24 times or more about these topics within this timeframe.
8. An analysis creator that goes through the CSV for the tweets of the last 24 hours and:
9. creates a stylometric profile for each tweet. This means it counts, for example, the number of dots, commas, exclamation marks but also the number of characters, words, unique words, lower and upper case words.
10. uses the trained and saved machine learning classifier model to asses for each tweet, based on its stylometric profile, whether it was created by a bot or not.
11. uses the saved list of all account-ids that tweeted 24 times or more within the timeframe 24 hours to 48 hours ago and updates it with account-ids that tweeted 24 times or more within the last 24 hours. Based on this count-based list it classifies whether a tweet was created by a bot or not.
12. counts how often each party is mentioned in tweets and creates a new csv file that shows how often a party is mentioned in tweets that:

-were determined as “bot-tweets” based on how often an account has tweeted

-were determined as “non-bot-tweets” based on how often an account has tweeted

-were determined as “bot-tweets” based on the random forest classifier machine learning algorithm

-were determined as “non-bot-tweets” based on the random forest classifier machine learning algorithm.

All of these modules were hosted on Amazon Web Service (AWS) EC2 server instances with the scraper being on one instance and the other three modules hosted on a separate server instance. A cronjob, a small line of code, made sure that the three modules on the latter instance were executed one after another in the right order. The created data files were all stored using Amazon’s S3 buckets, from which a frontend with simple graphics was run to constantly show the results of the analysis.

# Original research part I: problems, fixes and lessons learned

## Project management

As I was the only person working on this project, it did not make a lot of sense to implement a large set of AGILE methods as they are predominantly centered around the task of organizing group work. However, I tried to implement some basic principles of project planning that have proven valuable during my studies.

1. **Minimum viable product and other product stages**

Working in larger projects can easily lead to getting lost in details and ending up with a product that is not delivering the needed service. To avoid this, I clearly defined a minimum viable product and different evolutionary stages that build on this minimum version. These were:

* 1. *First prototype:* The first prototype should gather tweets that contain one or multiple hashtags that were pre-defined as referencing to the elections. The hashtags should state whether tweets are in favor or against one of the main candidates. Furthermore, it should be able to filter out tweets as bot-like if they come from accounts that post more than 24 times per day about these tracked hashtags.
  2. *Intermediate model:* This prototype should use the lexicon and the stylometric-based approach as laid out by Celli et al.(2016) to rate the sentiment of the tweets. The filter mechanism for the bot-like behavior is the same as in a).
  3. *Advanced model:* Instead of simply using a predefined lexicon, this model of the web-app should be based on a specific data set generated by human examiners that rated tweets being agreeing or disagreeing with a certain party or candidate. This dataset would then be used to train a simple form of machine learning that would then deliver sentiment analysis on each tweet.
  4. *Final stage:* The web-app constantly updates its bot-detection rules and its classification lexicon by asking visitors of the app to tell whether a tweet is bot-like or not and whether the tweets are in favor of against a candiate, before granting access to the website itself. Through this method, the web-app should be able to analyze the elections in a mixed model of crowdsourcing and machine learning based classification.

1. **Splitting the project up into modules**

As explained earlier, splitting up the task into different modules allowed easier bug-fixing and distribution of modules across servers. Furthermore, this also allowed to put together an application that was on different stages of the planned prototype scheme, so for instance, more sophisticated in its bot-detection but less so in its sentiment analysis.

1. **Scheduling of sprints**

Working towards one big deadline (in my case the start of my internship at the Financial Times at least for the coding part) is often too abstract to work. Hence, I portioned the amount of work I had to do by scheduling two-week sprints that defined what had to be done within the next 14 days. Again, the portioning of the code into several modules helped as it allowed to clearly define on which level each module had to perform by a given date.

## Configuration of the Twitter scraper and hosting on AWS

Given the fact that Theresa May called a UK snap election on April 2017, the idea was to use the General Election on 8th of June as an early test bed for the planned application. However, it turned out that this idea was overly ambitious.

An initial problem was that the chosen way of scraping the Twitter stream proved unreliable for large amounts of tweets retrieved over several hours. The core script of Dolinar (2015) was not ideal in cases where the internet connection was poor or intermittent, as it would not keep any of the data retrieved so far. Also, with bigger data sizes the programmed modules would run into the problem that in the process of storing, the RAM capacity of my computer would be exceeded and force the programme to crash without storing any data.

Unfortunately, due to several other circumstances, I did not come up with a workaround for these problems in time for the UK’s General Election, though in hindsight the solution was relatively simple.

Firstly, I decided to host the entire scraper on an AWS instance, so that the probability of a loss of internet connection was minimized compared to using my own computer or another device such a Raspberry Pi. I also then modified the scraper so that it would save tweets after running for 14.5 minutes, and then used a cronjob to restart the scraper every 15 minutes.

This meant that the amount of data saved in one process would be kept small and the RAM was not overloaded. It also made sure that ideally even in the case of malfunctioning at one particular moment in time, the scraper would only lose data collected over a timespan of 15 minutes rather than several hours. Furthermore, this approach turned out to be helpful, as it allowed me to constantly use the most recent data, instead of having to wait for an hour or more for the scraper to finish. On the other hand, the downside of this approach is of course that for every hour of data collection, for a timespan of an accumulated two minutes no data was collected.

After creating a pipeline for the retrieval of tweets, I started configuring the scraper only to monitor the following mentions or hashtags:

For the election in general: *'#btw17'*

For the CDU: *'#Merkel', '@CDU'*

For the SPD: *'#Schulz','@spdde','*

I made this choice in the beginning, as I wanted to reduce the Twitter polling to a binary vote between the candidates of the two major parties. I did this because I wanted to be able to compare my application’s results to results in the UK and the U.S. where a largely two party system is in place. Also, I assumed that the results would be more telling in case of the two main candidates and also closer to the research of Celli et al. that looked at a binary vote with the EU referendum.

However, I reverted from this idea at a later stage after it became clear from traditional polls that the performance of smaller parties might be the interesting storyline for the German elections and especially the results for the AfD.

Also, given the debate about the AfD’s possible use of social bots (Rosenbach, 2016), it seemed relevant to examine how this party was using automated content rather than just focusing on the two main parties. I then added also the following mentions and hashtags to my Twitter scraper:

For the AFD: *#AFD','@AfD\_Bund','#Gauland', '@Alice\_Weidel',*

For the FDP: *'#FDP', '@c\_lindner', '@fdp','*

For the Greens/Grüne: *#Gruene','@Die\_Gruenen','@cem\_oezdemir', '@GoeringEckardt',*

For the Left/Linke: '*#Linke', '@DietmarBartsch', '@SWagenknecht', '@dieLinke'*

This includes the name of each party and mentions of each party’s leading candidates. It should be noted that for the two major parties, the scraper was only set up to look for the name of the candidates being mentioned as hashtags, which is different to the approach for the smaller parties. This was done because Angela Merkel from the CDU does not have a Twitter account and hence I assumed, it would be unfair to include tweets that either mention her challenger Martin Schulz’ account or his name in a hashtag.

Yet, in hindsight I think that this might have been a wrong decision at least at the stage of data collection. It would have been better to collect Martin Schulz’ account mentions and later filter in the analysis process, if necessary. I also regret not having included the party names of CDU and SPD as hashtags, as I did for the smaller parties.

## Agreement/disagreement analysis and its poor performance

Building on the work of Celli et al.(2016), the idea was to create a module that would be able to distinguish between agreement and disagreement in tweets. The two approaches tested for this purpose were:

* lexicon-based agreement/disagreement analysis
* machine learning driven agreement/disagreement analysis, based on stylometric features

For the lexicon-based approach, each word of a tweet is checked against a lexicon in which words are rated as positive or negative. Using this approach, each tweet is then classified as agreeing, if it includes more positive words, or disagreeing, if it includes more negative words. Also, it is rated, how much the tweet agrees or disagrees. So for instance a tweet consisting of five words out of which two are rated positive would get a score of +0.4, whereas a tweet with two negative words out of eight would be rated as -0.25. The lexicon used was the OpeNER-lexicon for the Italian language in the test runs. However, for the German election, the plan was to use the German OpeNER-lexicon.

The stylometric analysis looks for each tweet at the occurrence of:

commas, exclamation marks, words, uppercase words, lower case words, special characters except of number and letter, numbers, characters, unique words, quotation marks, opening brackets, closing bracket, hashtags, three dots after another, links

* dotcount=cell.count(".")
* commacount=cell.count(",")
* exclamationcount=cell.count("!")
* qmarkcount=cell.count("?")
* apicescount=cell.count("'")
* quotescount=cell.count('"')
* openparcount=cell.count('(')
* closeparcount=cell.count(')')
* hashtagcount=cell.count('#')

*Fig 1. Code snippet that counts certain characters for stylometric analysis*

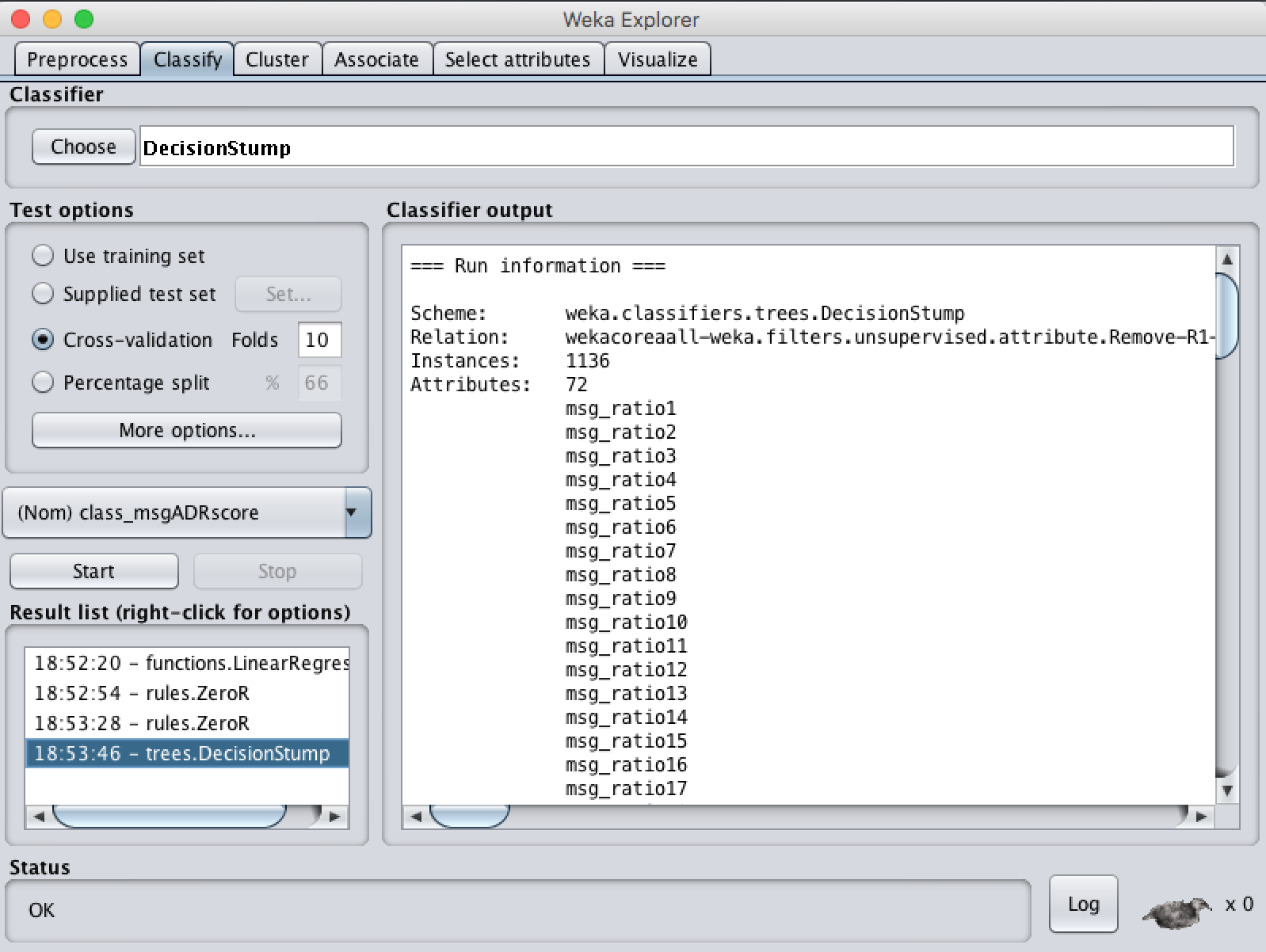
The idea is that tweets with disagreement have a specific stylometric fingerprint that differs from tweets with agreement because they are possibly longer, use more exclamation marks and more words in uppercase etc. By quantifying these features and creating for each tweet a vector consisting of numbers representing these parts of the fingerprint, the idea is that the machine learning algorithm can detect this pattern and then classify each tweet accordingly.

To evaluate both approaches, I used the method described by Celli et al. The test data set was the Italian CorEA dataset that consists of online comments that were manually classified as agreeing or disagreeing.

By running both programme modules against this dataset, I checked how often the code classifies the online comments in the same way as the human annotators did. Unfortunately, it turned out that I could not replicate the results of Celli et al. and yielded much lower accuracies.

The lexicon based approach was slow in its processing even after using a binary tree to restructure the lexicon in order to make the runtime more efficient. I decided hence, to only classify the first 472 entries and only achieved an accuracy of 50.7 per cent over this sample. Experiments of taking only the first ten words of each entry yielded similar results.

The stylometric approach was tested using the machine learning algorithm library of WEKA (Frank, Hall & Witten, 2016) and its graphic user interface (GUI). I decided to try out the different algorithms of the library by using a ten-fold cross validation and then compare the accuracy. This meant that I picked one of the algorithms and would check how often it would correctly classify one of the 1929 comments from the dataset as agreeing or disagreeing based on the stylometric profile that my script had earlier created for each comment.



*Fig.2 Interface of the WEKA GUI*

Ten-fold cross validation is defined as a process in which the data is divided into ten subsets and one of the subsets is held out for later testing the accuracy. The remaining nine subsets are used to train the algorithm and then the algorithm is tested against the hold-out subset. This procedure is repeated ten times with alternating subsets.

The best results for this stylometric approach were yielded with a random forest algorithm; however the accuracy was still relatively low at 56.4 per cent.

Looking for improvement, I asked Fabio Celli for advice. I wanted more information on why the lexicon-based approach was scoring so low and which stylometric features he looked at specifically, as they were not specified in detail in the paper. Finally, I also asked him for his processed stylometric data for each entry in the CorEA dataset.

Mr. Celli explained that his experience with lexicon-based approaches suggests that it is error-prone and it might be best to dismiss this approach. Also, he told me that he would not expect my code for the stylometric approach to yield a big difference in accuracy, even though I was not looking for complex features such as n-grams[[1]](#footnote-1).

Celli’s assurance, that my approach was correct, while helpful was also troubling because this meant that I did not know where the difference came from. However, this was resolved when I received Celli’s processed dataset that had a stylometric profile for each comment in the CorEa dataset.

After I ran Celli’s dataset within the GUI of WEKA, I indeed yielded a higher accuracy. Trying to isolate the reason for this difference, I used a decision stump algorithm, which means that the algorithm picks one attribute that on its own is the best one to predict the outcome variable.

This singular attribute that yielded on its own an accuracy of 63.5 per cent was named “msg\_sentiment”. After several e-mails between Mr. Celli and myself, it turned out that this score was based on sentiment analysis using the Python library Word2Vec.

* msg\_sentiment <= 0.5
* 0   0.5 1
* 0.6609195402298851  0.10632183908045977 0.23275862068965517
* msg\_sentiment > 0.5
* 0   0.5 1
* 0.2853982300884956  0.09955752212389381 0.6150442477876106
* msg\_sentiment is missing
* 0   0.5 1
* 0.44875 0.1025  0.44875
* Time taken to build model: 0.07 seconds
* === Stratified cross-validation ===
* === Summary ===
* Correctly Classified Instances         508               63.5    %
* Incorrectly Classified Instances       292               36.5    %
* Kappa statistic                          0.3379
* Mean absolute error                      0.3444
* Root mean squared error                  0.4152
* Relative absolute error                 88.0018 %
* Root relative squared error             93.894  %
* Total Number of Instances              800
* Ignored Class Unknown Instances                336

*Fig.3 Results of the Decision Stump with Celli’s dataset*

This was a frustrating revelation as Mr. Celli’s paper said in its description that it did not use sentiment analysis and yielded more accurate results. Following remarks from my side that this is rather confusing for readers of his paper, Mr. Celli replied in an e-mail that by mistake he had sent me a preliminary dataset of his Brexit study. In the actual study, he claimed to not have used sentiment analysis in his code because the process would have been too time consuming for a live application. In response, I asked him for the processed dataset that took into account only the features that he actually looked at during the EU referendum.

Unfortunately, at time of this writing, Mr. Celli has not responded to my repeated requests for his final dataset.

Also, I tried to implement the library Word2Vec but given the time constraints and Celli’s remark that using it would not allow to process data quickly enough, I decided to step back from implementing Word2Vec in this context.

After these poor results in agreement/disagreement analysis, I decided to consult an expert in machine learning. I talked with Daniel Kirsch, a data scientist who has been working together with the Berliner Morgenpost and held an hour-long workshop on machine learning at the German journalism conference Netzwerk Recherche. He said that in his opinion the approaches of Celli might be indeed difficult. Kirsch said:

“Language is much more difficult than we often realize and there is a lot of complicated interaction between words in sentences. So, it is not a surprise for me that the lexicon-based approach performed poorly.”

Furthermore, he also said that he had doubts about how effective the stylometric approach could be. Based on these troubling results, the unanswered questions around Celli’s approach and the doubts by another expert on the feasibility of my two approaches, I decided to scrap the use of agreement/disagreement analysis altogether and I made the choice to count only the mentions of parties regardless of whether they were positive or negative.

## Bot detection based on counts and machine learning

The programming of the bot detection was much easier. Setting up the count-based filter was relatively simple, with the only caveat being that the threshold for defining an account as a bot-account had to be lowered from 50 to 24 tweets per day. This was done, as I could only find one account on German Twitter that tweeted more than 50 times about the monitored hashtags at the end of July, when I started programming my bot filter. Therefore, I deviated from the approach of the Oxford Internet Institute in this regard (Neudert, Kollanyi & Howard, 2017a).

Furthermore, the machine learning and stylometrics method, that was not helpful in terms of agreement/disagreement analysis, proved to be fruitful for the distinction between tweets generated by bots and tweets generated by humans. For the implementation of this method the stylometric profile of each tweet was used and a sample dataset was created that consisted of:

* 1000 tweets from accounts that only tweeted once within 24 hours about the tracked hashtags defined as “non-bot tweets”
* 1000 tweets from accounts that tweeted at least 24 times within 24 hours about the tracked hashtags defined as “bot tweets”
* #assembling from the collected dataframes two samples of size 1000
* normaldf=(normaldf.sample(1000, replace=True))
* botdf=(botdf.sample(1000, replace=True))
* (...)
* #creating a training dataset for the algorithm by concating
* #the bot-dataframe and the normal-dataframe
* test=pd.concat([normaldf,botdf], ignore\_index=True)

*Fig 4. Code snippet for creating test dataset*

The reason for this sampling procedure was simple. In the case of a sample with either a strong majority of bot-tweets or normal-tweets, the algorithm can become “lazy” and yield a high accuracy by simply classifying all tweets into one category. However, this, of course, is not a helpful classification method. Hence, I picked a 50/50 sample that also made it easier to compare the accuracy of the algorithm with the base-line being 50 per cent, a coin toss.

Before testing different algorithms, I considered the use case. As I only wanted detection, but did not necessarily need to understand which rationale the algorithm was using for the distinction between bots and not-bots, I deemed that it was also possible to test more complex algorithms such as Support Vector Machine (SVM). SVM has shown quite accurate results with different tasks in the past and was recommended to me by Kirsch. However, SVMs are hard to interpret because of their mathematical complexity and thus not ideal if the machine learning algorithm needs to create a model that is also easily interpretable.

But as this was not the case, I opened up my search for the right machine learning method also to this class of algorithms. Also, I tried out a one-class classification approach, which basically means that instead of creating a sample consisting of two classes, the algorithm only gets a sample consisting of one class, so in my case the bot-tweets. The algorithm tries to define what the characteristics are within this sample and then tries to pick out entities that fit these characteristics from a data set. This meant it tried to pick out bot-tweets out from a sample of bot and not-bot-tweets.

Yet, after testing the different machine learning approaches on different datasets, the best solution was WEKA’s J48, a decision tree algorithm based on a two-class sample. The accuracy of this algorithm was ranging from 64.5 per cent to 70.8 per cent for a dataset of tweets about UK politics that was gathered from 24th to 30th of June (after the snap election).

However, as there is not an exact equivalent for the J48 in python’s machine learning library Scikit-Learn, I tried out different decision trees within Scikit-Learn. The random forest classifier yielded the best results and achieved an accuracy of 85 per cent in first tests with datasets of German tweets.

Furthermore, as explained above I changed the tracked hashtags from a two party focus to the entire spectrum of parties that were likely to make it into parliament. After this change, the accuracy of the classifier dropped during first tests to 78 per cent. However, the accuracy improved later on and went above 80 per cent for some periods. Additionally, I addressed this problem by implementing a threshold that would use the count-based method for distinguishing between bots and normal tweets, if the accuracy dropped below 80 per cent (more on this see the chapter on frontend).

## Hosting on Amazon Web Services

After checking that the separate modules of the application worked, I decided to host the modules on AWS. This process was all done during my time at the Financial Times, which meant that time for working on the application was limited to evenings and weekends. However, it allowed me to consult experienced colleagues about how they would build the application.

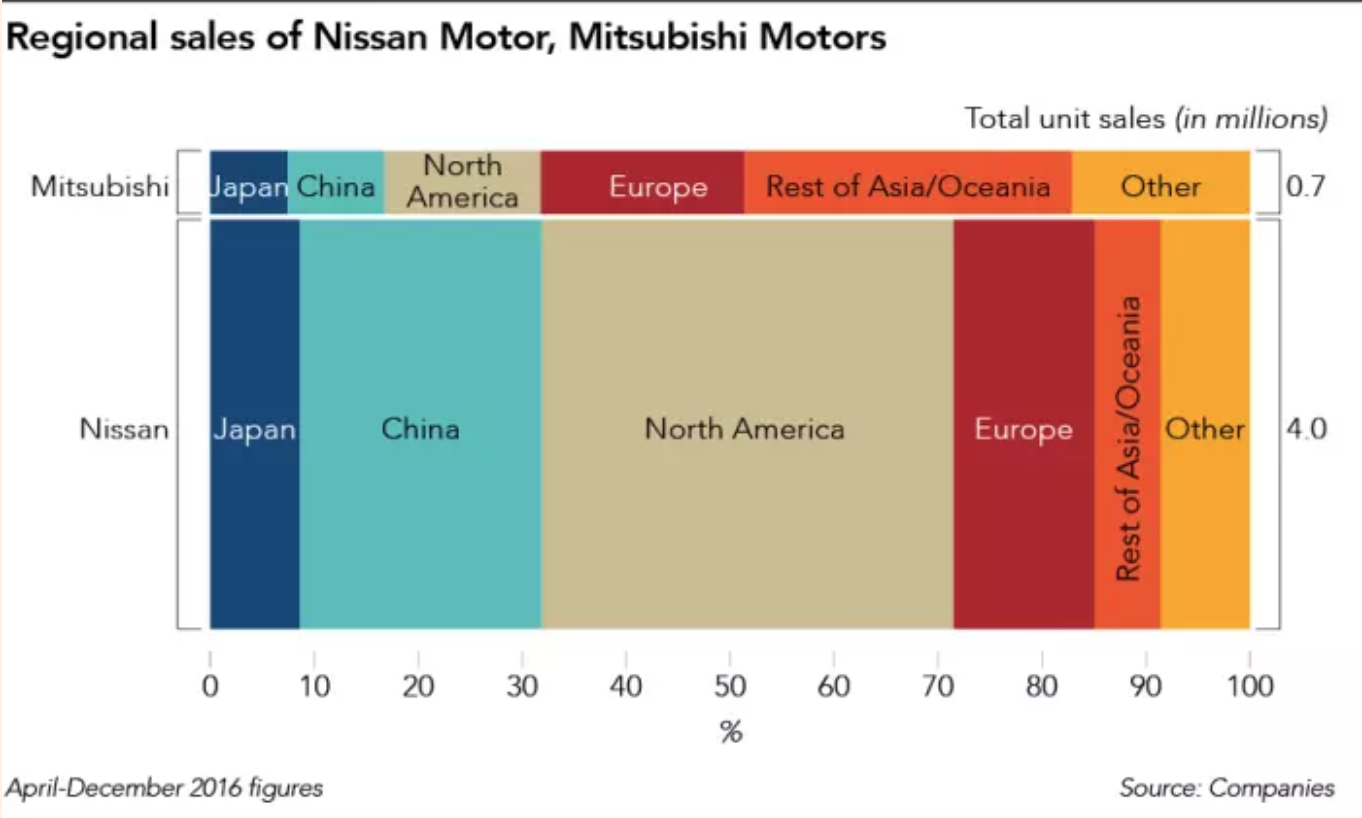
The Twitter scraper was hosted on one server instance and programmed to save the retrieved tweets constantly in an S3 bucket. This was a suggestion of my colleague Ændrew Rininsland, developer at the Financial Times. He also had the idea to use Amazon’s Lambda service for hosting the other modules. The advantage of Lambda is that users only pay for execution time and hence hosting a website with a backend that only does calculation for a certain amount of time becomes much cheaper.

However, the problem was that my code required the Python libraries Pandas and Scikit-Learn. Both, under the hood, are based on the programming languages C and Fortran (Brown, 2017), which meant that the libraries had to be precompiled. This eventually resulted in a package size that was too big for Lambda and hence, this service could not be used.

Thus, the three other modules were hosted on another regular web instance. A cronjob was implemented, which is a simple line of code that makes sure that one module is executed after another. The final result, that was going to be displayed on the website, was saved in a csv and pushed into another S3 bucket from which the frontend was run.

## Frontend website

The website was hosted under the domain “hashtagswahl.de” so that users could constantly look at the outcome of the Twitter polls. The HTML and CSS framework for the website was written by my German colleague and friend Michel Penke who also helped me with the header picture. Into this HTML template, two simple d3.js bar-charts were added that I had built for an earlier project and modified slightly for the new purpose.



*Fig. 5 Example of Marimekko chart from FT’s chart doctor column*

Alan Smith, data visualisation editor at the Financial Times, suggested to use a Marimekko chart to allow the users to compare absolute numbers of tweets against each other but also the relative proportions between parties. However, given the time constraints and also doubts about whether users would understand this form of visualization right away, I decided to not to use this suggestion and rather stay with the simple bar charts.

Additionally, the frontend also included a feature that decided whether the results from the count-based method or machine learning based method were displayed. This was done by including a threshold test in the d3.js script. If the accuracy of the machine learning based method against the sample dataset was higher than 80 per cent, this method was used. Otherwise, the d3.js script would use the results from the count-based method. The rationale behind this threshold was that I wanted to avoid reporting incorrect numbers based on an algorithm gone wild.

In the past, machine learning based projects in journalism have been used more for the narrative side of reporting (Narrative Science, n.d.). The task has been to create news snippets based on clearly structured data such as in the case of narrative science. This means that in a worst-case scenario writing about stock trading or sports performance, for example, could sound a bit clunky, but still the analysis based on market figures or sports’ statistics would most probably come across correctly.

It is only more recently that machine learning has also been used for the analytical and investigative side of the news industry. Examples for this are The Times’ decision trees for the UK elections (Ceccon, 2015) or Buzzfeed’s investigation into spyplanes (Aldhous, 2017). However, in these cases the analysis is not live on the web as the algorithm carries it out. Either editors look at the model of the algorithm on election night before putting it out or the machine learning is only used as a filter tool to reduce the amount of data a reporter has to check. So, there is some form of human in the loop-check.

However, in the case of my election tracker that constantly updates itself and uses vast amounts of unstructured data, checking by hand is not feasible and simply not quick enough. By setting a threshold for when these machine learning based results are used and also by setting this threshold at a rather conservative, high-level, a form of safety check was implemented that would allow the site and its graphs to fall back on a method that was well-established by other researchers in case the accuracy would drop.

## Story telling and narrative

Besides the technical aspects, I decided to write the explainers on the website in English and German as I thought that this would be the best fit for the audience being interested in this topic. As the structure of the narrative on the website, I divided the storytelling into four parts.

I started with an introduction about the possibilities of Twitter as an ever-updating stream of information on what people talk about. The next part was then shifting the focus to the problem of automated content and the first visualization showing the calculated number of bots. The third part was about how on the other hand the results for “normal-tweets” look like. Finally, I explained the idea behind the project and the methodology.

This structure obviously differs from the structure of most academic papers. The reason for this order in my narrative was that I assumed, the average user would not be very interested in the methodology.

To further enhance the narrative, I also implemented some simple div-elements that displayed the number of tweets, the share of likely bot-tweets and two elements that showed which party has the biggest difference between the two statistics and would lose the most “tweet-share” when looking at non-bot-tweets instead of bot tweets.

Also, after brief consultation with Martin Stabe, head of interactive news at the Financial Times, and Julius Tröger, head of the Berliner Morgenpost interactive team, I decided to phrase the idea of Twitter based election prediction in a cautious manner and present the whole website as an experiment and definitely not a sure shot. This was done, in order to pay tribute to the mixed results in the past of this method and also to the fact that a wrong prediction might face public scrutiny.

## Problems with high number of tweets, possible memory leaks

The application hashtagswahl went live on the 3rd of September 2017 and was supposed to run constantly from then on. However, multiple times the application stopped as the server instance was seemingly out of memory. The reason for this might have been high number of tweets that had to be processed on certain days when, for instance, a debate between the two main candidates was televised. Changing the size of the Amazon server instance from small to medium solved some of the problems. Yet, rerunning the code for election night to replicate the results from the election night showed a substantial problem. With over 180000 tweets the script took extremely long to process the data.

Furthermore, sometimes the tweet processing would completely stop and the ec2 instance would remain unreachable via the console and this could be only fixed by a stop and restart of the instance. A possible explanation for this problem was suggested by Rininsland: so-called “memory leaks”.

This means that certain memory is wrongly allocated and still stored, although it should have been released. There are several approaches to fixing this in Python by using garbage collection methods (Vilkeliskis, 2017). Yet, as again my time budget was small during my internship at the Financial Times, and these memory leak problems would only appear every couple of days, I decided to address the problem at this stage by stopping and restarting the server instance upon occurrence of this problem: a simple, though not most elegant fix.

There was another problem that occurred when I replicated the results of the election night by simply rerunning the script and setting it to the time of the closure of the election polls: the number of counts for bot-tweets and normal-tweets would increase every time the script was run. This was true even though the same set of tweets was used.

Even consultation with my supervisor did not help to find an answer and so I can only speculate whether memory leaks might be the reason for the problems here as well. However, despite the increase of absolute tweet numbers, the proportion of tweets between the parties would stay the same. So even though more tweets for the parties were counted, e.g. the SPD would always be mentioned in 16 per cent of the tweets.

## Assessing which criteria of Gayo-Avello checklist are met

Finally, before talking about how filtering out bots improved the election prediction and especially before assessing whether machine learning or the mere count method were more effective, it is important to consider another aspect: How much of this application is addressing Gayo-Avello’s criteria for a valid usage of Twitter as a tool for election prediction (2012):

* *To be a true prediction, it has to be made public in advance of the election.* This was done by setting up the website on the 3rd of September.
* *Chance is not a valid baseline because incumbency tends to play a major role in most elections and hence, has to be addressed*

My prediction model does not take into account the effect of incumbency, so this is where the model has a weakness.

* *There has to be a clear definition of what a “counting vote” is (tweets, users, some form of sentiment analysis) and also what the result of the “Twitter polling” has to be compared against (polling results, election results, parties’ vote share or parties’ ranking order)*

In line with Celli et al., every tweet was per se counted even if it was a retweet and even when a user tweeted multiple times. The only difference is that bot-tweets were filtered out.

* *If sentiment analysis is used, it has to be tested, validated and not simply be used as a black box*

Validation of sentiment analysis was tried but as it failed, it was decided to not use sentiment analysis.

* *Demographics of Twitter have to be addressed as Twitter is not representative*

This was not done for two reasons. Firstly, because Twitter as a medium has moved on with an older demographic increasingly using the service. Secondly, one possible outcome of my study could be that bot-filtering actually worsens the prediction model as bots might balance out a bias that exists on Twitter towards more liberal, younger users.

* *Take into account that not everyone tweets about politics and not every message is trustworthy*

This was done by filtering out bots.

* *The chosen method should be checked also against past elections*

This was not possible as the application was not build in time for the UK’s General Election.

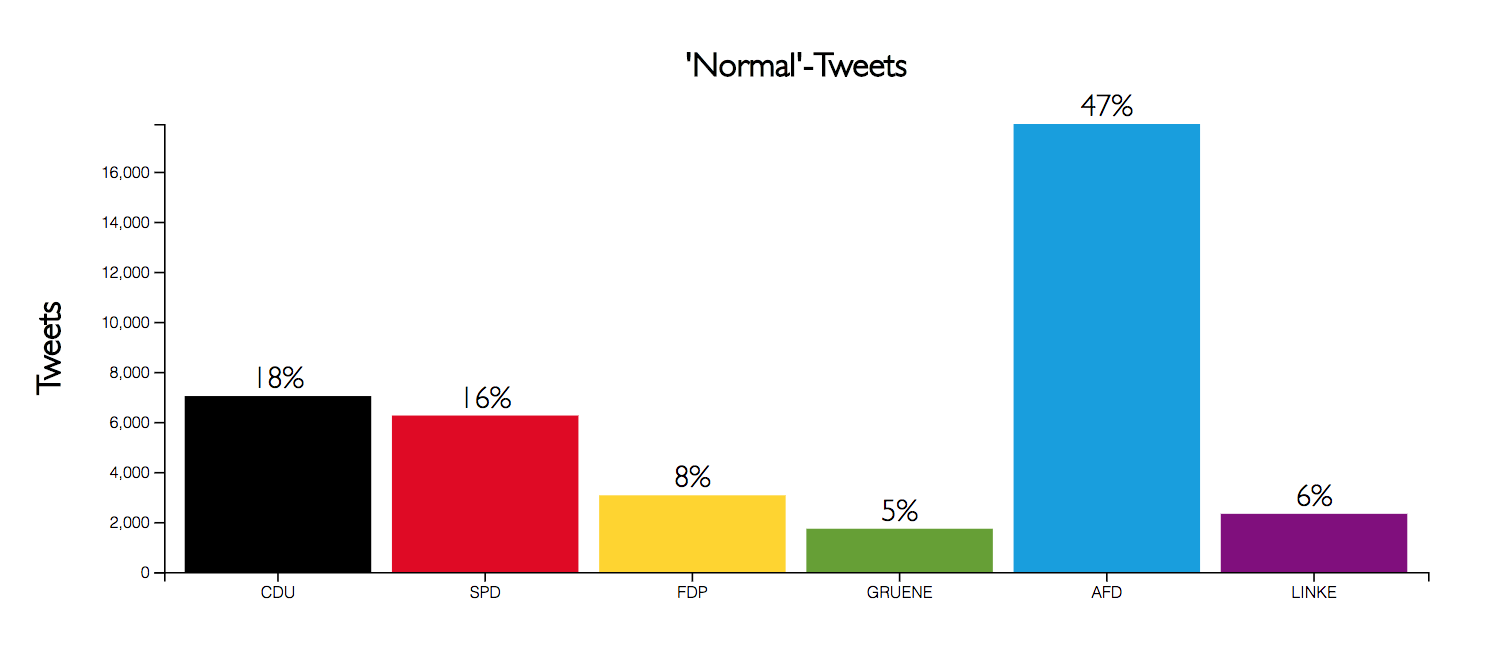
Taken all together, this means out of seven criteria at least four were met.

# Original research part II: Does filtering for bots improve the election predictions?

## The case with the AfD included

On the 24th of September, the new German parliament was elected and the biggest surprise was the performance of the right-wing party “Alternative für Deutschland”, AfD. With 12.6 per cent of the vote share, the party outperformed most predictions and became the third strongest party in the new parliament.

However, if it was for Twitter, the party would not only be in third place. It would have won with 47.3 to 45.9 per cent of the popular vote and be the overall winner of the election by all four measures that I recorded: mentions, non-bot tweets filtered by counts, non-bot tweets filtered by my machine learning algorithm or a combination of both (so every tweet that was neither classified as bot by the count-based method or the machine learning algorithm). This was the outcome after analyzing all tweets that were posted until the closure of the polls at 6 p.m. German time.

*Fig 6. Results of election prediction at 6 p.m. German time (closure of polls)*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Party name | Official result | Mentions | Normal tweets filtered by counts | Normal tweets filtered by ML | Normal tweets filtered by both  methods |
| CDU/CSU | 33.0 | 19.3 | 18.4 | 19.6 | 17.6 |
| SPD | 20.5 | 16.2 | 16.3 | 17.6 | 18.1 |
| AfD | 12.6 | 47.3 | 46.7 | 46.2 | 45.9 |
| FDP | 10.7 | 7.2 | 8.0 | 6.3 | 7.3 |
| LINKE | 9.2 | 6.0 | 6.1 | 6.4 | 6.7 |
| GRÜNE | 8.9 | 4.0 | 4.5 | 4.0 | 4.3 |

*Table 1: Official result and the predicted vote share based on the different models*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Party  name | Official rank | Rank based on mentions | Rank based on normal tweets and filter by counts | Rank based on normal tweets and filter by ML | Rank based by combining both methods |
| CDU/CSU | 1 | 2 | 2 | 2 | 3 |
| SPD | 2 | 3 | 3 | 3 | 2 |
| AfD | 3 | 1 | 1 | 1 | 1 |
| FDP | 4 | 4 | 4 | 5 | 4 |
| LINKE | 5 | 5 | 5 | 4 | 5 |
| GRÜNE | 6 | 6 | 6 | 6 | 6 |

*Table 2: Official ranking order of parties by vote share and the predicted ranking based on the different models*

To show how dramatic the difference is, I created an offset statistic similar to the one used by Sang & Bos (2012). The average offset between the Twitter based prediction and the official result was ranging from 10.7 to 10.3 per cent and even in terms of ranking the average offset was between 0.7 and 1.0.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Partyname | Official result | Mentions offset | Counts offset | ML offset | Both methods offset |
| CDU/CSU | 33.0 | 13.7 | 14.6 | 13.4 | 15.4 |
| SPD | 20.5 | 4.3 | 4.2 | 2.9 | 2.4 |
| AfD | 12.6 | 34.7 | 34.1 | 33.6 | 33.3 |
| FDP | 10.7 | 3.5 | 2.7 | 4.4 | 3.4 |
| LINKE | 9.2 | 3.2 | 3.1 | 2.8 | 2.5 |
| GRÜNE | 8.9 | 4.9 | 4.4 | 4.9 | 4.6 |
| **Offset sum** | **0** | **64.2** | **63.2** | **62.1** | **61.6** |
| **Average offset** | **0** | **10.7** | **10.5** | **10.4** | **10.3** |

*Table 3: Offset between predicted vote share and official result*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Partyname | Official result | Mentions offset | Counts offset | ML offset | Both methods offset |
| CDU/CSU | 1 | 1 | 1 | 1 | 2 |
| SPD | 2 | 1 | 1 | 1 | 0 |
| AfD | 3 | 2 | 2 | 2 | 2 |
| FDP | 4 | 0 | 0 | 1 | 0 |
| LINKE | 5 | 0 | 0 | 1 | 0 |
| GRÜNE | 6 | 0 | 0 | 0 | 0 |
| **Offset sum** | **0** | **4** | **4** | **6** | **4** |
| **Average offset** | **0** | **0.7** | **0.7** | **1.0** | **0.7** |

*Table 4: Offset between predicted ranking and official ranking of parties according to vote share*

Hence, one can certainly say that the approach of Tumasjan et al. (2009) would not have worked for this election. The main reason for this seems to be the AfD, that has an offset of over 30 per cent across all three methods.

This supports observations by journalists and findings by other researchers that pointed out how the AfD has changed the way in which political conversation evolves on German social media. It was shown, for instance, that during the German presidential election (Neudert, Kollanyi & Howard, 2017a) and again before the General Election (Neudert, Kollanyi & Howard, 2017b), content related to the AfD flooded Twitter. This seems to be partly due to bot accounts. However, one has to note as well that the party’s core strategy has been about neglecting taboos, provoking others and steering up political controversy (Hunter, 2016), so some of the tweets mentioning the party could be organically generated by actions that caused a lot of debate.

To get a better feeling of how dramatic the effect of the AfD was, I decided to have a second comparison between Twitter-based prediction results and the actual result in a scenario without the AfD.

## The case with the AfD excluded

The results for the case without the AfD were simply calculated by excluding the AfD and then recalculating the share of votes, mentions, non-bot tweets filtered by counts and non-bot tweets filtered by machine learning respectively. On a first glance, it is already striking how much closer the official result and the Twitter based predictions become.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Party name | Official result | Mentions | Normal tweets filtered by counts | Normal tweets filtered by ML | Normal tweets filtered by both methods |
| CDU/CSU | 40.1 | 36.5 | 39.5 | 36.4 | 32.5 |
| SPD | 24.9 | 30.7 | 30.9 | 32.7 | 33.4 |
| FDP | 13.0 | 13.6 | 11.7 | 11.6 | 13.6 |
| LINKE | 11.2 | 11.5 | 11.5 | 11.9 | 12.5 |
| GRÜNE | 10.8 | 7.6 | 6.5 | 7.4 | 8.0 |

*Table 5: Official result and the predicted vote share based on the different models, in the case of excluding the AfD*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Partyname | Official rank | Mention rank | Rank based on normal tweets filtered by counts | Rank based on normal tweets filtered by ML | Rank based on normal tweets and filter by both methods |
| CDU/CSU | 1 | 1 | 1 | 1 | 2 |
| SPD | 2 | 2 | 2 | 2 | 1 |
| FDP | 3 | 3 | 4 | 4 | 3 |
| LINKE | 4 | 4 | 3 | 3 | 4 |
| GRÜNE | 5 | 5 | 5 | 5 | 5 |

*Table 6: Official ranking order of parties by vote share and the predicted ranking based on the different models, in the case of excluding the AfD*

Yet, looking at the offset helps to quantitatively assess the difference between the outcome and the prediction.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Partyname | Official result | Mentions offset | Counts offset | ML offset | Both methods offset |
| CDU/CSU | 40.1 | 3.6 | 0.6 | 3.7 | 7.6 |
| SPD | 24.9 | 5.8 | 5.9 | 7.7 | 8.5 |
| FDP | 13.0 | 0.6 | 1.3 | 1.4 | 0.6 |
| LINKE | 11.2 | 0.3 | 0.4 | 0.7 | 1.3 |
| GRÜNE | 10.8 | 3.2 | 4.4 | 3.4 | 2.8 |
| **Offset sum** | **0** | **13.5** | **12.6** | **17.0** | **20.8** |
| **Average offset** | **0** | **2.2** | **2.1** | **2.8** | **3.5** |

*Table 7: Offset between predicted vote share and official result, in the case of excluding the AfD*

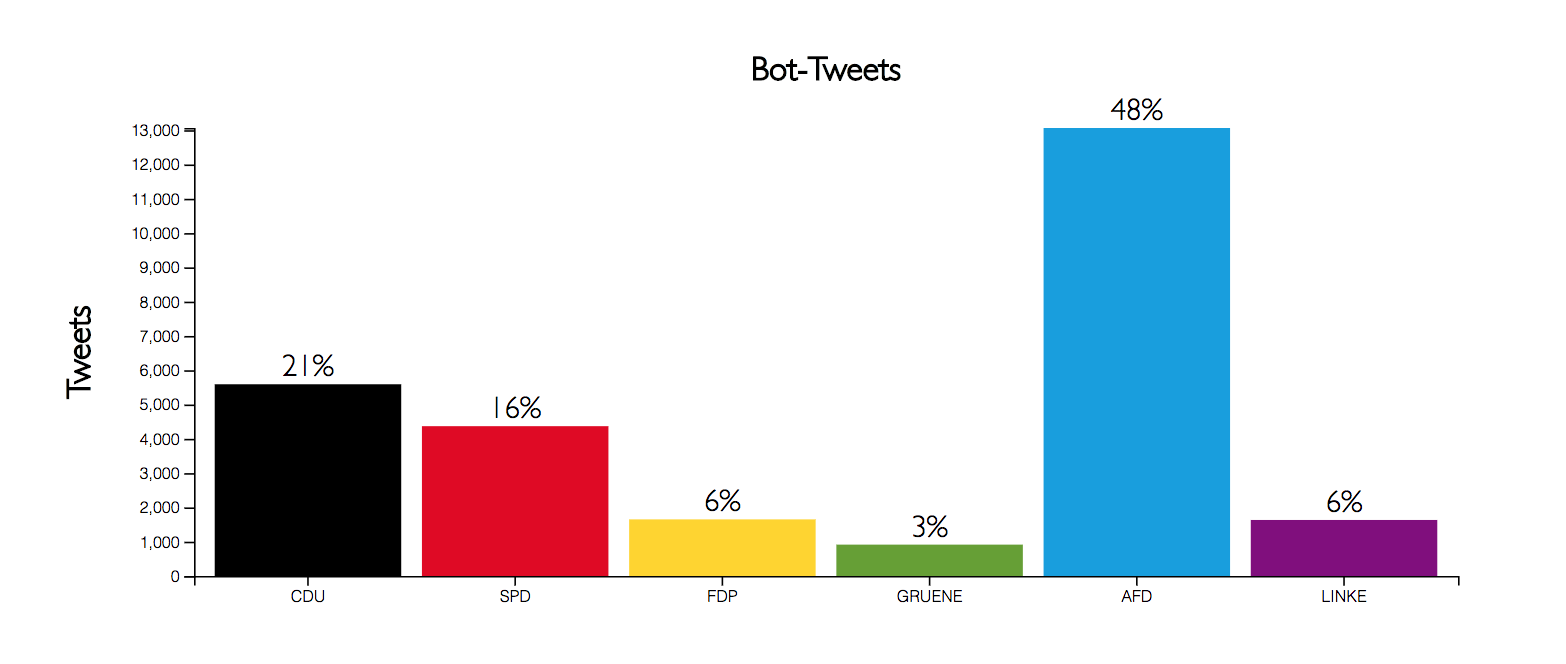
The difference between predictions and actual result lies between 2.1 and 3.5 per cent points on average, which is relatively close to the margin of error of traditional polls prior to an election have. These are normally estimated at maximum around two to three per cent points. Also, the prediction of the ranking order of the parties is completely correct in case of the mention based method without a bot-filter. Yet, it has to be noted that the results are not as accurate as the one reported by Celli et al. (2016), who even outperformed traditional pollsters.

## Impact of the AfD on Twitter predictions

The comparison of the two scenarios shows that including the AfD or not has a big influence on the accuracy of the prediction models and increases the margin of error by a factor of three- to five-fold. This seems to underline that the party and its supporters show a different dynamic and behavior on the social web than supporters from other, more established party. This finding is inline with studies from other researchers who pointed out that the AfD is dominating the discussion on Twitter. To better assess how much automation had contributed to this phenomenon I would like to speak about the statistics for the bot-generated tweet statistics.

## What do the bot-statistics show?

The picture of the bot-statistics is clear: according to both filtering methods the AfD leads the rest of the parties, which hints at a problem. Even, with an accuracy of 69 per cent for the Machine learning bot filter on the election night or the threshold of 24 tweets per account, the chances are high that enough bots passed the filter to significantly influence the outcome. More importantly, taken together with the observation that the accuracy of the machine learning algorithm dipped by 5 per cent after taking all parties rather than only CDU/CSU and SPD, there might be another underlying aspect. Bots might behave differently depending on the party that they tweet about.

*Fig 7. Results of bot counting at 6 p.m. German time (closure of polls)*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Party name | Bot-tweets Count | Bot-tweets Count (%) | Bot-tweets ML | Bot-tweets ML(%) | Bot-tweets both | Bot-tweets both(%) |
| CDU/CSU | 5591 | 20.5 | 7717 | 19.4 | 9697 | 19.8 |
| SPD | 4369 | 16.0 | 6224 | 14.7 | 7609 | 15.6 |
| AfD | 13064 | 48.0 | 19396 | 49.0 | 23310 | 47.7 |
| FDP | 1651 | 6.1 | 3149 | 7.5 | 3492 | 7.1 |
| LINKE | 1634 | 6.0 | 2353 | 6.2 | 2837 | 5.8 |
| GRÜNE | 915 | 3.4 | 1651 | 3.2 | 1919 | 3.9 |
|  | 27224 | 100 | 40490 | 100 | 48864 | 100 |

*Table 8: Statistics for the bots counted according to the different methods*

## The case for a bot-filtering approach

This two scenario analysis can be interpreted as an indicator that in a pre-far-right world, the simple method of Tumasjan et al. (2009) was accurate enough to track political conversations and their effect on elections. Interestingly enough, it also seems to be the case that more simplistic approaches used to perform better than more sophisticated ones that for instance account for demographic differences between Twitter users and the general population, as the literature review showed.

However, currently bot-filtering is definitely needed. The outcome for the scenario with the AfD shows that bot-filtering improves the accuracy of the election prediction, albeit on a very small scale. Therefore, the approach is promising but, at best, at a very early stage. Especially, the machine learning method only slightly improves the accuracy compared with the mere count method, which hints at the fact that more work needs to be done.

# Conclusion

## Simple Twitter election predictions based on mentions do not work

The thesis shows that the concept of election predictions based on Twitter mentions does not work currently. Tweet-based predictions would have wrongly projected the AfD to be the winner of the election, when in reality it was just the party in third place. Furthermore, the offset between predicted vote share (based on mention of parties in tweets) and actual result was in double figures with a mean error of 10.7 per cent points.

## Bot filtering can improve predictions but only on a small scale

Using methods that are trying to filter out bots, the offset could be reduced from 10.7 down to 10.3. This is an improvement but on a very small scale.

However, the results show that bot filtering has become necessary as bots can create the impression that a certain party is supported by more people than it actually is. Using more sophisticated methods than the ones used in this master’s thesis might help to estimate the support for a party more correctly.

## The far-right seems to have been heavily supported by bots

In the case of the German election, it seems that content related to the right-wing party AfD was especially pushed by highly automated accounts. This is shown by the different bot statistics retrieved from my application and also by the fact that the prediction model improves its accuracy significantly once it excludes the AfD.

## Sentiment analysis of text is much more complicated than expected

The poor results of sentiment analysis based on a lexicon-approach and on a stylometric/machine learning approach show how complicated it is to detect sentiment in text. Several iterations and changes could not improve any of the approaches so that the algorithm would detect correctly the sentiment of a text at least in 60 per cent of the cases.

## Agile concepts and modular organization of code are helpful

As much as the outcome of the sentiment analysis was unsuccessful, the minimum viable product concept of the application meant that the overall project was never threatened - merely the aim for the application had to be modified slightly. Also, the set up of sprints and partial deadlines stopped me from spending too much time on sentiment analysis, although setting up all the modules on servers was not done until I was halfway into my internship at the Financial Times, which means the overall delivery was delayed by four weeks.

Furthermore, the idea to code in separate modules allowed me to quickly adjust the project depending on the problems encountered and it also made the handling of bugs much easier.

## Amazon instances and S3 buckets are a lean and efficient hosting solution

Using Amazon’s web services proved to be a helpful decision. Hosting the different Python scripts on the server instances was a solution that I could handle easily and the scripts ran mostly stable on the instances. Also, it allowed me to scale up from a smaller sized instance to a bigger one, once it became clear that more capacity was needed for data handling. Combining the AWS instances with the idea of hosting the website from a S3 bucket was also fitting for the purpose. It allowed me to host the website in a very simple manner and ensured that the graphics on the website would update every 15 minutes when new CSVs were pushed into the bucket.

# Discussion

Looking back at the process of creating the entire application, several points come to mind that could have been executed more efficiently. Setting up the Twitter stream module took unnecessarily long. This was partly because I did not think early enough about web hosting and also was preoccupied with trying to solve data-saving problems in only one way instead of trying several more pragmatic solutions. As a result, I could not trial my approach on data from the UK elections, which would have been useful as it would have allowed me to run my method against two test cases. Also, it would have allowed me to optimize the modules of the application earlier on.

As mentioned earlier, I think that I should have tracked tweets mentioning the Twitter account of the SPD candidate Martin Schulz as this would have been more consistent in terms of methodology. Also, I would still have had the chance to filter these tweets out again, if it had proved problematic to include a greater variety of hashtags and forms of mentions for one candidate than for the other.

Regarding the sentiment analysis and machine learning part, I am fairly content because the thesis gave me the chance to dive into these fields of computer science that I have been interested for quite a while. Understanding how simple the basic implementation of machine learning and sentiment analysis approaches can be and yet, how difficult it is to fine-tune them, was a valuable lesson.

In hindsight, not focusing so much on the agreement/disagreement approach of Celli et al. would have been beneficial as it narrowed my view and I did not look into alternative solutions enough. Especially, the sentiment analysis library Word2Vec might have been something that could have helped and I regret not giving it a try earlier. Also, it might have been better to use more test datasets for the sentiment analysis than relying only on the CorEA dataset.

However, I am fairly content with the outcome of the different bot filters. It is clear that they are still in the early stages of development, but having yielded an accuracy of well above two-thirds and at times even above 80 per cent was a first success.

Also, the fact that my results in terms of bot-detection were comparable with findings by Neudert, Kollanyi & Howard (2017b) who said that before the elections they could attribute about 30 per cent of tweets to the AfD was reassuring.

For the future, I think that helpful enhancements to this method would be to include the mentioned party into the training data when the machine learning model is built or even build different models for different parties altogether. This might be a fruitful new pathway, as it seems that bots behave differently depending on which party they mention. Also, seeing how the amount of data retrieved from the Twitter stream depends on the daytime, I think that including the hour of a tweet might further help to filter out bots, as the chances of a “normal” user to tweet at 3 a.m. are smaller than during the day. The Digital Forensic Lab assembled an interesting list of hints for whether an account is likely a bot or not (DFRLab, 2017) in its overview on this topic and it might be worth implementing it in the future.

The Swiss public broadcaster SRF recently used some of these concepts in a story about fake accounts that increase the number of followers of Instagram influencers. Interestingly enough, the SRF (Scurell & Grossenbacher, 2017) also applied machine learning with a random forest classifier to spot these fake accounts and optimized for their false-positive rate, an additional aspect that would surely have improved my approach as well. Also, the SRF examined the accounts themselves rather than their posts. So, instead of determining through stylometric features for each tweet or post whether it was sent by a bot, they used several clues on the account page to predict whether the account was a bot or not. This might indeed be a more promising method than the one that I used or at least something to use for triangulation with my stylometric approach.

One of the biggest problems that my application still has is that it crashes in times of high traffic, which seems to be due to memory leaks and missing garbage collection. Hence, learning more about this aspect of programming would be of great value for this application but also in general for future projects.

Also, looking at the script, I think that there is some unnecessary repetition in some of the steps. For instance, the current script is recreating the stylometric fingerprint for each tweet from scratch every 15 minutes. As I have not used absolute numbers of commas, dots etc. but always divided them by the average number for this category, this fingerprint constantly changes when the average changes as well. Yet, I could have at least stored the absolute numbers for these categories for each tweet and then recreated the fingerprint by dividing by the new averages instead of calculating everything from scratch. This would have saved the script from going through each letter of each tweet over and over again.

Lastly, I would like to address two aspects that are important when looking at using Twitter as a tool: how volatile Twitter is and the lack of demographics in the approach of Twitter polling

Twitter is incredibly fast in its response to news events, which can be both positive and negative. On one hand, it allows to live track what people are talking about, which is much quicker than any classic opinion poll. On the other hand, breaking news can change the conversation rapidly.

For instance, when I was replicating the analysis of the prediction on election night, I made the mistake of calculating results based on tweets from 8 p.m. on the day before the election until 8 p.m. of the election night. However, poll stations closed at 6 p.m.. This meant my mention based calculation also used tweets that were posted after the first exit polls. As a result, my application wrongly predicted the SPD to be in second place, right after the most probably bot-driven, highly mentioned AfD. This was probably due to the fact that right after the first exit polls after 6 p.m., it became clear that the SPD had not only lost the election by a landslide but also scored the worst results since the end of World War II. This probably lead to the SPD being heavily mentioned on Twitter, skewing my result.

The other issue that still exists with Twitter predictions is its legitimacy regarding the demographic structure. As pointed out by several researchers, Twitter’s user base clearly is not representative for the entire population, yet few studies have actually tried to address this.

The result is that we can definitely say that an application like mine allows to say what the “mood” on the web is. However, despite several studies that show how close Twitter-based predictions can come to the actual result, it still lacks the causal justification for why something as unrepresentative as Twitter can say something representative about the entire voting population.

Hence, I conclude that Twitter was and still is an interesting source for data that allows to see political conversations in real time. Nevertheless, caution is advised when choosing time frames for the collection of data and there is definitely a need for more examination of the links between Twitter’s user behavior and the mood of the population of a country. Lastly, I think that bot-detection is an emerging field of research and I hope that more sophisticated models will allow us in the future to better distinguish human from non-human conversation on the web.

# Link to the Github Repository with the code:

https://github.com/HalukaMB/masterthesis\_final

Lexiconapproach -> Code for the failed sentiment analysis approach

Backend->Code and libraries split up for the two server instances

Website-> Code and csvs for the website

AllResults.xls->Breakdown of the statistics and results used in this thesis

# Literature

1. Aldhous, P. (2017, August 8). BuzzFeed News Trained A Computer To Search For Hidden Spy Planes. This Is What We Found. Buzzfeed. Retrieved October 25, 2017, from <https://www.buzzfeed.com/peteraldhous/hidden-spy-planes?utm_term=.ug4W08DAD#.cnDJQNb8b>
2. Blood, D. (2017, March 14). Is social media empowering Dutch populism?  Retrieved May 4, 2017, from <https://www.ft.com/content/b1830ac2-07f4-11e7-97d1-5e720a26771b>
3. Bollen, J., Mao, H., & Zeng, X. (2010). Twitter mood predicts the stock market. Arxiv. Retrieved October 25, 2017.
4. Borger, J. (2017, July 5). Investigators explore if Russia colluded with pro-Trump sites during US election. The Guardian. Retrieved October 25, 2017, from <https://www.theguardian.com/us-news/2017/jul/05/donald-trump-russia-investigation-fake-news-hillary-clinton>
5. Brendon, P. (2017, March 11). Death of truth: when propaganda and 'alternative facts' first gripped the world. Retrieved November 01, 2017, from <https://www.theguardian.com/media/2017/mar/11/death-truth-propaganda-alternative-facts-gripped-world>
6. Brown, R. S. (2016). Using Scikit-Learn In AWS Lambda. Retrieved November 01, 2017, from https://serverlesscode.com/post/deploy-scikitlearn-on-lamba/
7. Burckhardt, P., Duch, R., & Matsuo, A. (2016). Tweet as a Tool for Election Forecast: UK 2015 General Election as an Example. Third Annual Meeting of the Asian Political Methodology Society.
8. Burnap, P., Gibson, R., Sloan, L., Southern,R., &Williams, M. (2015). 140 characters to victory?: Using Twitter to predict the UK 2015 General Election. Electoral Studies, 230-233. Retrieved October 25, 2017, from <http://www.sciencedirect.com/science/article/pii/S0261379415002243>
9. Cao, Q., Sirivianos, M., Yang, X.m & Pregueiro, T.(2012). Aiding the Detection of Fake Accounts in Large Scale Social Online Services. Proceeding NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (pp. 12-15).
10. Ceccon, S. (2015, May 18). The election explained by the Times Data Team. Retrieved October 25, 2017, from <https://medium.com/digital-times/the-election-explained-by-the-times-data-team-482d3ab95f5c>
11. Ceron, A., [Curini](https://www.researchgate.net/profile/Luigi_Curini), L., Iacus, S., & Porro, G. (2014). Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. New Media Society, 16, 340-358. Retrieved October 25, 2017, from <http://journals.sagepub.com/doi/abs/10.1177/1461444813480466>
12. Celli, F., Stepanov, E., Poesio, M., & Riccardi, G. (2016). Predicting Brexit: Predicting Brexit: Classifying Agreement is Better than Sentiment and Pollsters. Retrieved May 4, 2017, from <http://www.sensei-conversation.eu/wp-content/uploads/2016/11/NLP_brexit_final_copyright.pdf>
13. Choy, M., Cheong, M., Laik, M., & Shung, K. (2012). A sentiment analysis of Singapore Presidential Election 2011 using Twitter data with census correction. Arxiv. Retrieved October 25, 2017, from <https://arxiv.org/abs/1108.5520>.
14. Davis, C. A., Varol, O., Ferrara, E., Flammini, A., & Menczer, F. (2016). BotOrNot. Proceedings of the 25th International Conference Companion on World Wide Web - WWW 16 Companion. doi:10.1145/2872518.2889302
15. Dolinar, S. (2015, January 28). Collection Twitter Data: Using a Python Stream Listener. Retrieved October 25, 2017, from [http://stats.seandolinar.com/collecting-Twitter-data-using-a-python-stream-listener/](http://stats.seandolinar.com/collecting-twitter-data-using-a-python-stream-listener/))
16. DFRLab. (2017, August 29). #BotSpot: Twelve Ways to Spot a Bot. Retrieved October 25, 2017, from https://medium.com/dfrlab/botspot-twelve-ways-to-spot-a-bot-aedc7d9c110c
17. Election 2015. (n.d.). Retrieved October 26, 2017, from http://www.bbc.com/news/election/2015/results
18. Election 2017. (n.d.). Retrieved October 26, 2017, from http://www.bbc.com/news/election/2017/results
19. Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. Communications of the ACM,59(7), 96-104.
20. Frank, E., Hall, M. & Witten, I.(2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.
21. Gayo-Avello, D. (2012). No, You Cannot Predict Elections with Twitter. IEEE Internet Computing, 16(6), 91-94. Retrieved October 25, 2017, from <http://ieeexplore.ieee.org/abstract/document/6355554/>
22. Howard, P. N., & Kollanyi, B. (2016). Bots, #Strongerin, and #Brexit: Computational Propaganda During the UK-EU Referendum. SSRN Electronic Journal.
23. Hunter, J.S. (2016, December 20)Dieses geleakte Strategiepapier zeigt, wie die AfD bewusst provoziert. Business Insider. Retrieved October 25, 2017, from http://www.businessinsider.de/dieses-geleakte-strategiepapier-zeigt-wie-die-afd-bewusst-provoziert-2016-12
24. Jungherr, A., Jürgens, P., & Schoen, H. (2012). Why the Pirate Party Won the German Election of 2009 or The Trouble With Predictions: A Response to Tumasjan, A., Sprenger, T. O., Sander, P. G., & Welpe, I. M. “Predicting Elections With Twitter: What 140 Characters Reveal About Political Sentiment”. Why the Pirate Party Won the German Election of 2009 or The Trouble With Predictions: A Response to Tumasjan, A., Sprenger, T. O., Sander, P. G., & Welpe, I. M. “Predicting Elections With Twitter: What 140 Characters Reveal About Political Sentiment”. Social Science Computer Review,30(2), 229-234. Retrieved October 25, 2017, from <http://journals.sagepub.com/doi/abs/10.1177/0894439311404119>
25. Kirk, A. (2017, February 20). EU referendum: The claims that won it for Brexit, fact checked. Retrieved November 01, 2017, from <http://www.telegraph.co.uk/news/0/eu-referendum-claims-won-brexit-fact-checked/>
26. Konnikova, M., Taylor Gee, Brent Griffiths and Ruairí Arrieta-Kenna, Shafer, J., Graff, G. M., Schreckinger, B., Zilber, N., & Knott, S. F. (n.d.). Trump’s Lies vs. Your Brain. Retrieved November 01, 2017, from https://www.politico.com/magazine/story/2017/01/donald-trump-lies-liar-effect-brain-214658
27. Lampos, V.[, & Cristianini, N.](https://research-information.bristol.ac.uk/en/persons/nello-cristianini(1ebaf390-5345-4993-8200-15c7fa98548a).html) (2010). [Tracking the flu pandemic by monitoring the social web](https://research-information.bristol.ac.uk/en/publications/tracking-the-flu-pandemic-by-monitoring-the-social-web(8d9abf6f-a41b-4f71-afd4-8ccb9c2209b6).html). Cognitive Information Processing (CIP), 2010 2nd International Workshop on, 411 - 416.
28. Lee, K., Eoff, B. D., & Caverlee, J. (2011). Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter. AAAI Int’l Conference on Weblogs and Social Media (ICWSM}.
29. Natural Language Generation Technology. (n.d.). Retrieved October 25, 2017, from https://narrativescience.com/#advanced-nlg
30. Officer, T. F. (n.d.). Results. Retrieved October 25, 2017, from <https://www.bundeswahlleiter.de/en/bundestagswahlen/2017/ergebnisse.html>
31. Neudert, L. M., Kollanyi B., & Howard, P. N (2017a). Junk News and Bots during the German Federal Presidency Election: What Were German Voters Sharing Over Twitter? Retrieved October 25, 2017, from http://comprop.oii.ox.ac.uk/wp-content/uploads/sites/89/2017/03/What-Were-German-Voters-Sharing-Over-Twitter-v6-1.pdf
32. Neudert, L. M., Kollanyi B., & Howard, P. N (2017b). Junk News and Bots during the German Parliamentary Election: What are German Voters Sharing over Twitter? Retrieved October 25, 2017, from <http://comprop.oii.ox.ac.uk/wp-content/uploads/sites/89/2017/09/ComProp_GermanElections_Sep2017v5.pdf>
33. Newman, N. (2017). Reuters Institute Digital News Report 2017 . Retrieved October 25, 2017, from <https://reutersinstitute.politics.ox.ac.uk/sites/default/files/Digital%20News%20Report%202017%20web_0.pdf>, p.10
34. Presidential Results. (2016). Retrieved November 2, 2017, from http://edition.cnn.com/election/results/president
35. Ratkiewicz, J., Conover, M., Meiss, M., Gonçalves, B. , Flammini, A. & Mencze, F. (2011). Detecting and Tracking Political Abuse in Social Media. Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (pp. 297-304).
36. Rosenbach, M.(2016, October 21). Internet-Kommentare von Automaten: AfD will im Wahlkampf Meinungsroboter einsetzen - SPIEGEL ONLINE - Netzwelt. Retrieved November 02, 2017, from http://www.spiegel.de/netzwelt/netzpolitik/afd-will-im-wahlkampf-social-bots-einsetzen-a-1117707.html
37. Sang, E., & Bos, J. (2012). Predicting the 2011 Dutch Senate Election Results with Twitter. Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (pp. 53-60).
38. Scurrell, J. V., & Grossenbacher, T. (2017, October). Identifying a Large Number of Fake Followers on Instagram. Retrieved October 25, 2017, from https://srfdata.github.io/2017-10-instagram-influencers/
39. Shi, L., Agarwal, N., Agrawal, A., Garg, G. & Spoelstra, J. (2012). Predicting US Primary Elections with Twitter (2012). Retrieved October 25, 2017, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.252.8161
40. Tsvetkov, Y., Faruqui, M., & Dyer, C. (2016). Correlation-based Intrinsic Evaluation of Word Vector Representations. Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP.
41. Tumasjan, A., Sprenger, T. Sandner, P., & Welpe, I. (2010). Election Forecasts With Twitter: How 140 Characters Reflect the Political Landscape. Social Science Computer Review, 29(4), 402-418. Retrieved October 25, 2017, from http://journals.sagepub.com/doi/pdf/10.1177/0894439310386557
42. Tumasjan, A., Sprenger, T. Sandner, P., & Welpe, I. (2012). Where There is a Sea There are Pirates: Response to Jungherr, Jürgens, and Schoen. Social Science Computer Review, 30(2), 235-239. Retrieved October 25, 2017, from <http://journals.sagepub.com/doi/abs/10.1177/0894439311404123>
43. Twitter Usage Statistics. (n.d.). Retrieved November 01, 2017, from http://www.internetlivestats.com/Twitter-statistics/
44. Vilkeliskis, T. (2017, February 5). Three Techniques for Catching Memory Leaks in Python and How to Write Smart Code. Retrieved October 25, 2017, from <http://vilkeliskis.com/articles/three-techniques-for-catching-memory-leaks-in-python>
45. Wang, G., Mohanlal, M., Wilson, C., Wang, X., Metzger, M., Zheng, H.,& Zhao, B. (2012). Social Turing Tests: Crowdsourcing Sybil Detection. Arxiv. Retrieved October 25, 2017, from https://arxiv.org/abs/1205.3856.
46. Wang, H., Can, D., [Kazemzadeh](https://dl.acm.org/author_page.cfm?id=81100096565&coll=DL&dl=ACM&trk=0&cfid=822518163&cftoken=79403031" \o "Author Profile Page" \t "_self),A. ,Bar, F., Narayanan, S. (2012). A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (pp. 115-120).
47. Wakabayashi, D., & Shane, S. (2017, September 27). Twitter, With Accounts Linked to Russia, to Face Congress Over Role in Election. The New York Times. Retrieved October 25, 2017, from https://www.nytimes.com/2017/09/27/technology/Twitter-russia-election.html

1. N-grams are defined as cluster of words that appear next to each other and might be characteristic for the sentiment of a text, such as in the case of the bi-gram “not convincing”. [↑](#footnote-ref-1)