Delft University of Technology

WI4231 Mathematical Data Science

Analysis of Donald Trump's Tweets

Project Report

Erik Jansson	4985370
Tuomas Koskinen	4989325
Ville Kujala	4991974
Emanuel Ravemyr	4989503
Federica Trevisan	4988078

April 11, 2019

Contents

1 Introduction

In this assignment, the tweets of the current President of the USA, Donald Trump will be analysed by use of clustering methods. In this case, the data set contains all the tweets from the year 2016, the election year, from Donald Trump's twitter account. These are easily accessible at `https://trumptwitterarchive.com/`[6]. The purpose of this assignment is to find interesting patterns and themes in the President's tweets using unsupervised learning methods, specifically clustering algorithms. The code of this project is fully available at the following Github repository: `https://github.com/federikovi/WI4231-MathematicalDataScience`

2 Data Understanding

The data set is composed of 4224 records and 7 attributes:

- *created_at* represents the timestamp in which the tweet has been published. It has been collected as metadata from the Twitter API in Greenwich Mean Time (GMT) and translated to Eastern Standard Time, 5 hours behind. This method used is imperfect because it doesn't take into account the location data, since not all Trump's tweets come with location.

- *favorite_count*, is the amount of likes received. This value ranges from zero for retweets (RT) to the maximum of 633253 for his 2016 most liked tweet, the first tweet after the election on 2016-11-09: "Such a beautiful and important evening! The forgotten man and woman will never be forgotten again. We will all come together as never before". The second most liked tweet with 573283 likes has been published one day before the election: "TODAY WE MAKE AMERICA GREAT AGAIN!".
  This feature has a mean of 20978 and a standard deviation of 25972. BO MAYBE MEAN AND STD ARE USELESS

- *id_str* represents the id given to that record, it has been discarded because not informative for the analysis.

- *is_retweet* takes value True if the tweet is a reposted message, False otherwise. The total number of RT in 2016 is 187.

- *retweet_count* represents the amount of times that the tweet gets reposted. The two most retweeted tweets are also the most liked followed by: "How long did it take your staff of 823 people to think that up–and where are your 33,000 emails that you deleted? https://t.co/gECLNtQizQ" in response to "Delete your account." from Hilary Clinton.

- *source* is an additional information describing the context of the tweet: 'Twitter for Android' 1836, 'Twitter for iPhone' 1956, 'Twitter Web Client' 340, 'Twitter Ads' 63, 'Periscope' 1, 'Twitter for iPad' 22, 'TweetDeck' 2, 'Media Studio' 1, 'Instagram' 2 and 'Mobile Web (M5)' 1.

- *text* the text of the tweet, is the most important feature for the analysis, it has been processed as described in the next section.

For the clustering algorithms, only the text column will be used. This is due to the fact that the content of the tweets are the feature of interest. Data such as favourites and retweets does not add or subtract to that feature but just adds some extra unrelated information, which could be considered to be mainly noise in terms of the interest feature. The different columns can then be used in the analysis part to add extra insight to the clusters.

3 Preprocessing

To make the clustering of tweets possible the dimensionality has to be lowered substantially. In the English language there are a lot of stopwords, such as "the" and "and", that appear very often and might make very different tweets look similar. Words also have what are called "stems" that make similar words look different. For example the words "sleeping" and "sleep" might look very dissimilar depending on what distance measure is used where in reality they are very alike. In addition to these, tweets often have punctuation, hashtags, links, mentions or even emojis that do not go well together with many distance measures and also do not affect the general theme of the tweet too much.

Python has a library called ekphrasis which contains a number of functions for analysing social media data. Ekphrasis performs tokenisation, word normalisation and word segmentation (for splitting hashtags), using word statistics from 2 big corpora (English Wikipedia, Twitter - 330mil English tweets). It was used to handle many unique properties of the Twitter language model reducing the feature space.

- **URL:s and email addresses.** All of the URL:s in the data are in shortened form so they do not tell anything about the content of the tweet. Email addresses usually also do not give any additional information on the content of the tweet. Therefore these are removed.

- **Numbers, including percentages, money, phone numbers and plain numbers.** Numbers by themselves do not give any information about the content of a tweet. They also make distance measuring a lot harder.
- **Times and dates.** The time and date of the tweets do not necessarily tell anything about the content of those tweets.
- **Mentions to usernames.** While it is true that some mentions tell something about the target or subject of the tweet, most of them are meaningless, especially when answering to someone else's tweet.

In addition to marking these, ekphrasis is also used to split hashtags into separate words, so for example #MakeAmericaGreatAgain becomes "Make America Great Again". This makes it possible to find the meaning contained in the hashtags.

After marking and splitting, the marked strings were removed from the tweets alongside with the aforementioned stopwords, punctuation and the most frequent words in the tweets. The list of stopwords can be found in [7]. In order to reduce the dimensionality even more, the words in the data were lemmatized, which is a more gentle version of stemming, removing suffixes from the words leaving only the root words. It was found that stemming gave very strange results due to being to aggressive so we felt that it would remove too many features, so instead lemmatization was used which gave the desired results.

4 Clustering

Clustering is a method which is based on grouping data in so called clusters, where the grouping is done with respect to some characteristic. These methods are performed directly through algorithms and does not require detailed instructions from the user about the clusters or what they should contain. This makes clustering algorithms so called "unsupervised learning" algorithms. There are three popular algorithms used for clustering that are going to be considered in this report. These are K-Means, DBSCAN and Hierarchical clustering. These are introduced below with a brief description of how they work and how they are used in the setting of this report.

4.1 K-Means

In K-means, the algorithm first initializes K centroids by selecting K points in the data set randomly. The remaining points are then assigned to the nearest centroid. This collection of points is the cluster in question. After this the centroid position is recalculated as an average of the coordinates of points belonging to the cluster. This procedure is then repeated either until convergence, meaning that the centroids do not change significantly, or or if the number of iterations is larger than some pre-decided value.

Before running the clustering algorithm, the preprocessed data was run through a tf-idf library function which transforms words to scores. tf-idf stands for "Term frequency - Inverse document frequency" and weighs the occurrence of a word in a data point, eg. a tweet with the occurrence of the datapoints that have that word. It then uses this weight as a score for the word, which can then be used for different comparisons. The scheme gives high scores for words that occur seldom and low scores for words that often appear in the data set. tf-idf is however not our preferred method since it does not take semantics into account. This means that synonyms will be treated as separate entities, despite being close in meaning. The problem with tf-idf is further exacerbated due to the length of the tweets being relatively short, which will mean that the resulting matrix will be sparse. It was decided to be the used method despite these issues, since for instance it was the preferred method in [2] and is a natural way of giving a numerical score to words.

Due to the time constraints and complexity of the methods, the implementation in the scikit-sklearn package was used. This means that the user input of interest, apart from the score matrix, is the number of iterations and the number of clusters. The maximum numbers of iterations can be set to some very high number, since this is method is comparatively fast.

There is no straightforward method for finding the value of parameter K, but different visualisations can be used in order to find a suitable value. One of these visualisations is the so-called elbow method. This method is based on running the K-means algorithm using various values of K, and plotting the sum of squared distances (SSE) as a function of K. SSE is computed by summing together the distance to the nearest centroid over all data points. SSE will naturally always decrease as the number of clusters increases. A suitable value of K can be found by looking for a point in the plot where the rate of change decreases substantially. However, sometimes no such point can be found. The SSE was computed for values of K ranging from 2 to 50, the resulting elbow plot is presented in figure 1. SSE seems to decrease at a constant rate with respect to K and no clear elbow can be seen in the plot. Thus additional analysis is required to decide the number of clusters.

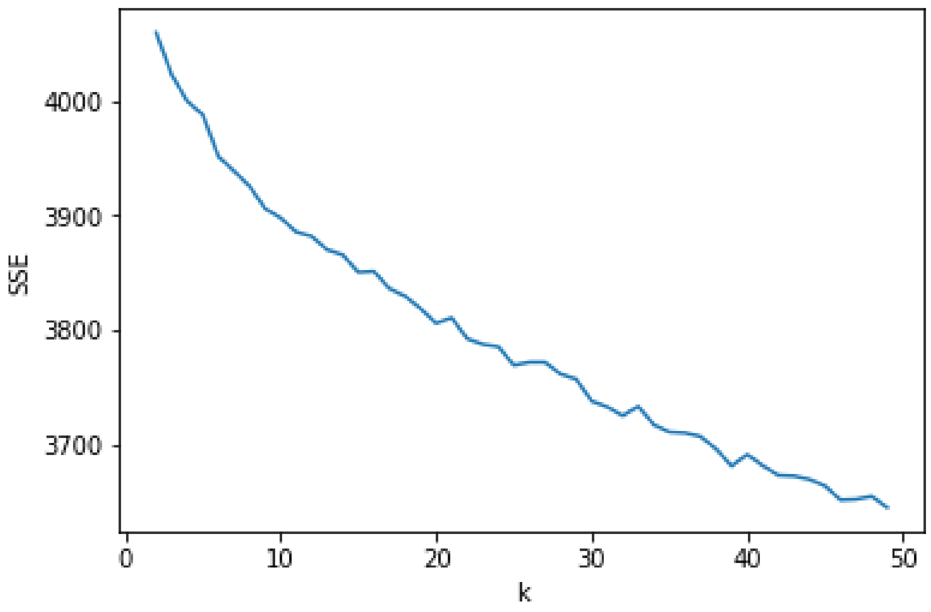


Figure 1: SSE plot.

Another method that can be used to find the value of K is the silhouette plot. The silhouette plot aims to visualise how large the distances within points in the same clusters are compared to the distances between points in different clusters. For each data point i the silhouette score is defined

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{1}$$

where a(i) is the average distance between point i and points in the same cluster, and b(i) is the smallest average distance to points in a different cluster. Thus the silhouette score of point i captures how well the point is separated from points in other clusters compared to points in the same cluster. The score is always between -1 and 1. A negative silhouette score indicates that the data point is actually closer to points in a different cluster than the cluster it was assigned to whereas a value close to 1 means the point is much farther away from any other cluster. The silhouette scores were computed from $K = 5, 10, 15, 20$, the resulting plots are presented in figure 2. Points within each cluster are sorted in descending order with respect to $s(i)$ and the clusters are sorted by $\max s(i)$. It can be noticed that none of these values result in a particularly good clustering and regardless of the value of K a big cluster emerges with negative silhouette scores which indicates that the cluster is noisy. Increasing the value of K beyond 10 does not seem to result in significantly better separation, instead multiple small cluster are created. Thus 10 seems like a reasonable value for K based on the silhouette analysis.

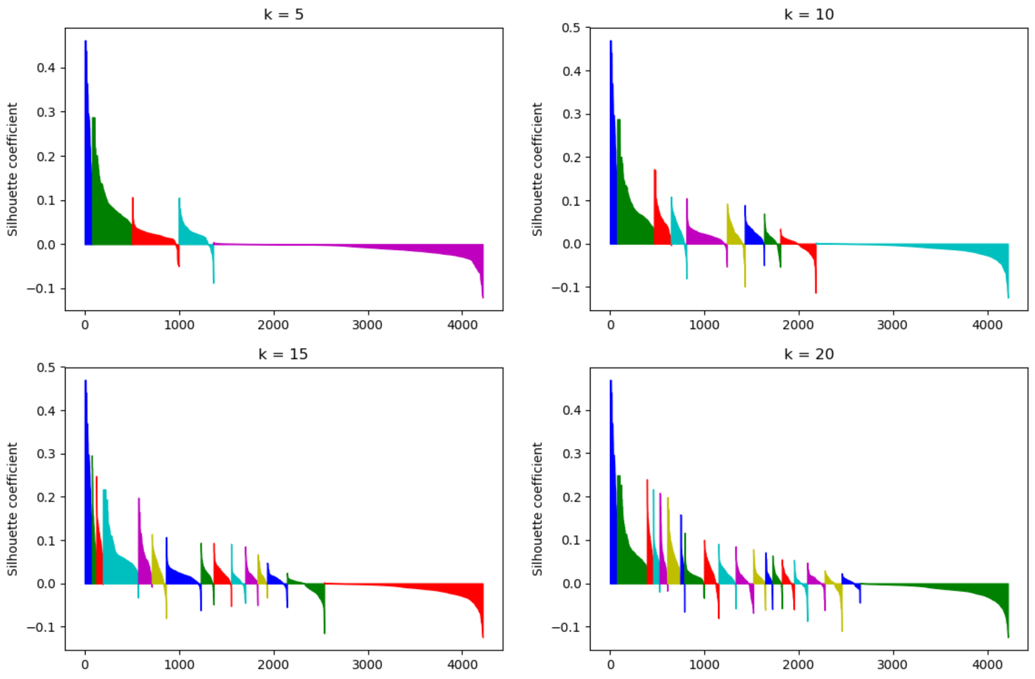


Figure 2: Silhouette plots for different values of K.

In addition to the elbow method and silhouette plot a clustering can be evaluated by using the similarity matrix. First the data points are sorted by their cluster number and then the pairwise euclidean distance matrix is computed. Assuming that data points are closer to points within the same cluster than points in other clusters, a diagonal block structure should emerge. The similarity matrix is given for $K = 5, 10, 15, 20$ in figure 3. Some diagonal block structure can be noticed within the smaller clusters but the largest cluster seems to be mostly noise. Both 5 and 10 seems to be reasonable choices for K based on this plot. Increasing the value to 15 or 20 results in many small clusters with no clear diagonal block structure.

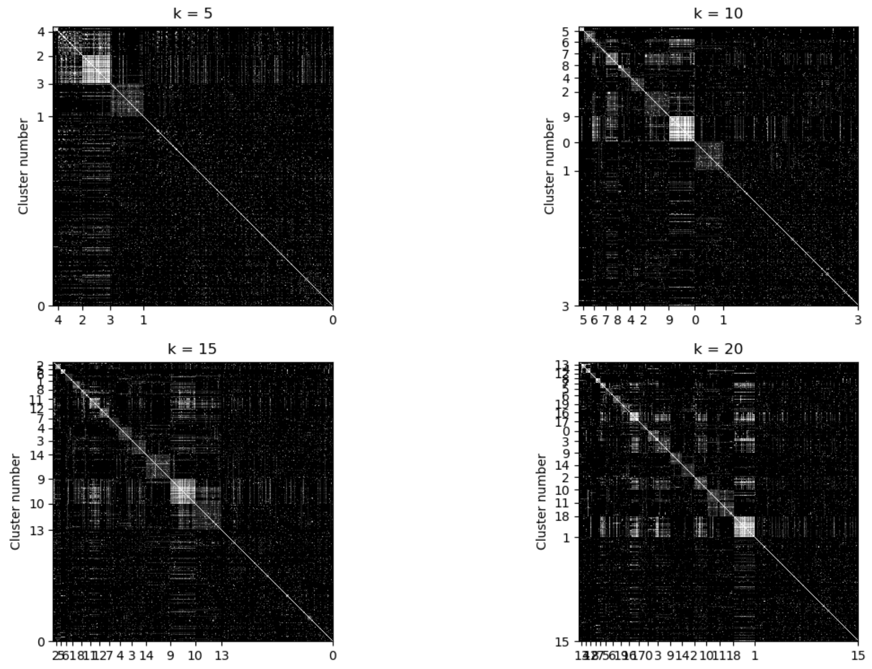


Figure 3: Similarity matrices for different values of K.

Based on the elbow method, silhouette plot and similarity matrix no value of K seems to be clearly better than others, however $K = 10$ seems to results in a slightly better clustering than $K = 5$ while still being a low enough number so that further qualitative analysis is viable, which is why it is chosen in favour of other values. The silhouette plot and similarity matrix for $K = 10$ are given in higher resolution in figures 4 and 5 respectively.

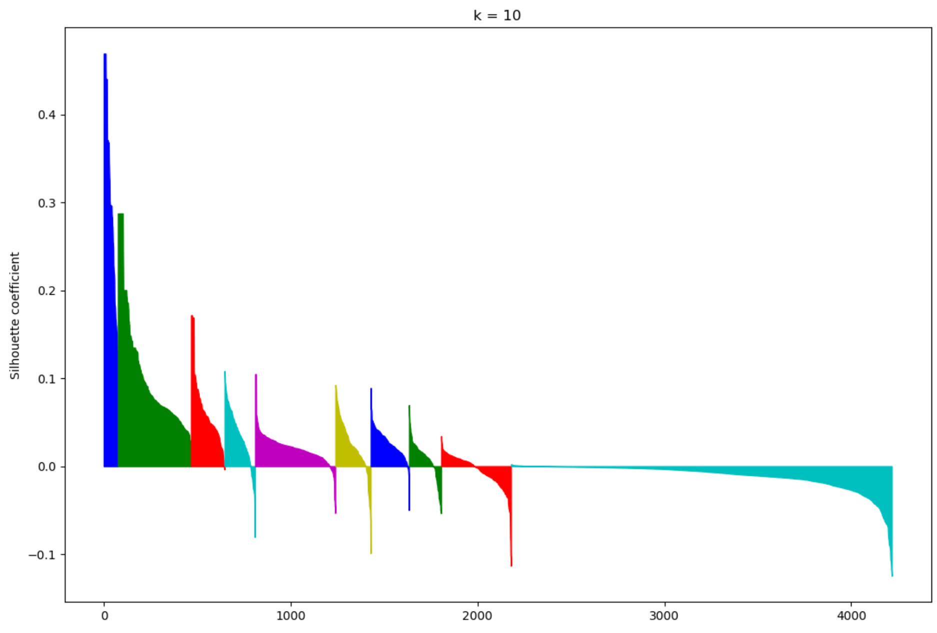


Figure 4: Silhouette plot for K=10.

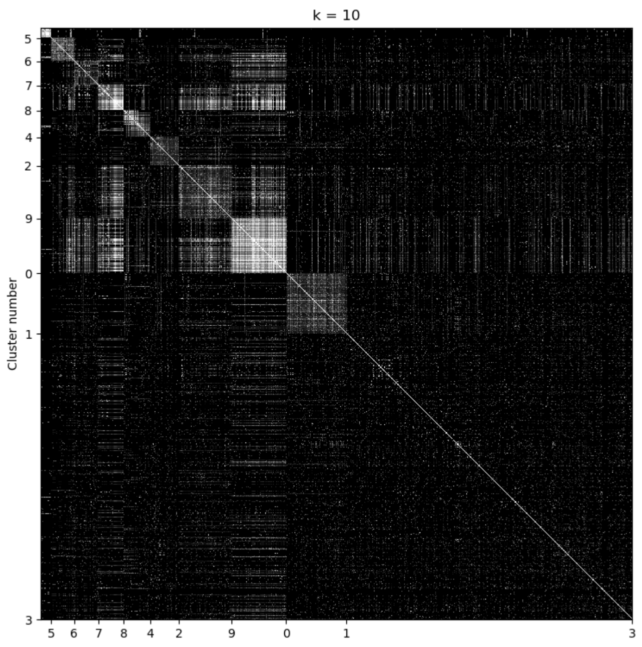


Figure 5: Similarity matrix for K=10.

4.2 DBSCAN

DBSCAN is a clustering method that requires a notion of density methods, which means that some way to measure density of tweets is needed. In general, this is done by introducing an $\epsilon > 0$ and some minimum density m and measuring densities within ϵ from the points in the set. The points are then classified based on their surrounding densities. Those with sufficiently high densities ($> m$) get connected and put into a single cluster. The algorithm removes outlier points that do not fit into any cluster, so-called noise.

However, in the setting of text analysis, this raises some issues, at least when tf-idf is used as a metric to deal with tweets in a numerical way, as the distance matrix becomes sparse. To combat this, a different distance metric could be used, Wordnet. This scores minimal distances between words using a synset (synonym set). Here the distances between tweets are measured as the average of the shortest paths, eg. the average of the shortest distance between every word pair in each tweet following the methodology in [3]. However, it turns out that the resulting matrix shows that everything is roughly equidistant in a similar manner as in the distance matrix resulting from tf-idf. This could be due to fact the processed tweets are very short. Furthermore, it is computationally demanding, and since it does not appear to improve the results, this method was discarded.

A challenge with DBSCAN is to make a good selection on m and ϵ. A too small ϵ will make all densities 1 while a too large one will make the densities m. Another issue is the selection of m. If m is chosen too large, then few groupings are qualified as clusters. However, if the value of m is too low, then even very small groupings qualify as clusters. The selection of these parameters is not only hard, but also very impactful on the result of the clustering.

4.3 Hierarchical clustering (Agglomerative)

In Hierarchical clustering all data points are initially assigned to their own clusters. The algorithm then merges the two clusters which are "closest" to each other. There are many different ways of defining this notion of cluster distance. One is to take the "mid point distance" which would be the distances between the average coordinates in each cluster. This is known as simple mean linkage clustering. Another, is to take as cluster distance as the largest distance of any two points in the clusters, which is called complete linkage. Furthermore, one could use single linkage in which the distance is the nearest neighbour. For the first iteration, this would purely be the distance between points. In each iterations clusters there is one merger of two clusters. A user can then select a desired level to stop at.

A main concern with Hierarchical clustering is the computational power that is required, since it is necessary to update the distance matrix in each iteration. This means that hierarchical clustering is not as fast a K-means, since K-means only need to calculate the centroid-distance

Before running the clustering algorithm the tf-idf score was computed of the preprocessed tweets. Then, complete linkage was selected over single- and average-linkage. This was due to we saw better results with complete linkage. This could be due to the fact that single-linkage gives rise to a chaining phenomenon, where close clusters in the single linkage sense could be forced together since some points could be very close to each other. In complete linkage, clusters could be preserved better, which is good since our data is very uniform. Regarding the number of clusters, $K = 10$ was once again selected as in K-Means. There are mainly two reasons for this. For one, if 10 clusters was an appropriate candidate for the same data with a different algorithm, it would not be strange to see the same number of clusters when only the algorithm is changed. The other reason is that 10 is a manageable number of clusters to analyse manually.

In figures 6 and 7 the silhouette plot and similarity matrix are given for the hierarchical clustering with $K = 10$. In the similarity matrix no diagonal blocks can be seen and the silhouette plot does not great either. The results given by K-means seem more promising.[4]

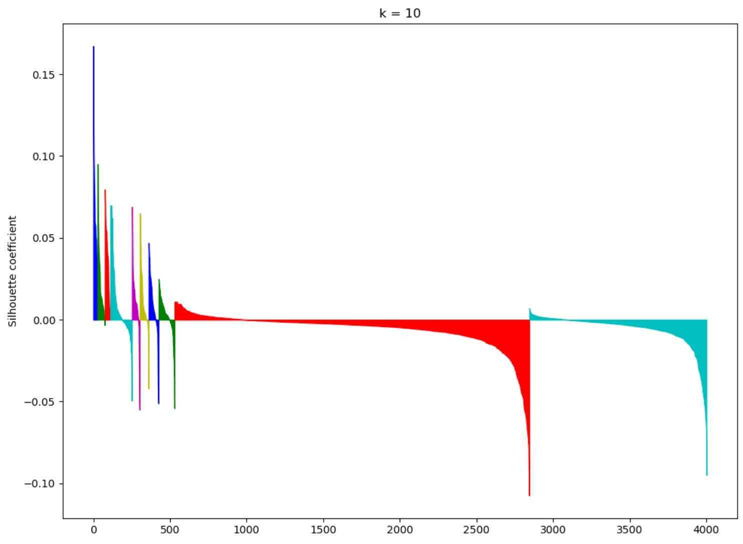


Figure 6: The silhouette plot for hierarchical clustering with 10 clusters.

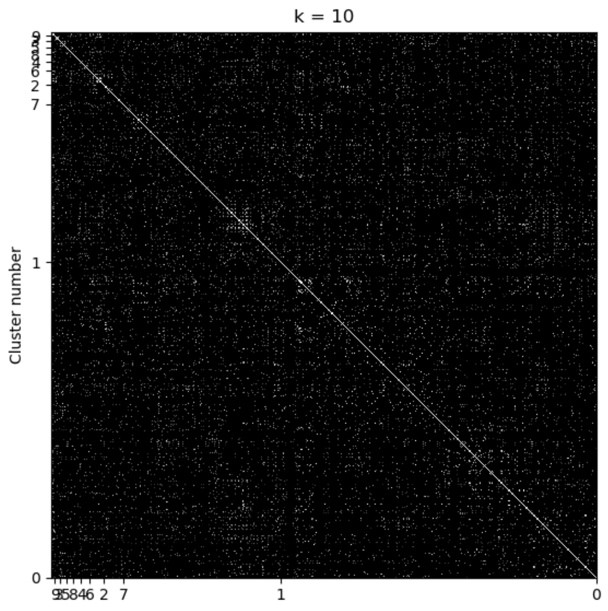


Figure 7: The similarity matrix for hierarchical clustering with 10 clusters.

5 Analysis

5.1 DBSCAN

During this project, some major issues were encountered with the DBSCAN algorithm. Even with use of different distance metrics, attempts ended in one of two scenarios. One of these is all data points being placed in one single cluster, the other is considering almost all data points as noise, returning an empty cluster, or many very small ones as a result. Obviously, this points to something going very wrong either with the algorithm, or with the data. In this case, it is probable that the data is simply a "poor match" for the algorithm, in the sense of a density notion. This has been touched upon when discussing the sparsity of data. Since almost all data is sparsely distributed and equidistant, the notion of density is mostly meaningless, as it will be distributed the same way in the feature space. DBSCAN excels at separating high density regions from low density regions which is not the case here, as we generally lack any high density regions. Further, DBSCAN is very sensitive to the input parameters, ϵ and m and it is simply not feasible in this project to try and find parameters which would yield good results, if any such values even exist in this case.

5.2 Hierarchical

The Hierarchical clustering gave some interesting results with 10 clusters, which gave two larger clusters and 8 smaller ones. A few of these contained a larger theme, but far from all of them did. One interesting feature is that the Hierarchical clustering was able to detect a "security" cluster, which contained tweets that regarded military, police and border security etc. This captures a larger feature in the tweets and is as such very interesting. This suggests that the algorithm is able to discover larger features in data, which is a very good thing. However, most other clusters were mostly very noisy and had no greater subject theme in them. This suggests that Hierarchical clustering may be a very good algorithm for several situations, but is not the best for this data, or that the parameter choice was a poor one.

5.3 K-Means

K-means will, for a huge number of iterations and $K = 10$ return a setting with one big cluster and several smaller, the big containing around 2000 tweets and the smaller ones containing tweets in the region from 50 to a few hundred. Since it is a randomised initialisation, the actual sizes will vary, but this is the general size distribution of clusters. In order to determine if the clustering is reasonably made, tweets are randomly sampled from each cluster. The big cluster will in general be seemingly without meaning, therefore the analysis will on the other clusters. Clusters that contain unrelated tweets will also be disregarded. The clusters are also not extremely stable, so some of them appear only on some runs of the clustering. Here are some examples of the relatively stable and meaningful clusters and the tweets in them.

Cluster number 0. This is the big cluster, containing as previously stated random types of tweets.

Cluster number 2. The biggest connecting factor in this cluster seems to be the word America. This might mean that the word should have been removed during preprocessing but the cluster seems to have a "Let's go, I'm with you!" - theme also.

- I'm with you! I will work hard and never let you down. Make America Great Again! https://t.co/ajAJ13xQvr
- Lets go America! Get out amp; #VoteTrump! #Trump2016 #MakeAmericaGreatAgain! #SuperTuesday https://t.co/w0eAglHeh8 https://t.co/rGanPzZHmS
- I'm with you! I will work hard and never let you down. Make America Great Again! https://t.co/V92PkdHbG6 https://t.co/SQs6ERk6El

Cluster number 3. This is a cluster of tweets relating to Hillary Clinton. It has a lot of noise in it as well, as all of the clusters tend to have, but many of the tweets talk about Hillary.

– What They Are Saying About realDonaldTrump's GREAT Debate and HillaryClinton's Bad Performance… https://t.co/1O86a49vJZ
– Wow, 30,000 e-mails were deleted by Crooked Hillary Clinton. She said they had to do with a wedding reception. Liar! How can she run?
– Hillary Clinton is unfit to be president. She has bad judgement, poor leadership skills and a very bad and destructive track record. Change!
– On my way to see the great people of Maine. Will be landing in Portland in 2 hours. Look forward to it! #Trump2016

Cluster number 6. The tweets in this cluster are generally about interviews. This is the most stable cluster, showing up almost every time the algorithm is run.

– I will be interviewed on the @TODAYshow at 7:00 A.M. this morning. Enjoy!
– On @FoxNews at 7:00 P.M. "Special: Meet the Trumps" Hope you enjoy!
– Melania will be interviewed by @morningmika on @Morning_Joe now (8:30 A.M.). ENJOY!
– Will be on @FallonTonight with @JimmyFallon on @NBC at 11:35pmE. Enjoy! #Trump2016 https://t.co/Z9zbKfXJqb

Cluster number 7. This is a clear Drain the swamp cluster, having almost only tweets with the hashtag DrainTheSwamp. There usually is one cluster that is based on a hashtags. Another one of these hashtags that form a cluster was AmericaFirst.

– Crooked @HillaryClinton's foundation is a CRIMINAL ENTERPRISE. Time to #DrainTheSwamp! https://t.co/89eOMTsIjt #BigLeagueTruth #Debate
– It is time to #DrainTheSwamp! https://t.co/U2XeM2vDJK
– Someone incorrectly stated that the phrase "DRAIN THE SWAMP" was no longer being used by me. Actually, we will always be trying to DTS.

Cluster number 8. This is a thank you and join me cluster. For some reason these tweets tend to get clumped together. There are often a couple different versions of the thank you cluster, one of them being noisy and the other having another common factor besides thank you.

- Thank you Louisiana! Get out amp; vote for John Kennedy tomorrow. Electing Kennedy will help enact our agenda on behal... https://t.co/sHXeyreEZI
- Join me tomorrow in Sanford or Tallahassee, Florida!
- Thank you! #AmericaFirst https://t.co/MYG45LxGmH
- Join me in Rome, NY- tomorrow! #Trump2016 #NYPrimary Tickets available: https://t.co/VGCdQubia1

Table 1 shows the sizes of the clusters as well as the means and standard deviations of likes and retweets of the tweets in them. It can clearly be seen that the one big cluster has about half of the tweets, which is suboptimal since nothing seems to connect these tweets together. It can also be seen that the standard deviations of likes and retweets compared to the means inside the clusters are so big that any extra information hardly can be derived from them.

Cluster	0	1	2	3	4	5	6	7	8	9
Size	2478	259	97	208	178	100	93	79	158	351
Mean likes	21041	20664	22902	25580	19139	24326	11630	23318	16855	23267
Std. likes	26487	21142	28851	25180	14148	28592	9399	12246	10596	10596
Mean rt.	7659	7011	8344	8633	6715	8088	2850	11415	6146	7864
Std. rt.	9412	6690	9496	7291	4605	7256	2084	5414	3836	8015

Table 1: Different statistics from the clusters

5.4 Analysis continued

It is possible to further the analysis in order to say more interesting things about the clusters. In order to do this, the clusters resulting from K-means where selected as a starting point, since those seemed to give the most coherent results. While Hierarchical discovered some interesting clusters that did not show in K-means, it generally failed to separate topics into separate clusters and mostly mixed up the tweets arbitrarily. This must be considered a major issue and was enough motivation not to pick it for the final analysis, as the results would have been unreliable. DBSCAN gave such poor results that it was never really considered for final analysis, as there were no real clusters to analyse.

In the K-Means clusters the meaningful clusters are fairly small in size and the larger clusters are mostly nonsense. This is not necessarily surprising, but it is perhaps slightly worrying. What it means is that many of the tweets does not contain any real meaning. Whether this is something that actually occurs, or just an effect of the clustering algorithms is an interesting question to ask ourselves.

5.5 Time

In the data pulled from the Trump twitter archive, there is a time label. It could very well be so that the meaningful clusters discovered by K-means have different time distributions, either over the course of the day or how the tweets in it are distributed over the year. Employing the datetime package it is possible to treat the datestrings into usable number and make a boxplot of each clusters distribution. For the distribution over the course of a day, see figure 9 and for the distribution over the course of the entire year, see 8. There is not actually that much interesting to see in the hours-distribution of the clusters, other than that the "Hillary cluster" as well as the "America"-cluster seems to be more skewed towards the evening, but given the wide spread, it is hardly a significant result. In order to further the analysis here, one could try to check from which platform it was tweeted, to see if these more polemical tweets tend to come from a specific platform. This might reveal additional information on the time distribution from the different systems from which the tweets were sent.

For the the distribution of the clusters over the entire year it can actually be seen that the "Drain the swamp"-cluster is distributed very narrowly in the end of the year, around the election. This was because the slogan was launched in October 2016[5], so this is actually not so strange, but it is good to see that the clustering algorithm has managed to catch this behaviour while disregarding time as a feature. Another thing that might, without much statistical significance just by qualitatively looking at the plots, is that the "thank-you cluster" seems to be skewed towards the end of the year, which also is not strange since this is from the final, intensive months of the election campaign.

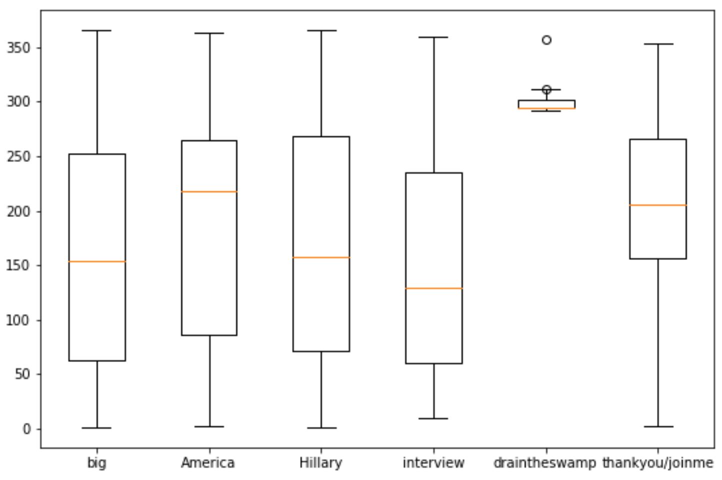


Figure 8: Boxplot for each interesting cluster for distribution over year

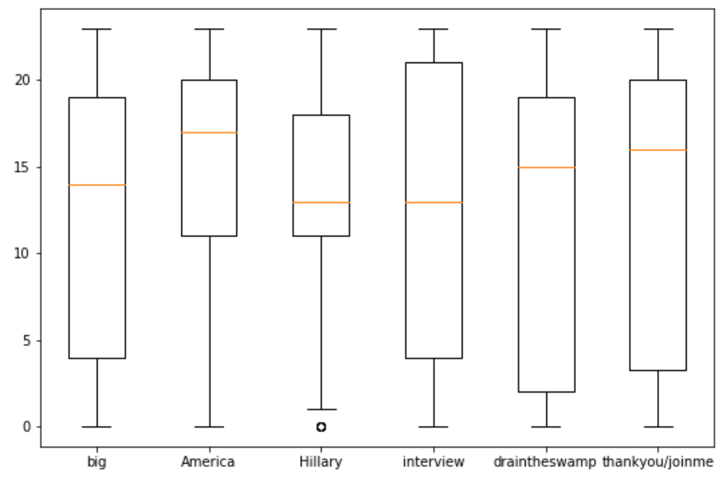


Figure 9: Boxplot for each interesting cluster for distribution over day

5.6 Sentiment Analysis

For a final analysis of the data set, the sentiment of the tweets was studied.

Sentiment is defined as a personal negative or positive feeling. In this particular case sentiment is a new feature of the data set obtained by the application of VADER (Valence Aware Dictionary for sEntiment Reasoning) unsupervised algorithm and is used for the validation of the best clustering algorithm.

VADER (Valence Aware Dictionary for sEntiment Reasoning) is a computational sentiment analysis engine that has good performances on social media texts. It is built on valence-based and human-curated gold-standard sentiment lexicon which has been attuned to microblog-like contexts. This lexicon, derived from other sentiment lexicons and produced using quantitative and qualitative methods, has been combined with grammatical and syntactical rules used by humans to emphasise sentiment intensity and for handling negations. VADER has been validated by humans: starting from a group of 20 human raters for sentiment of tweets, it has been proved that VADER perform as well as human raters [8]. When applied to tweets, VADER checks if the analysed tweet contains any of the words present in the sentiment lexicon and produces four metrics, Positive, Negative, Neutral and Compound: the first three represent the proportion of the text that falls into those categories; the last one is the sum of all lexicon ratings which have been standardised to range between -1 and 1.

The VADER algorithm has been applied, first at the *text* column of to the whole data set, see 10, then on the *text* column of each cluster found with k-means. As emerges in figure 10, the sentiment distribution of the tweets in the whole data set goes from -1 from negative tweets, takes a high value for neutral tweets (VADER score = 0) and end in +1 for positive ones.

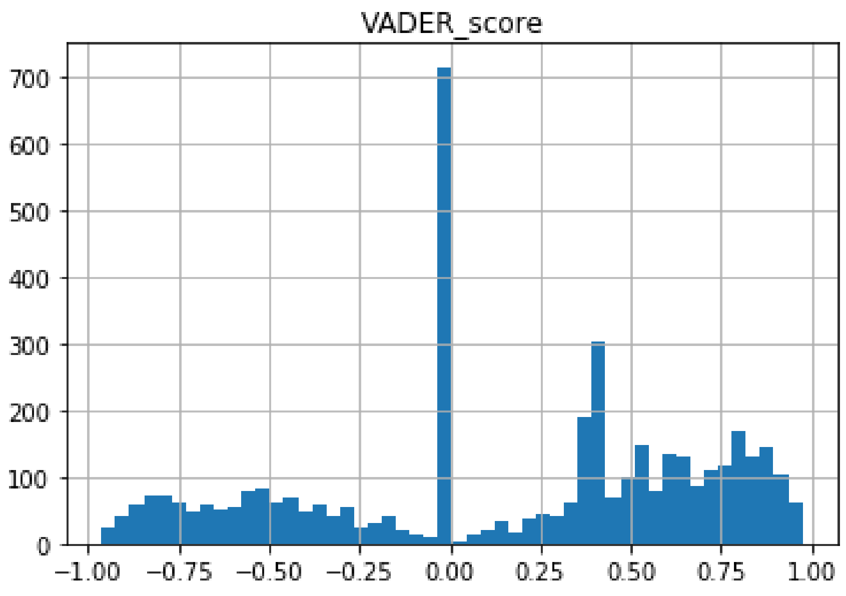
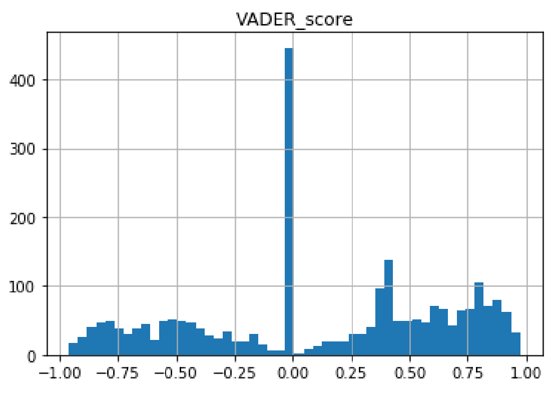
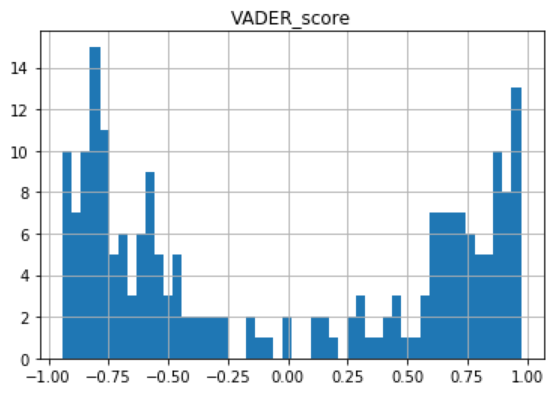


Figure 10: VADER scores distribution on full data set.

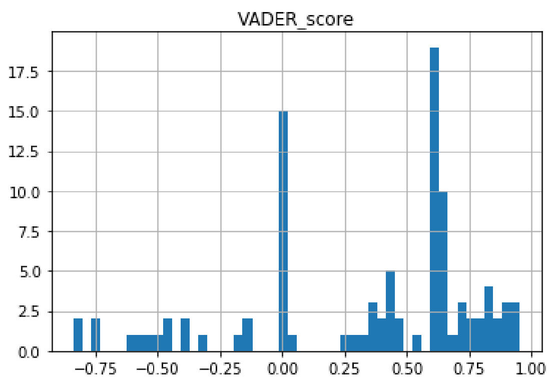
Analysing the best defined clusters found with K-means, it can be seen that the "big cluster" in Figure 11*a* has the same sentiment distribution as the sentiment on the whole data set. The "big cluster" has a size of 2478 tweets and contains mainly random tweets, so the sentiment distribution is not surprising. On the other hand, the Hillary cluster in Figure 11b has a complete different sentiment distribution: there are almost no neutral tweet detected and the polarity is either strongly positive or strongly negative. It contains a lot of noise, and it has been noticed that the negative tweets are almost all about Hillary ("Crooked Hillary Clinton overregulates, overtaxes and doesn't care about jobs. Most importantly, she suffers from plain old bad judgement!" with a VADER score of -0.8015), while the positive ones don't talk about Hillary much, they are the noise. From the polarity distribution of the America cluster (Figure 11c), interview cluster (Figure 11d) and thank you/join me cluster (Figure 11e) it emerges that almost all tweets are detected as positive.



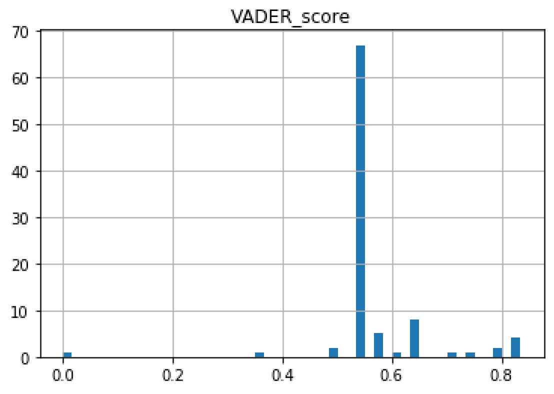
(a) Big cluster



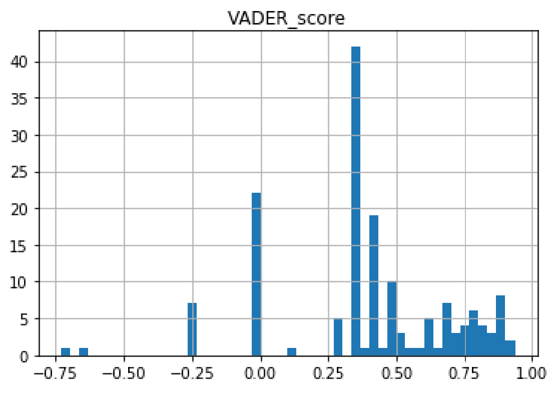
(b) Hillary cluster



(c) America cluster



(d) Interview cluster



(e) Thank you cluster

Figure 11: VADER scores for each cluster.

6 Conclusion

Clustering is a powerful method to find known and unknown patterns in large data sets, as well as grouping these together by feature. In this report, the different methods showed very different results, some of which were non conclusive. This suggests that clustering is not great for the data set in question. This may be due to the size of data (tweets are short), the size of the data set (around 4000 tweets in total) or something connected to the distance measures or some intrinsic property of the data points. Despite the issues, some significant results were found, which reveal some key characteristics of the way Trump tweeted throughout the year. This reveals that most of his communications is not the outrageous tweets that gets major attention in media, but rather normal public communication from a politician, with some additional flavor of controversy and blunt statements.

References

[1] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of machine learning research* 12.Oct (2011): 2825-2830.

[2] Dela Rosa, Kevin and Shah, Rushin and Lin, Bo and Gershman, Anatole and Frederking, Robert "Topical Clustering of Tweets" *3Rd Workshop on Social Web Search and Mining*, 2011

[3] Bogdani, "Compute sentence similarity using Wordnet", `https://nlpforhackers.io/wordnet-sentence-similarity/`

[4] Tan, Steinbach, Kumar, "Introduction to Data Mining" *Pearson Education Inc, 2006*

[5] Harrington, Rebecca "Here's what Trump means when he says 'drain the swamp'" `https://www.businessinsider.nl/what-does-drain-the-swamp-mean-was-dc-built-on-a-swamp-2016-11/?international=true&r=US`

[6] Trump Twitter Archive `"https://www.trumptwitterarchive.com/"Accessed2019-03-04`

[7] nltk stopwords `https://gist.github.com/sebleier/554280` Accessed 2019-03-14

[8] Gilbert, CJ Hutto Eric. "Vader: A parsimonious rule-based model for sentiment analysis of social media text." `http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf`

A Group assessment

As a preface to this assessment, it should be said that most of the work was done as a group, mostly through discussion on how to approach problems that were encountered. This means that all group members were heavily involved in most steps along the way to finishing this project, even if special tasks have been carried out by different members of the group.

Erik - Took part of the preprocessing and experiments related to those. Did early work with algorithm and wrote and edited parts of the report, also performed some analysis of the final results, mainly the time distributions of the clusters.

Tuomas - Took part in experimenting with different data preprocessing and clustering methods. Created the figures for clustering evaluation and writing related to them.

Ville - Came up with a lot of implemetation and analysis ideas. Did cluster validation by reading original tweets. Tried to make wordnet similarity to work as a distance measure. Wrote the section about preprocessing.

Emanuel - Researched theory and background on clustering algorithms. Wrote and edited large parts of the report. Did some early implementations of the algorithms for testing.

Federica - Completed the initial tasks for the preprocessing of the data set and did the sentiment analysis. Wrote the data understanding and sentiment analysis sections.