

# Analysing Tweet Sentiments Relative to Brexit-Related Events

Ned O'Hara, Felix Nirsberger & Tilmann Sager

FACULTY OF MATHEMATICS & COMPUTER SCIENCE

APPLIED COMPUTER SCIENCE

DIGITAL HUMANITIES

—PROJECT REPORT—

This project report is the examination for the lecture "Introduction to Digital Humanities" at the University of Leipzig. The project aim was to analyse the sentiment of British newspaper publisher's twitter followers in relation to Brexit-related events to show how social-media communities express their opinion on macro-scale to political events. Statuses of followers of different British newspaper publishers were filtered for Brexit-related statuses in an attempt to provide an overview of different political camps in the discussion surrounding the exit of the United Kingdom from the European Union. These follower's tweets were then analysed for their sentiment via open source natural language processor. The following report demonstrates the overall negative trend in the Brexit-related sentiments and portrays the varying reactions that newspaper's followers expressed relative to Brexit-related events.

## Introduction

Amidst some of the wildest political demonstrations within the last decade, the Brexit fiasco certainly has not failed to gain the attention of the entire European continent and change the way that democracy is viewed in our digital age. With the recent Facebook-Cambridge Analytica data scandal having been regarded as the watershed moment in public understanding of personal data, a new heightened awareness for the level of protection in the way data is handled and processed has emerged. , see (4)) As accessibility to information regarding citizens increases alongside technological capability to steer the thoughts and opinions of said citizens, one might suggest that the fundamental structure of democracy is bending to the will of those at the spearhead of these technologies.

Taking Brexit-specific events as our contextual focus, our intention was to analyse the course of political dialogue respective to newspaper publishers - with the aim of providing a visual way to interpret the political landscape, as a sum of the concerns and praises (analogous to positive and negative sentiment) of the opinion of a respective newspaper publishers following. Similarly, at its most basic level, politicians in the digital age are obligated to hear the praise and concerns of the

citizens they represent. Provided that the democratic election process places them in their position of representation based on their adherence to the claims made to satisfy said concerns and uphold praises. In reality, the role of a politician is multi-faceted. Some of the duties of a politician include the management of public image, which involves closely working with newspaper publishers and managing social media platforms, which play a vital role in the exposure of a given politicians positive or negative congruence with their claims, or multiplying/maintaining a politicians following through public relations work.

In the completion of this project with time and resource constraints, we intended to recreate a visualisation that represents the sentiment of dialogue that occurred throughout Brexit from its inception to the time of writing - in an attempt to document the public opinion of users relative to specific newspaper publishers on the somewhat politically-focused micro-blogging platform, Twitter - in the hopes of identifying relations between a newspaper publishers following and the opinions they represented surrounding key events.

## Research Aim and Motivation

At the time-of-writing, Brexit is reaching a peak in its relevancy. With the article 50 deadline reaching its expiration date on the 29th of March, citizens of Britain are in a heightened state of political activity. Due to the high level of political discourse in parliament and throughout the country, conversations on social media are consequently representative of the high level of discussion. Twitter, a social media platform which harbours a large political community, accumulates 650 tweets per-hour tagged solely with the #brexit hashtag at the time-of-writing. Needless to say, Brexit - in addition to being a relevant point of discussion on social media platforms, indirectly affects the global climate. Inclusive of the team working on this report! As EU Citizens, (and one dual DE/GB national) motivation to undertake this project was partially due to being either directly or indirectly affected by the implications of Brexit. Prepositions with respect to the perspectives of writers of this report include that of two EU citizens (German nationals) and the perspective of a British/German dual-national. One could postulate that the two EU citizens writing are indirectly affected by the Brexit affair, additionally the DE/GB national is also indirectly affected, (due to his status of dual citizenship, he is not strongly-bound to the positioning of his home nationality) yet a personal note of direct influence is to be made, as his upbringing was spent in the UK.

As European citizens who are outcome-dependant of the decision made by the British people in 2016, in light of the recent Facebook-Cambridge Analytica scandal we believed it would be worthwhile pursuing exploration in the sphere of social media analysis due to both its relevancy and the level of public vulnerability to the technology allegedly used by Cambridge Analytica in the run-up to the referendum, providing a transparent overview of the landscape of opinions found on social media platforms at macro-scale. Leading us to the research question: How does the mood of newspaper publisher's followers on Twitter change relevant to Brexit-specific events?

## Organisation

We scheduled our project in four phases: planning, research, development and evaluation. In the first phase we discussed the higher goal of our project, especially

formulating a research question and what is feasible given the limited time-frame and resources available. The second phase involved the analysis of the Twitter API, the finding of a suitable text analysis framework and the selection of suitable visualisation frameworks regarding the key variables we wanted to document and monitor given the technical limitations. In the third phase we set-up data retrieval by using the Java library `twitter4j`, then went to filtering and calculation of the sentiment values and lastly, the visualisation of the calculated sentiment values.

## Technical Methodology and Setup

Using tools such as Trello and Telegram, we were able to communicate and organise weekly meet-ups in which all team members could effectively discuss project-related issues. In addition to this, we used a SCRUM-like methodology to effectively manage each team-members progress in their field of responsibility using "To-Do", "Doing" and "Done" task columns. From a technical perspective, our project methodology was structured as follows: a twitter API crawler was to be built, which only filtered for user-suitability and twitter-determined language (English) of the tweets crawled, (see Data Retrieval section) but did no Brexit-related semantic filtering of keywords. The crawler was designed to get the tweets of the followers of a given user and put them in a CSV file. Then Brexit-related tweets had to be found in the output files and the analysis tool had to assess these tweet's sentiments. Finally the retrieved and analysed data had to be migrated to a relational database. As a framework to manage the data, undertake the programming for the visualisation and implement a small web-app that primarily acts as our front-end - we used the Python MVC-framework Django. It gave us the possibility to easily manage data stored in a relational PostgreSQL database by defining data access objects, so called "Models". The models we defined can be seen in the following relational scheme:

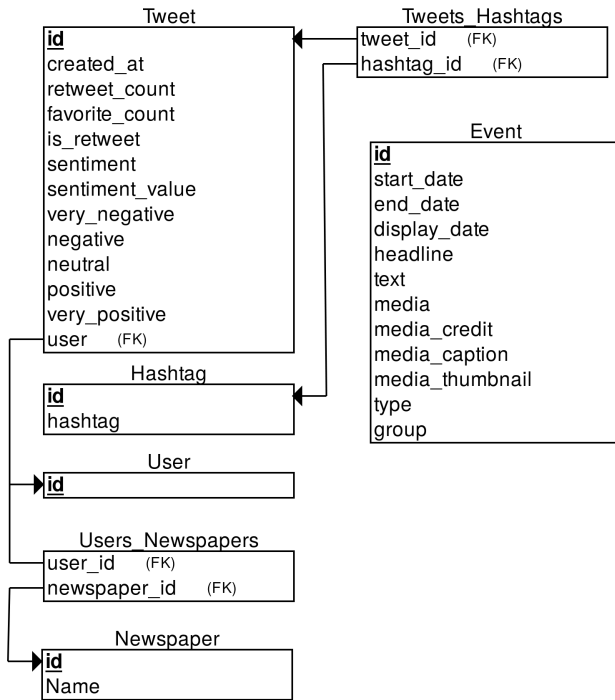


Figure 1. Relation Scheme of Database

Another benefit Django gave us was to easily connect the models we migrated to views in the front-end which we used for our visualisation, (see section Visualisation).

## Data Retrieval

Initially, we decided *tweepy* - a python library designed to access the twitter API, would be a suitable fit for project needs and would fit to the technical affinity of the individual to which the task was allocated to. However, although Python is deemed an easy-to-learn language, the library documentation was incomplete and lacking the level of completion necessary for an individual unfamiliar with the language to work with, given the project time constraint. For this reason, a Java library was preferred, namely *twitter4j*. Using Java over Python provided a number of benefits and drawbacks - benefits included, given that Java is considered a more low-level language than Python, the ability to meet our technical requirements specifically with precision and the predisposed technical affinity of the designated team member. Drawbacks included the increased size of the code, time taken to program the data collection portion of the project and encountered difficulty in the compilation execution of the data-gathering on our project server.

The following procedure describes how the data was collected: in the execution of the data-gathering code, arguments were passed with the command line instruction. Where argument 1 is a public twitter user display name (in our use-case, we input newspaper publishers display-names here), argument 2 was the approximate threshold for the number of tweets to gather from the audience of the public user.

We used one authenticated personal twitter account to provide the authentication necessary to access the Twitter API. We applied for institutional API authentication but unfortunately twitter has not granted access to additional API-keys we applied for, as of the time-of-writing. Having only one authenticated account severely limited our resource gathering rates, as it was technically possible to swap API-keys when rate limits became depleted. However, in sticking to one authenticated API-key, we adhered to the twitter terms of use guidelines (13). Free authentication meant we were also more strongly rate-limited by the API, which in turn extended the computation time significantly and prohibited us from using twitter search functions behind paywalls.

Twitter ID's of the respective newspaper publishers followers were accessed via the GET followers/ids API reference, (12) at 5000 followers per request, with 15 requests per 15 minute rate-limiting window. (Effectively 5000 follower ID's per minute.) After having made a GET followers/ids (12) request, the program would begin by caching the user objects of the 5000 follower ID packet and check processing suitability using GET users/show, (12) this was done primarily to screen accounts that were protected, (protected is a setting which twitter users can enable to restrict the visibility of tweets to twitter users not followed by the protected user) but was also used to calculate the suitability of the user in concordance to project goals and constraints.

Following a project check-in with Dr. Thomas Koentges, we discussed how to determine the suitability of individual users following a given newspaper publisher, due to the magnitude of followers, users had to be filtered intelligently in order to adhere to project team/re-

source limitations, like rate limiting. This suitability was determined using the following pseudo-code:

```
for (User user : eachUser) {
    if (user.hasTweetsCount() >
        3200) {

        posterFrequency = user.
            hasTweetsCount()
            / (DaysSince)user.
            dateAccountCreated();

        idealFrequency = 3200/
            DaysSinceUnofficialReferendum
            ;

        if (posterFrequency <=
            idealFrequency)
        {
            user.process();
            /* Attempts to gather the
            tweets of the selected user.
            */
        }
        else
        {
            continue;
            // select next user for
            suitability check.
        }
    }
}
```

Listing 1: Pseudocode for Suitability

Considering the twitter GET statuses/user\_timeline reference only allows the crawling of 3200 of a users most recent tweets, (12) it would obscure the data-set if users with tweet counts much greater than the 3200 limit are regularly crawled, as this would result in the addition of data which did not completely range back to the time-frame we intended to investigate, which in turn would disproportionately result in a larger number of tweets having been collected with more recent date metadata. In order to combat this issue, we calculated the ideal posting frequency of a user by taking the maximum amount of tweets crawl-able per user and dividing it by the number of days since the

unofficial start of the referendum (16).

User posting frequency was determined by dividing the posting frequency of the user selected, with the number of days since the user's account was created. In the code above, users were processed on condition of having a posting frequency less then the ideal frequency. To increase program efficiency in terms of rate limit effectiveness given the limited project time restriction, if this condition was not met - the user would not be processed and the program would select the next user to check the suitability of.

User processing continued provided the aforementioned condition was met, the suitable user's tweets were gathered in 200 tweet batches using GET statuses/user\_timeline using pagination supported by twitter4j. Up to 3200 tweets (Status objects) were gathered for each user, which were cached and written to a CSV file in UTF-8 encoding after the program had finished processing each individual user, in order to minimise the amount of data cached by the program. 900 requests per 15 minute rate-limiting window were permitted with the free twitter API and data was saved in the following format:

```
{"FollowerID", "TweetID", "Status",
    "Date_Posted", "Country", "Geometry_Coordinates",
    "In-Reply-To_ScreenName", "Favourite_Count",
    "Retweet_Count", "Is_Retweet?", "Hashtags"}
```

In order to save data in the above formats, User and Status object methods were used. Hashtags were saved with a ";" delimiter. Conveniently little regex was needed to turn hashtags into a string, as a hashtag set was provided by *twitter4j* within the respective status object. During the process of creating the program, the question of whether or not retweets should be added as part of our data-set was still up to debate. For this reason a boolean *Is\_Retweet?* column was introduced.

After finishing the processing for the cached packet of 5000 user IDs, the program would get another 5000 user IDs and add it to the cache, repeating this process until the threshold for tweets collected (provided in the 2nd argument in the program execution code) was surpassed. However, the program would finish processing the currently cached user IDs before halting,

resulting in the collection of tweets significantly over that which was provided in the 2nd argument of the program execution code. As a result of this inconsistency, we decided to run the program for each newspaper on the server for a duration of exactly a day, in an attempt to complete the collection of data with the least bias possible given our project time/resource constraints.

Rate limit status checking was completed using the `RateLimitStatus` object, which also had rate limits. In order to complete large amounts of data collection without checking (and subsequently exhausting) the rate limit for each individual unit of data, rate limit count checks were made using `.getRemaining()` at the start of the program and decremented as the program completed varying types of GET requests. Once the count for the rate limit type reached 0, a `Thread.sleep()` for 15 minutes was enforced and the rate limit count remaining was replenished.

Following the execution of the program for a day per newspaper publisher on our home server, we were left with a CSV file for each newspaper publisher, containing the tweets of a large sample of their following.

### Pre-processing

Upon the completion of the data-collecting process we had 8 CSV files - one for each newspaper publisher, of which ranging from a size of between 2.5 - 7.5 million tweets. Our next challenge was to filter these files for the Brexit-related tweets and analyse their sentiments. Some pre-filtering was performed, as stated in the Data Retrieval section, such as the filtering by twitter-determined detected language of cached tweets. Using the language determined by twitter stored in the `status` object of the tweet, we filtered for tweets with the English language tag as we wanted to focus our data-set on the discourse and opinions occurring in the UK and avoid external EU commentary.

**Filtering.** Firstly, our approach to filter for Brexit-related tweets encompassed Brexit-related hashtags we gathered from a service called RiteTag (6). This service provides an API that returns hashtags relevant to a target description/keyword, in our case this was the singular keyword *brexit*. A problem we encountered after looking at the results was that there were hashtags

which were indirectly-related to the Brexit topic which would have consequently added noise to the data-set, given that they were also related to other terms and cultural nuances. A good example for this kind of hashtag is *#eupolitics*.

Hence, we decided not to use all the hashtags from the API and rather filter for a smaller amount of keywords and hashtags with the intention of collecting Brexit-related tweets with maximal probability. Even though using this approach would ignore some tweets that may be Brexit-related, those tweets would have been associated with a high level of noise in the data-set. The terms we were using for our filtering are: 1. *brexit*, 2. *bremain*, 3. *vote leave*, 4. *vote remain*, 5. *voteleave*, 6. *voteremain* and 7. *eucitizens*. We attempted to neutralise bias by selecting equal keywords from both sides of the political spectrum of opinion, however we discovered this method may have been problematic and it will be discussed further in the *Biases* section.

By trimming hashtag symbol and searching for each of the above keywords in the status-body string, we handled keywords and hashtags exactly the same. In doing so, tweets containing e.g. *brexit* as a part of the status-body and tweets containing *#brexit* would both appear in the resulting data. This is also why *vote leave* and *vote remain* both appear twice, once with space and once without.

**Sentiment Analysis.** Initially, our first thoughts for the sentiment analysis included the use of IBM Watson's Cloud AI Service (Tone Analyser) which is known to give reasonable results for sentiment analysis via API both technically and contextually in the form of multidimensional analysis of the query into various categories of human emotions. The problems we faced here were twofold - restriction of API Calls for free users and the performance via remote API, which would limit use to about 1 query per second. We consequently began looking for alternative approaches to the sentiment analysis portion of the project.

The following table shows the problems and benefits of different Sentiment Analysis Tools we tested with sample data from movie reviews.

	Pro	Contra
<b>IBM Watson</b>	Accuracy	Low Performance, Query Limitation of 2500 per month
<b>Textblob</b>	very fast, free and unlimited	not accurate
<b>Opinionfinder</b>	free and unlimited	bad usability, not accurate, no overall sentiment
<b>Stanford NLP</b>	Accuracy, gives sentiment distribution	needs to be self-hosted to be fast

After comparing the different analysing-tools we decided to use the Stanford NLP Kit , see (9)) which delivers an Artificial Intelligence written in Java, that creates sentiment values from *Very Negative* to *Very Positive* as so called "Annotations". Though this method was not multidimensional in its analysis of human emotion like the IBM Tone Analyser, it sufficed our aim of representing an overview of public opinion regarding the British EU referendum. Its approach involves building a tree structure out of a text body (a status-body in our use-case), assessing the sentiment of single words and joining the sentiments of child nodes to finally receive a sentiment for the root node, which represents the analysed document. We found that the easiest way to use the Stanford NLP Kit involved starting it as a server and sending API calls to it. There are several useful wrappers, that make communicating to the API quite easy. We used a python wrapper called `PyCoreNLP` , see (7)). In order to provide adequate project infrastructure, we set-up a static server by taking an old gaming PC, installing Ubuntu Server 18 and connecting it to a Virtual Private Network to be able to access it remotely. This server also gave us a platform to run the filtering and analysing script and to host our database and the web-app we built to visualise the data.

Based on these considerations we wrote a python script that first reads one of the newspaper CSV files and iterates through the rows. In doing so, it cleans the tweet with a regular expression, that removes twitter-specific content like hashtag symbols, links and the lead-

ing "RT" that marks retweets. Next, the cleaned tweet was sent to the locally-hosted NLP Server, that returns its sentiment values. A CSV writer finally appends the gathered sentiment information to the CSV line and writes it to an output file. With this approach we were able to detect and analyse approximately 1000 Brexit-related tweets per minute. To migrate the database with our collected data we created another python script that imports the created Django data-models and creates `Tweet` objects with relations to uniquely-created users newspapers and hashtags.

## Visualisation

In terms of discussing the visualisation data, several points have to be discussed first:

1. What common aspects should be highlighted throughout all figures?
2. Using which type of visualisation can these aspects be displayed with highest interpretability?
3. Which visualisation libraries fulfil these requirements?

With the intention of answering these questions we began researching several visualisation frameworks. The main goal was to visualise the particular sentiment-shift of different newspaper followings according to Brexit-specific events. Therefore a line chart, showing a time-span from 10 days before-and-after a specific event, suits our needs best. To visualise our raw-data for highlighting possible biases we chose a pie chart. After some time researching we decided to use `D3`, a popular visualisation framework written in JavaScript, see (2), for the web app and `seaborn`, see (15) for the report charts. The decision for `D3` was based, on the one hand, on the richness of features `D3` offers and on the other hand, the interactivity of its charts. The Python library `seaborn` bases on the `matplotlib`, and can plot several chart types that are highly-customiseable. To simplify the usage of `D3`, we implemented the Python wrapper `django-nvd3`, see (1), that allows us to work completely in Python, while `django-nvd3` takes care of the chart rendering.

Having a working visualisation environment and chart-types, we had to prepare the data to display our

key-aspects in the most visually-interpretative way possible, whilst preventing bias. As described before, we chose to use the line chart for the visualisation of the sentiment-shift under specific circumstances, e.g. a specific event. The x-axis shows the dates, while the y-axis the sentiments. Though one tweet's calculated sentiment could have an integer value between 0 and 4, the drawn line was jagged. For improving the meaningfulness of the chart we tried to implement several statistical regression methods. A linear regression was easy to implement, because `django-nvd3` and `seaborn` offer built-in regression methods. But a linear function only shows the general trend, not the particular shift. Therefore we tried to implement the statistical model ARIMA, Autoregressive Integrated Moving Average, which is often used for time series analysis. The Python library `statsmodels`, see (10), also includes several methods calculating an ARIMA model. Unfortunately, ARIMA needs fairly-balanced data. That means, in our use-case, that the frequency of tweets has to be constant over the entire time-frame. Since this is not the case, as we only could get 3200 of the most recent tweets of a user - we collected a lot more tweets with a time-stamp between November 2018 and March 2019 than in 2017 itself. Due the difficulties calculating a proper ARIMA model, we decided to implement a simpler method named `rolling().mean()` of `pandas`, see (5) library. It calculates the mean of specific period of time with the result that the shifts are still visible but the line looks far more meaningful. Unfortunately it manipulates the raw-data, so the origin span of y-axis from 0 to 4 was consequently reduced to a span between 1 and 2 as a result of the `rolling().mean()`. Understanding this project as a feasibility analysis, we disregarded that aspect due to increased interpretability of the resulting visualisation.

## Results

For our purpose, we selected eight newspapers from UK, eleven important Brexit events and seven keywords by which filter the tweets. According to the first point we tried to include newspapers from political far-left to far-right position based on: (8)

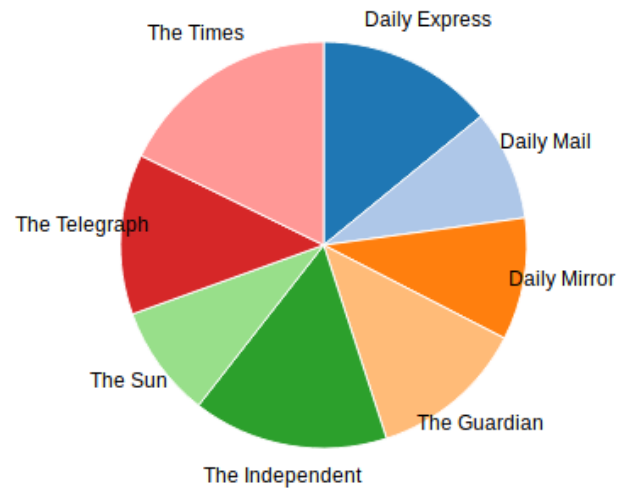


Figure 2. Amount of Followers per Newspaper

Newspaper	Position	Followers
The Guardian	left-winged	7.51m
The Mirror	left-winged	1.06m
The Independent	centre	2.64m
The Times	centre, right-winged	1.18m
The Telegraph	right-winged	2.58m
The Sun	right-winged	1.45m
The Daily Express	right-winged	1.45m
The Daily Mail	right-winged	1.45m

Surprisingly and in contrary to the original amount of followers on Twitter the filtered data in Figure 2 shows a nearly uniformly-distributed amount of followers per newspaper.

In the Commons Library Briefing (14) a good overview about all the Brexit-related events was given. Thereof we chose eleven events, among others the Referendum June 2016, the trigger of Article 50 in March 2017, the Prime Minister's lost of majority in June 2017, the win of the Confidence Vote in December 2018 and the lost of the Meaningful Vote in January 2019. Because we received a lot more data for the end of 2017 till the beginning of 2019 we focused on events in this time span with the exception of the referendum.

We selected seven expressions (keywords) for filtering the retrieved tweets: *brexit*, *bremain*, *vote leave*, *vote remain*, *voteleave*, *voteremain* and *eucitizens*. The keywords were selected in a similar way like the newspapers. There should be a similar amount of keywords in the context of *leave* like in the context of *remain*. In

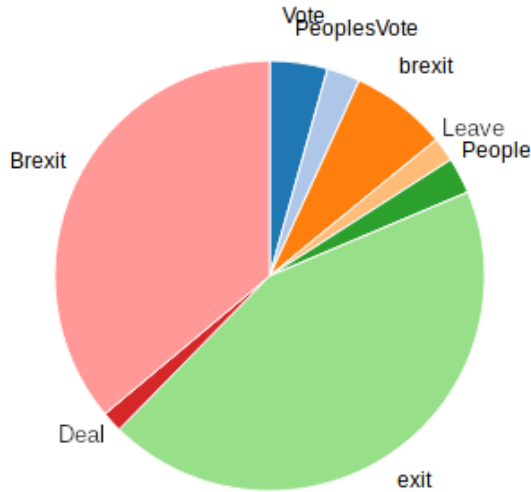


Figure 3. Hashtag Distribution in our data-set

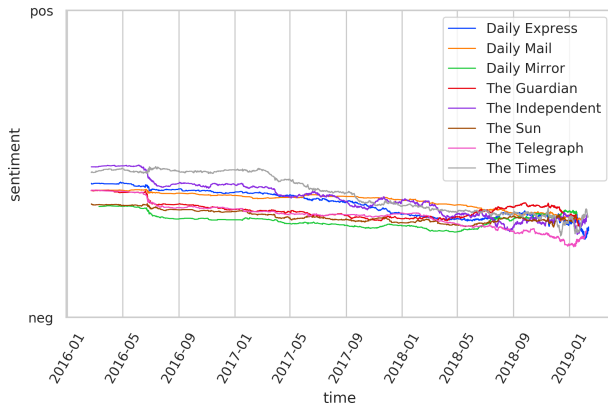


Figure 4. Follower sentiments of different newspapers

addition *Brexit* or *brexit* is a neutral keyword, that can be used for both sides. Note the difference between keywords and hashtags. A keyword is a regular expression, that can occur in the tweet text and in a hashtag as well, while a hashtag is restricted to the text after #. According to Figure 3, the hashtags #*exit*, #*Brexit*/*brexit* and #*Vote* dominated in our data.

Figure 4 shows the development of sentiments from 20 February 2016, the unofficial Brexit campaigning beginning in UK, of the different newspaper follower groups. Obviously a general negative trend of all follower groups is visible, whilst Daily Mirror follower tweets seems to be more negative than the others till

mid of 2018. In comparison the Times follower's tweets have the most positive value till the line drops in March 2017, when, by the way, the then-Prime Minister triggered Article 50.

### Biases

There are a multitude of biases which may have affected the outcome of our data and ultimately our interpretations. First of all, a visualisation is always an abstraction and can never show the reality as it is. Accepting this point, the data should represent the reality as accurately and objectively as possible. In our case, the only sources of data are tweets from specific newspaper followings without any background context, which in itself represents only a small part of the effect on the British people throughout the course of Brexit.

**Data Retrieval Bias.** Taking into account the project time-constraints and resource limitations, we were not able to completely crawl 100% of a given newspaper publishers following. This begs the question of whether our data is completely-representative of the respective newspaper publishers following? Though a large sample-size was taken for each newspaper publisher, (with equal duration in terms of server processing time) number of followers per newspaper, both in the preliminary filtering-process and the tweets remaining after the filtering-process varied. This may have led to disproportionate representation of the user-base of a newspaper's following and should be accounted-for in the interpretation of the visualisations.

**Filtering Bias.** Filtering bias may have occurred in our selection of Brexit-related keywords, we initially decided to restrict the keywords to those which directly relate to the Brexit referendum and the political spectrum of views leading up to the referendum. We decided to focus on the aforementioned keywords, mentioned in section Results. Though these keywords were relevant and used frequently in the run-up to the referendum, they were not relevant keywords throughout the entire duration of the Brexit time-frame as the political event evolved. Perhaps dynamically-changing the keywords filtered-for in the data-retrieval process may have solved this issue without adding additional noise to the parts of the data-set which were not associated with the time-specific relevant keywords, however - hand-picking keywords relevant



to the Brexit event as it unfolded would have required a tremendous amount of monitoring work to be done considering the Brexit time-frame was that of 2 years, we decided against this by considering the project as a feasibility analysis. Subsequently moving the need for a tool with event-specific dynamic keyword-filtering to further developments.

**Sentiment Bias.** A problem we encountered with the bias appearing in the sentiment analysis is the training data used for the Artificial Intelligence of the Stanford Natural Language Kit. The paper (9) released for the sentiment-annotator of the kit states that the neural network used was trained with data from movie reviews. One could remark that the language used to rate movies could be significantly different to the language used in tweets for political opinions and discussions on twitter. Hence, some of the tweets might not have been analysed reasonably.

Another bias problem for the sentiment analysis might also have something to do with the language used on twitter in relation to political discussions. The perceived sentiment of such discussions in social media often is often a negative one. This might have something to do with the psychological effect that people are more likely to speak up to word on political topics when they feel strongly about them. This effect can also be seen in our results, where about three quarters of the tweets were analysed negative or very negative. With regard to this effect positive reactions on Brexit events could perish between the disproportionately large amount of negative ones.

**Visualisation Bias.** In addition to the general bias throughout the data retrieval, filtering and pre-processing, one should not forget that the visualisation influences the perception of the information. First, indeed using the `rolling().mean()` from `pandas` enables us to improve the figure appearance and make it more understandable, but it cuts-off potential outliers, which might be important. That bias can be reduced by choosing a small window, in that the values will be averaged. Second, having unbalanced data makes it more difficult to compare. For example, Figure 5 shows a comparison between the follower groups of Daily Express and The Times. But our data of Daily Express only covers roughly nine days, even not the event itself, what makes

it hard to compare these two groups.

## Events

In the following we give three examples for sentiment curves around events for the two newspapers we gathered the most data for, which are The Times and The Daily Express. The observed events are: 1. The Referendum in 2016, 2. Theresa May wins the vote of confidence from her conservative Party. 3. the loss of the meaningful vote for Theresa May's Brexit Deal.

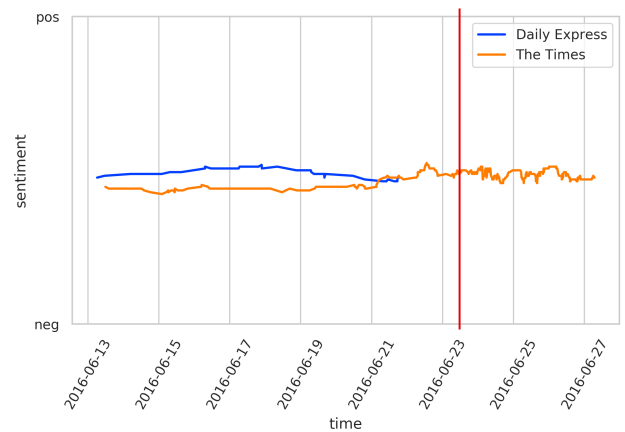


Figure 5. Referendum

Figure 5 contains the sentiments of Daily Express and Times followers around 23rd February 2016 when the referendum took place (red line). Unfortunately we had no data of Daily Express followers on the referendum date itself, which makes it hard to compare the two follower groups. However it should be accounted for, that The Times followers increased their tweeting frequency shortly before the referendum, which is visible by the increase in fluctuation of the graph. After roughly one day after the referendum the sentiment curve dropped. We only suppose that before-and-after the referendum the follower group of The Times was pretty active, but it is hard to say that there is a significant shift of sentiment.

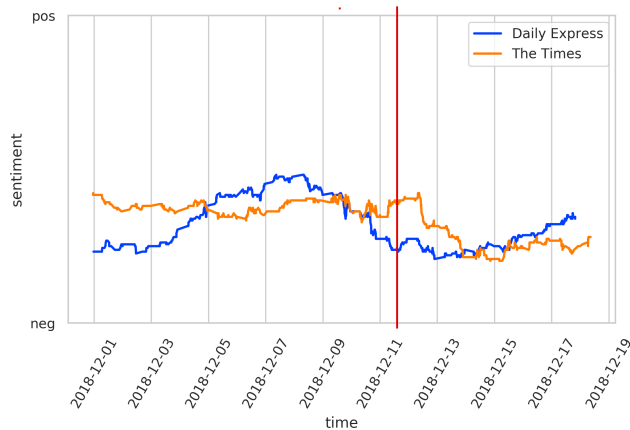


Figure 6. Theresa May wins vote of confidence

In December 2018 Theresa May won the confidence vote from within her Conservative Party. Figure 6 shows that Times' followers sentiment drops shortly after the event and also stays down, whereas Daily Express' followers mood seems to be negative even before the event and there can not be seen any significant change in that, it even rises a little.

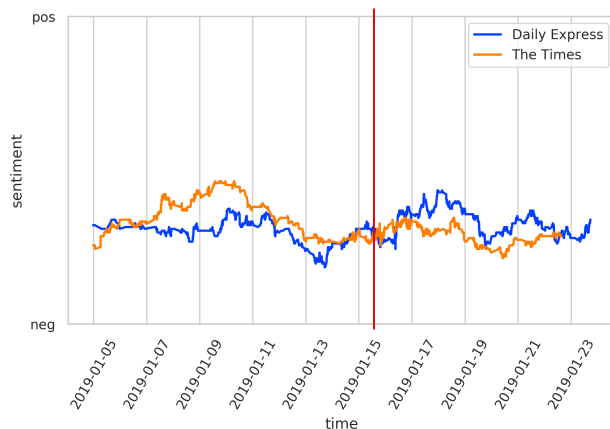


Figure 7. Theresa May loses Meaningful Vote

Figure 7 shows the sentiments of two newspapers before-and-after the *meaningful vote*, which was the parliaments dis-affirmation for the first rendition of the deal which Theresa May brokered with the European Union. Here can be seen, that the sentiments of the Daily Express followers as-well-as those of The Times

followers before the event show a downward spike, whereas after Theresa May's loss of the meaningful vote - The Times' followers sentiment curve stays quite stable and Daily Express' followers one increases shortly after. Of course we are speculating this result in its relation to the context of the event.

## Summary

### Evaluation

**Educational Objective.** According to the website (3) the primary goal of this course is to impart first steps dealing with data structures like XML or JSON in the usage of Python, R and Go. In the context of a micro-project the students should apply this knowledge to a chosen topic, where they can deepen the class content. In many aspects of our project we had to acquire experience in new areas. That required a deeper research in best practices and technologies dealing with APIs, data filtering and visualisation.

For the data retrieval, we had to improve our Java skills using `twitter4j` therewith we can send API-queries to Twitter. In the area of filtering and data pre-processing, because of the huge amount of data we collected, we had to learn about the time- and cost-efficient working with different methods and tools like the keyword-filtering and sentiment analysis using APIs or libraries of our chosen language, Python.

In terms of visualisation we made efforts selecting fitting charts for a specific purpose, working with different visualisation frameworks like `D3` or `seaborn` and data processing with `pandas` and `numpy`.

In summary we improved our skills in managing these kind of projects, evaluating fitting methods and technologies, handling big data-sets, and visualising aspects of the data-sets.

**Adherence to Twitter Developer-Guidelines.** Considering the project motivation, to provide a transparent overview of the landscape of opinions found on social-media platforms in light of the recent Cambridge Analytica scandal, it would be hypocritical not to have paid thought to the level of respect in which we used the twitter API and adherence to its guidelines. Concerning the privacy of users, as recommended in the Twitter API Restricted Use-Cases Guidelines, (11) we only displayed the user ID's of the individual users to respect the anonymity of sensitive user-data. Protected

accounts were not included post data-retrieval. Though regular users of the twitter API are not permitted to retrieve over 1.5 million `Tweet` IDs within a 30-day period, Use-Cases for academic institutions are granted unlimited access. However, though we were operating with educational intent - our academic institutional API key was not granted access in the time-frame of this project. Meaning we operated in a grey-zone under these premises.

## Further Development

### Data Retrieval

An interesting venture in the continued development of transparency-work and reports related to the data-analysis of macro-scale social-media sentiments might include the addition of other data-sets from different social-media platforms. Considering that micro-blogging takes place in various forms (and most importantly for various reasons) on different platforms, it would be useful to include them in the data-set to increase the level of representation of the sentiment of the community in the focus of the analysis.

### Pre-processing

To improve the data pre-processing portion of our project the first approach would be to target the problems stated in the Filtering Bias Section, e.g. by implementing dynamic filters. Another idea would be to optimise the sentiment analysis by using more accurate tools like IBM Watson, which was not feasible for us because of cost and time-efficiency. Perhaps even training our own neural network based on the Stanford NLP AI with training data more similar to political tweets could improve the result's quality. To decrease the human effort of data pre-processing and make our software more generic in comparison to the currently manually-initialised scripts, that do filtering, analysing and the database import could be joined and be integrated in the existing Django web-app. This way a specified JSON or CSV file including social media data could be uploaded to the app. A form e.g. with keywords to filter the data could be implemented and the app would be suitable for more generic purposes.

## Visualisation

The current setup could be improved in usability and performance. First, the visualisation selection, currently only roughly-implemented, could be extended by adding a selection of the desired chart-type, e.g. scatter chart or line chart. Also a form which the user can choose the data he/she wants to visualise can be added. Second, the performance can be improved. By using patterns like Singleton, the number of database queries can be reduced and the queried data can be used for several features, like visualising the data with D3 and exporting charts with `seaborn`.

## References

- Belaïd, A. (n.d.). *Django-NVD3 Documentation*. Retrieved 2019-01-10, from <https://django-nvd3.readthedocs.io/en/latest/>
- Bostock, M. (n.d.). *D3 Documentation*. Retrieved 2018-12-19, from <https://github.com/d3/d3/wiki>
- Humboldt Chair of Digital Humanities at the University of Leipzig. (2018). *DH.EDH*. Retrieved from <https://www.dh.uni-leipzig.de/wo/courses/winter-semester-20152016/introduction-to-digital-humanities/>
- ICO. (2018). *Democracy disrupted ?* (July). Retrieved from <https://ico.org.uk/media/action-weve-taken/2259369/democracy-disrupted-110718.pdf> doi: 10.9783/9780812290745
- Pydata.org. (n.d.). *Pandas Documentation*. Retrieved 2019-01-17, from <https://pandas.pydata.org/pandas-docs/stable/>
- RiteTag. (n.d.). *RiteTag: Find the best Hashtags*. Retrieved 2019-03-06, from <https://ritetag.com/>
- Smilli. (n.d.). *py-corenlp:Python wrapper for Stanford CoreNLP*. Retrieved 2019-03-06, from <https://github.com/smilli/py-corenlp>
- Smith, M. (2017). *How left or right-wing are Britain's newspapers?* Retrieved 2018-12-03, from <https://www.thetimes.co.uk/edition/news/how-left-or-right-wing-are-britain-s-newspapers-8vmlr27tm>
- Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*. *Statsmodels.org*. (n.d.). *ARIMA Python Documentation*. Retrieved 2019-02-16, from <https://docs.scipy.org/doc/numpy/reference/>
- Twitter. (n.d.-a). *More On Restricted Use-Cases*. Retrieved 2019-03-07, from <https://developer.twitter.com/en/developer-terms/more-on-restricted-use-cases>
- Twitter. (n.d.-b). *Twitter API Reference*. Retrieved 2019-03-07, from <https://developer.twitter.com/en/docs/api-reference-index.html>

Twitter. (n.d.-c). *Twitter Developer Agreement and Policy*. Retrieved 2019-03-07, from <https://developer.twitter.com/en/developer-terms/agreement-and-policy>

Walker, N. (2018). Brexit timeline: events leading to the UK's exit from the European Union - Briefing Paper. *House of Commons Library*(07960), 1–14. Retrieved from <http://researchbriefings.files.parliament.uk/documents/CBP-7960/CBP-7960.pdf>

Waskom, M. (n.d.). *Seaborn Documentation*. Retrieved 2019-03-07, from <https://seaborn.pydata.org/>




Wikipedia. (n.d.). *Campaigning in the 2016 European Referendum*. Retrieved 2019-03-07, from [https://en.wikipedia.org/wiki/Campaigning\\_in\\_the\\_2016\\_United\\_Kingdom\\_European\\_Union\\_membership\\_referendum](https://en.wikipedia.org/wiki/Campaigning_in_the_2016_United_Kingdom_European_Union_membership_referendum)

(DH.EDH18/19), held by Dr Thomas Koentges at the University of Leipzig. We hereby declare that we only used the literature listed in the citation section of this report - and that the three authors of this report had no help in the coding and the writing of this report, inclusive of the programming of the programs mentioned within it. All named-authors contributed equally in research, coding and writing.

---

### *Authors' Notes*

This project report and the project itself was part of the lecture *Introduction to Digital Humanities*

	Ned O'Hara	Felix Nirsberger	Tilmann Sager
	no13beki	fn63kuca	ts99nami
	3743371	3713674	3705682