IT UNIVERSITY OF COPENHAGEN

MASTER THESIS

Detection of Appearance-Based Tweets Mentioning U.S. Politicians with fastText and VADER

Authors:
Jenny Nguyen
Tone Mie Pagh Rosenthal

Supervisor:
Michele Coscia

A thesis submitted in fulfillment of the requirements for the degree of MSc in Software Design

IT University of Copenhagen

June 20, 2023

Declaration of Authorship

We, Jenny Nguyen & Tone Mie Pagh Rosenthal, declare that this thesis titled, "Detection of Appearance-Based Tweets Mentioning U.S. Politicians with fastText and VADER" and the work presented in it are our own. We confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
- We have acknowledged all main sources of help.
- Where the thesis is based on work done by our self jointly with others, we have made clear exactly what was done by others and what we have contributed our self.

Signed:

Date:

Signed:

Date:

IT UNIVERSITY OF COPENHAGEN

Abstract

IT University of Copenhagen

MSc in Software Design

Detection of Appearance-Based Tweets Mentioning U.S. Politicians with fastText and VADER

by Jenny Nguyen
Tone Mie Pagh Rosenthal

This study investigates the difference in how female and male politicians are referred to in regard to their physical appearance on Twitter. The aim of the project is to use some prevalent models within Natural Language Processing (NLP) like Latent Dirichlet Allocation (LDA) to detect topics in tweets. Still, we find that for the purpose of seeing appearance, it requires a new approach. Therefore, we propose Scoring Appearance with fastText (SAFT) which gives a score for the presence of appearance-based language in tweets, and this score is used as an indicator of physical appearance in the tweets. Besides looking at differences in gender, we find it essential to expand the perspective of the study by investigating other groups like party identification and age group as well because these factors may influence the discourse of these public persons. Additionally, sentiment analysis with VADER is performed on the tweets with high appearance scores to discover if there are differences in the sentiment of the gender distributions. Results of this study show that the presence of appearance-based language is higher in tweets with female politicians than male politicians but male politicians receive more positive tweets than female politicians. This indicates that female politicians are mentioned most frequently for their appearance but in a less positive way compared to male politicians. In addition, the younger politicians have a higher degree of appearance-based language than the older ones but results are ambiguous for the political party.

Acknowledgements

Firstly, we would like to express our gratitude and appreciation to our thesis supervisor, Michele Cosica, for his guidance and support during our master's thesis. He has been thorough, available, flexible, and particularly skilled at communicating at our level of understanding. In addition, he has consistently shown concern for our well-being and provided encouragement and support during challenging times. We would also like to acknowledge his prompt and effective responsiveness to our emails, which has further facilitated our communication and progress. We are incredibly fortunate to have had him as our supervisor, and we extend our thanks for his invaluable contributions to our project.

Secondly, we would like to thank our study friend, Scarlett Avalon, for her assistance with this thesis. Her availability and willingness to help at all times with expertise in LaTeX layout skills. It has been crucial in shaping the presentation and visual appeal of our thesis.

Contents

1 Introduction

In the Western part of the world, many countries strive to pursue gender equity but in every country, there is nevertheless discrimination between the genders like women earning less than men for the same job. Gender discrimination is mostly hidden in a culture's norms and is difficult for insiders to spot. Feminism and perhaps more specifically the #MeToo movement have impacted many parts of society and have led to questioning aspects of our culture that were not problematized in the public debate before. One of them is how public persons are sexualized and how the twist of focus is from the persons' statements and opinions to their physical appearance. On social media, everyone can contribute to the public debate. It is possible to comment directly on a statement of a politician and people who would otherwise not be heard get a greater opportunity to share their opinions. Twitter is among the most popular social media platforms in the world and is widely used by politicians in the United States of America. Politicians use the platform strategically to brand themselves and communicate politics to their constituents. Messages on Twitter are called *tweets*. With the above-mentioned problem in mind, we specify the following research questions:

> *What is the difference in the distributions of appearance-based tweets related to male and female U.S. politicians? And of the appearance-based tweets, is there a difference in the sentiment of the distributions?*

For this project, we select eight American politicians and retrieve about $4M$ tweets to make the investigation. The aim is to use Natural Language Processing (NLP) to make an automated way of measuring gender-related differences in how American politicians are mentioned for their appearance on Twitter. To investigate tweets in relation to the research questions, we use tools in the field of NLP that enables machines to handle and analyze text data in human language. Besides the investigation of gender, it is also interesting to explore other factors that may influence the discourse of politicians. We accommodate this by expanding the investigation of factors like the party they represent and their age group.

If we use the NLP tool word embedding to find words association of tweets for the selected American politicians, we get the following most similar words:

Alexandra Ocasio-Cortez:	`['elonmusk', 'people', 'like', 'get', 'one']`
Kamala Harris:	`['joebiden', 'potus', 'people', 'like', 'speakerpelosi']`
Majorie Taylor Greene:	`['repmtg', 'twitter', 'catturd', 'donie', 'daveyalba']`
Lauren Boebert:	`['like', 'people', 'trump', 'know', 'one']`
Pete Buttigieg:	`['people', 'secretarypete', 'pete', 'like', 'get']`
Cory Booker:	`['kamalaharris', 'joebiden', 'realdonaldtrump', 'people', 'trump']`
Matt Gaetz:	`['michaelcohen', 'like', 'witness', 'trump', 'matt']`
Ted Cruz:	`['ted', 'trump', 'like', 'cruz', 'people']`

Pure NLP itself shows some interesting political connections without even needing too much text preprocessing and this is one of the tools we will use for answering the research questions in this project.

2 Related Work

According to Anspach and Carlson (2020), the main source for getting political news in the U.S. is social media and the popularity of using it is only rising. As news is posted on these platforms, it gives people the opportunity to comment and these comments may *"challenge or distort the information contained in [political news] articles"* because they work as a shortcut for information which can lead to misinterpreted news getting around.

In Cassese and Holman's (2008) research, they find that attacks and negative material are passed on to a large number of political campaigns both locally and on social media. If politicians intend to smear other politicians on these campaigns, it can contribute to setting the discourse on how people talk about politicians in general. The negative discourse can be spread among people who spread it further to their social network by reacting to these messages and news on social media (Anspach, 2017). People like to react to false news and novel stories which is the conclusion of a study at the Massachusetts Institute of Technology (MIT) where three scholars find that false news (especially political) is shared and spread around much faster than true stories on Twitter because people like new things and sharing news about these new things show that they are "in the know", but what is happening is just misinformation getting around and the ones responsible are those who share (Dizikes, 2018). Research of political communication on social media is therefore an interesting and widely studied field – also in the field of data science.

An older study detects offensive language in social media by proposing a solution called Lexical Syntactic Feature-based that constructs two dictionaries based on the level of offensiveness, assigning degree modifiers to words' offense levels, and then calculating a sentence's offensive score based on the word scores (Chen et al., 2012). Another study is detecting hate speech on social media by passing n-grams term-frequency inverse document frequency (TFIDF) to three machine learning models: Logistic Regression, Naïve Bayes, and Support Vector Machines, to classify the tweets into three classes (hateful, offensive and clean) (Gaydhani et al., 2018). Topic models are used to find topics in a document based on the words in it. They are powerful tools to identify implicit patterns and it seems that the generative statistical model Latent Dirichlet Allocation (LDA) is one of the most used methods to make unsupervised classification of topics. However, LDA has been criticized for its efficacy in analyzing data on social media as it is unsuitable for noisy and sparse datasets because of the missing features for statistical learning (Egger & Yu, 2022). Other researchers agree that it is more challenging to classify short texts (such as tweets) compared to document-level classification (Hong & Davison, 2010; Zirn et al., 2016). A reason for tweets being difficult is that these contain symbolic or other kinds of non-lexical information which make the contents grammatically incorrect.

In the research of Esposito et al. (2016), they compare the performance of LDA with running k-means on a dimensionality-deduced Word2Vec model. They find that when accurate linguistic text preprocessing has been carried out, the simple clustering algorithm on word embeddings performed as well as LDA. The same models (Word2Vec for creating word embeddings, Principal Component Analysis (PCA) for dimensionality reduction, and k-means for clustering) have been implemented successfully in a study about political polarization detection on social media (Adnana et al., 2021). In a comparison study, Beltran et al. (2021) take advantage of the machine learning model Lasso logistic regression to investigate gender differences in political communication on social media by classifying the words in the contents as male-linked or female-linked. In their results, the words most associated with male and female politicians suggest that people confirm stereotypes by treating politicians differently depending on their gender. *"We confirm*

the suspicion that the hostile talk addressed specifically at women is often related to criticisms of feminist positions and female MPs are more likely to receive words that are apparently positive but are in fact sexist as they relate to their physical appearance, along with condescending words that infantilize them" (p. 10). The fact that sexism is internalized and masked as positive comments indicate that it is not enough to search for sentiment alone or to just search for hate speech. Appearance-based statements are not necessarily explicitly hateful, sexist or racist but they can also be positive.

This project aims to categorize selected politicians by three groups: gender, political party and age group, and detect if tweets mentioning them are appearance-based by looking into the presence of appearance-based language. The baseline for the project is to prove and investigate gender differences which are similar to the study of Beltran et al. (2021) but we find it important to expand the perspective of the investigation by looking at groups like party and age as well since appearance is a broad and unsettled topic and other factors than gender may influence the discourse of public persons.
At first, we implement some of the known methods for topic modeling (LDA and clustering on word embeddings). As we do not get results that can answer our research question, we continue the work by creating an algorithm called Scoring Appearance with fastText (SAFT) which detects the presence of appearance-based language. Moreover, we conduct a sentiment analysis of the tweets with high appearance scores (tweets that are considered appearance-based) and compare the results of the male politicians with the female politicians to investigate if there is a pattern in the sentiment of the appearance-based comments each group of politicians receives.

3 Methodology

In this section, we describe two essential models used in this project: 1) a word embedding model (fastText) used to detect appearance in tweets, and 2) the sentiment analysis model (VADER). Other models/methods for failed or incomplete approaches to finding appearance in tweets are presented in section 6.

3.1 fastText

Proposed by Facebook in 2016, fastText is an extension to the supervised learning algorithm Word2Vec. Word2Vec is used to generate word embedding which is the process of turning words into vectors so they are represented in numerical form through a simple neural network. Word embeddings are often used in NLP as a way of representing words for text analysis. These representations as vectors have the words' meaning encoded in a way that in the vector space, words with similar meanings are close to each other. However, what makes fastText superior to Word2Vec is that it compares on n-grams instead of whole words. Both Word2Vec and fastText contain two methods for making word embeddings: Skip-Gram and Continuous Bag of Words (CBOW). These methods perform equally well but they work as opposites with Skip-Gram taking a word and predicting its context while CBOW uses the context to predict the next word (Rong, 2014). CBOW is faster and better at representing frequent words but Skip-Gram performs best on smaller datasets and can better represent less frequent words (İrsoy et al., 2020). The differences between the methods are not decisive for us because we train the model on a lot of the same data we use for tests. Nonetheless, we have specified it to run on Skip-Gram because it captures more fine-grained information compared to CBOW.

The idea behind word embeddings is that words appearing in similar contexts are closely related to each other. One of the characteristics of word embeddings is that their encoded semantic relationship can be found by adding and subtracting word vector representations. For example, the distance between the words "Hanoi" and "Vietnam" is the same as the distance between "Berlin" and "Germany" while the city capitals "Hanoi" and "Berlin" is represented close together and the country names "Vietnam" and "Germany" is represented close.

3.1.1 Training the Neural Network

The word embeddings are made by training a simple neural network with a single hidden layer to perform a task. The goal is to learn the weights of the hidden layer because they are representing the word vectors. This means that we are not interested in using the neural network on the task and the output layer (that represents probabilities for being the nearest neighbor to the target word) but only the weights.

The task is to compute the probability of adjacency for every word pair in a corpus. What is considered a nearby word is defined in the parameter `windowsize` which declares the number of words on each side of the input word that is being compared to. The output probabilities are how likely it is to find each vocabulary word nearby the input word. For example, the probabilities for the words `'man'` and `'king'` are higher than for `'woman'` and `'king'` because the number of times the word pair of `'king'` and `'man'` is represented is higher. The first step in training the fastText model is to build a vocabulary of word pairs because it can not take a plain string of words. An example of a word pair created from the input sentence *"The quick brown*

fox jumps over the lazy dog." If the input word is `'the'` and the window size is 2, the word pairs are `[(the, quick),(the, brown),]`. For the input word `'brown'` the word pairs are `[(brown, the),(brown, quick),(brown, fox),(brown, jumps)]`. When the model is trained on word pairs for every unique word in the vocabulary, the neural network will output a vector with the number of different features corresponding to the number of words in the hidden layer of the neural network. The neural network represents an input word as a one-hot vector (a combination of bits with a single 1 and the others are 0) with the number of components equal to the number of words in the vocabulary. The 1 will be placed in the position corresponding to the input word (McCormick, 2016; Rong, 2014).

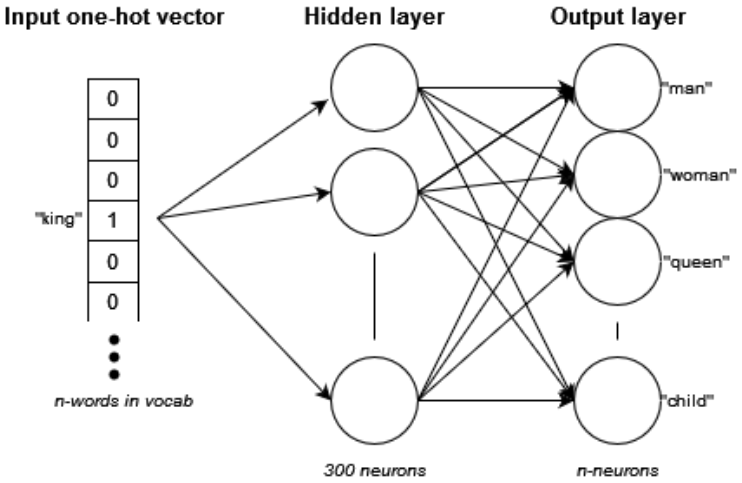


FIGURE 3.1: Illustration of the fastText neural network.

The model that is created from the neurons is called a *weight matrix* and works as a lookup table with each word in the vocabulary as a row and the neurons as columns.. This makes every row a word vector. The vectors are useful for our experiment to measure the semantic similarity between two words by calculating the cosine similarity between two-word vectors.

3.1.2 Similarity Between Words

The similarity between two words is measured with the cosine similarity which is the cosine of an angle between two vectors from the origin in the multidimensional vector space (for the fastText model the default is 100 dimensions). For example, if two words appear in a vector space with an angle of 45°, the cosine similarity is 0.71[1]. If the angle between the lines of the points in vector space is 0°, the cosine similarity is 1 which happens if the similarity is calculated for the same word[2]. In contrast, if two words do not have anything in common, the angle between two vectors will be 90° and the cosine similarity is 0[3]. It can be computed directly from the model matrix by using dot product and magnitude by the formula:

$$cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.1}$$

given two n-dimensional vectors of attributes, A and B where A_i and B_i are the i'th attribute of a neuron (or feature of a word) of word vectors **A** and **B** (Wikipedia, 2023e).

[1]The cosine of the angle is: $cos(45°) = 0.71$.
[2]The cosine of the angle is: $cos(0°) = 1$.
[3]The cosine of the angle is: $cos(90°) = 0$.

3.1.3 Character n-grams

The biggest challenge with Word2Vec is that it is not able to represent words that do not appear in the training dataset. On contrary, instead of feeding individual words into the neural network, fastText breaks words into several character n-grams. A character n-gram is the set of characters that occur together in a given window. For example, if $n = 2$ then the n-grams for the word `'that'` is `<t, th, ha, at, t>` (Wikipedia, 2023l). This means that for fastText a word is represented as a sum of its character n-grams. Basically, this means that fastText works on a character level, while Word2Vec works on a word level. Both models have the same goal of learning vector representations of words but one could argue that fastText is interested in the comparison of every single relation of characters within a word compared to the relation of characters within another word. Meanwhile, Word2Vec focuses on taking the whole word as an argument and comparing them to its neighbor words. There are two advantages of representing a word with n-grams. First, the model can learn morphologically rich languages better. For example, in Danish the words `'opvask'` and `'opvaskemaskine'` will be learned separately in plain Word2Vec but with fastText it is easier for the model to learn that the two words are related because they share overlapping n-grams, and so the words will appear close in vector space. Rare words have a higher chance of being represented properly by fastText since it is highly likely that some of their n-grams also appear in other words. Although it takes a longer time to train a fastText model because the number of n-grams is larger than the number of words, it performs better than Word2Vec in finding similarities between both morphologically related and rare words.

3.2 VADER

Sentiment analysis is a way of computationally quantifying and identifying the emotional state in a cohesive text. Examples of positive emotions are: `'love'`, `'happy'`, `'good'` and `'nice'`. Negative emotions are: `'sad'`, `'angry'`, `'bad'` and `'sick'`. As with many other tasks, it is important to choose the right sentiment analysis model for a specific purpose. Some models perform better at some tasks than others. For example, SentiWordNet assigns a polarity score on a scale from positive to negative to a sentence based on each word in it and it also assigns a subjectivity score on a scale from subjective to objective. This is based on WordNet (a large lexicon database) and has the advantage that it assigns a sentiment score based on word senses rather than on individual words which means that for every meaning a word has, there is a pre-computed sentiment score. However, this approach of assigning scores to senses is based on traditional linguistic definitions that do not work well on social media language because these texts often consist of sarcasm, irony, and new modern ways of using a word. SentiWordNet does not take the context of a word into account when assigning a sentiment score and for this reason, it makes it less suitable for our Twitter-based dataset (PrincetonUniversity, 2023).

3.2.1 VADER Specialized for Social Media Text

The limitation that a tweet can only consist of 280 characters means that users invented some techniques to expand the semantics of a message. For example, self-defined hashtags, starting with `'#'` to identify certain events or topics (Hong & Davison, 2010). Even with this limitation, the messages can still be rich in meaning. The sentiment analysis model Valence Aware Dictionary and sEntiment Reasoner (VADER) is "specifically attuned to sentiments expressed in social media" (Hutto & Gilbert, 2022) which makes the model ideal for this project. It is lexicon-based that uses a dictionary to map lexical features to emotional intensities which makes it faster than machine learning algorithms because it requires no training. The lexicon of VADER is suitable for short texts on social media because, on top of the vocabulary covered by the existing lexicons, it has added a list of emoticons (such as `':-)'` that denotes smiley-face and is usually used to indicate positive sentiment in the western world) (Wikipedia, 2023h), commonly used slang like

'lol and 'nope' and a list of acronyms like 'LOL' and 'WTF' (Hutto & Gilbert, 2014). Additionally, VADER's emotional intensities can catch grammatical and syntactical changes in sentiment intensity. The list of intensifiers is:

1. Punctuation (exclamation point, !) does not change the sentiment of a sentence but adding '!!!' at the end will make it more intense than '.'.

2. Capitalization is often used to intensify a specific word in a sentence as for example *"Your hat is AWESOME"*.

3. Degree modifiers can either increase or decrease the intensity of sentiment. For instance, given the sentence *"The service here is good"*, then *"The service here is extremely good"* will increase the sentiment while *"The service here is marginally good"* will reduce the sentiment intensity.

4. Conjunctions signify a shift in the sentiment of a sentence where the part after the conjunction is dominant. The sentence *"The sunset was beautiful, but I was freezing"* has mixed sentiments, but the last part is leading. (Hutto & Gilbert, 2014)

VADER tells how positive, negative or neutral a text is by giving four scores: positive, neutral, negative, and compound. The compound score or "normalized, weighted composite score" (Hutto & Gilbert, 2022) is the sum of scores for each word in the lexicon, modified according to the rules for the list of sentiment intensifiers. Although VADER can handle all of the above intensifiers too many degree modifiers will confuse the model and it will result in reducing the compound score. The compound score is normalized between -1 and 1, where -1 is extremely negative and 1 is extremely positive. This score is useful to set a threshold to classify sentences as positive, neutral, or negative. The other three scores are normalized ratios from 0 to 1 indicating how positive, neutral, and negative a sentence is. So, a sentence can have a different rating in these three scores. The result of VADER for a tweet will be neutral if the text is not referring to anything specific. An example of a sentence with a compound score of 0.0 (and neu: 1.0) is *"The service is exceptional and quality of food is very high"*. We use the compound score for sentiment analysis of all appearance-based tweets related to the politicians in the project.

3.3 Experiment Data Flow

The data flow of the project is illustrated in three steps in figure 3.2. The processes are automated so that the project can be iterative to improve the results from the analysis. After retrieving data with the Twitter API of both politicians and celebrities (as a part of the training set for fastText), we preprocess the text data for SAFT and VADER in section 5. The analysis step is described in section 6. Then fastText produces word embeddings from a combination of the preprocessed politicians-tweets and celebrities-tweets. The embeddings are used in SAFT which calculates the appearance scores of tweets and the results are visualized for all politicians and groupings (reasons for choosing the politicians can be read in section 4). Finally, sentiment analysis is conducted on the tweets with high appearance scores in the second part of section 6. Discussion of the results and approaches on how to improve our experiment and results is raised in section 7.

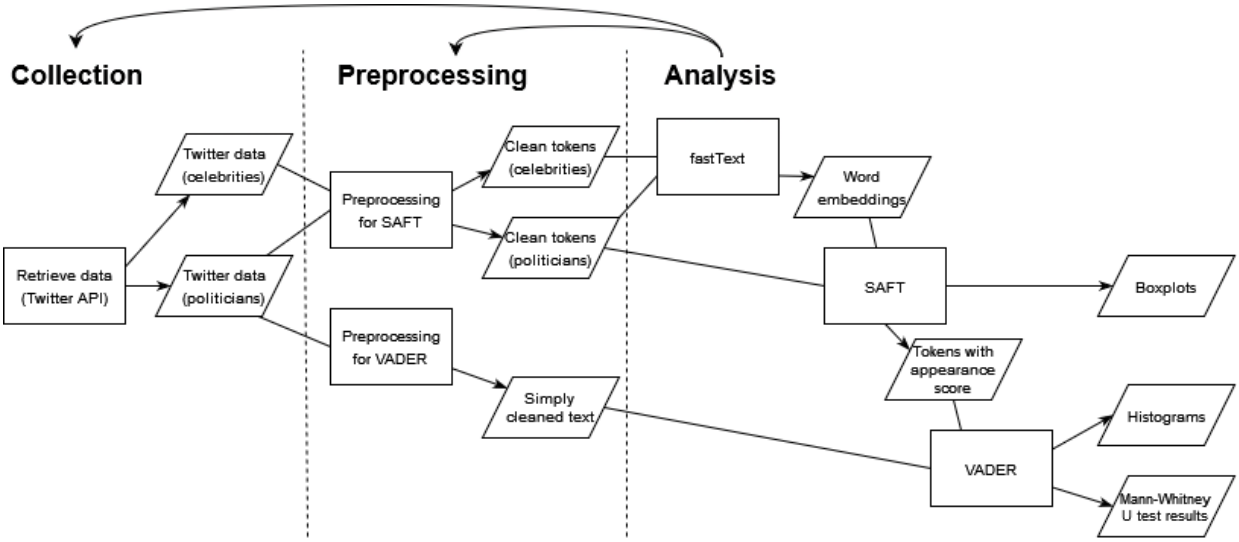


FIGURE 3.2: Chart of the overall experiment data flow.

4 The Dataset

4.1 Overview

With academic research access to the Twitter API, we use the tweepy library for Python to collect data for around $4M$ tweets that separately mention eight selected top politicians in the USA. The source code for this is in appendix A. As seen in figure 4.1 we define three groupings of politicians: gender, party, and age. Politicians are carefully chosen for this project to fit the distribution of the groupings.

	Female	**Male**	**Democrat**	**Republican**	**Age <45**	**Age >45**
Alexandria Ocasio-Cortez	x		x		x	
Kamala Harris	x		x			x
Lauren Boebert	x			x	x	
Majorie Taylor Greene	x			x		x
Pete Buttigieg		x	x		x	
Cory Booker		x	x			x
Matt Gaetz		x		x	x	
Ted Cruz		x		x		x

TABLE 4.1: Overview of politicians.

The collected data is saved in pickle files, each file consists of data for one politician with a time period of a month. Each politician has two accounts; one for personal use and one for work-related matters although, personal accounts are also used to publish very political content. The data is retrieved for the same time period for every politician's two accounts. With a total number of 53 files, some politicians got data that run over a few months and some several, which depends on how frequently a politician is being tweeted about. As a set criterion, we want a minimum number of $200K$ tweets for each politician.

When we retrieve the data, we do not know which information is actually needed besides the tweets and date-times of the tweets, so we choose to save more information than is intentionally needed in case it becomes useful. Therefore, the data consists of 12 columns which include information like the number of retweets, replies, likes, and quotes count. The tweet id and author information like id, username, followers, number of posted tweets, description, and location are also included. All this information can be used to make network analyses on users to find out if there are accounts responsible for a large part of the negative tweets, to investigate whether the tweets concerning a certain politician are also created in the state where their politics are most relevant or if a certain type of tweet receives more popularity. In order to be able to distinguish between the politicians' tweets when merging the 53 files, we add a column and annotated the politician's last name.

4.2 Criteria for the Dataset

As we are working with Twitter and politicians in one of the biggest nations in the world, we expect there will be a lot of political content. The U.S. also tends to be more polarized than smaller countries like Denmark, and the democratic system is binary with only two influential parties as they have the left–right political spectrum which makes it a good choice for conducting a solid foundation for comparative analysis. To make sure the dataset has some quality, we have set a few criteria for each politician to make them comparable.

1. **The politician needs to have at least $2M$ followers on one Twitter account (either on the personal or work account).** This is how we measure if a politician is popular enough to be widely mentioned by users on the platform who are not only their supporters. Of course, we do not know whether it is representative of both supporters and non-supporters, but we believe that the more well-known a politician is, the greater the likelihood that it is equally distributed. A lesser known politician will mostly be mentioned in tweets by supporters. Also, popular politicians get tweeted about more which is necessary as we need a great number of tweets to work with.

2. **The tweets are from a relatively neutral period in the politician's career.** It is important to retrieve data for the politician in a period where he/she is not in the middle of a scandal or similar because if this is the case, users on the platform will more likely tweet about this incident which makes the tweets biased and not comparable with politicians who have less controversies. Therefore, we aim for a somewhat neutral time in their political career. We also strive to avoid electoral periods because we believe the tweets will weigh towards the impulses of the political campaigns which are most likely negative (Cassese & Holman, 2018). Each politician meets this criteria at different time periods. Therefore, the dataset is composed of tweets distributed over approximately 3 years as seen in 4.2.

Politician	Time Period
Alexandria Ocasio-Cortez	April – May 2022
Kamala Harris	July – August 2022
Lauren Boebert	February – March 2022
Majorie Taylor Greene	February – March 2022
Pete Buttigieg	March – July 2022
Cory Booker	April – October 2020
Matt Gaetz	January – March 2019
Ted Cruz	January – February 2022

TABLE 4.2: Overview of the periods for the eight politicians.

3. **Minimum $200K$ tweets for the politician for their work account and personal account combined.** It is difficult to set a proper limit on how many tweets are enough to make the politicians comparable. Therefore, we choose to set it to $200K$ but less might also work. This is a great number for the fastText to have a good amount of sentences for training. Note that the number of tweets will decrease after preprocessing.

4. **If possible, we retrieve data for the newest period where the politician has the most recent political position.** We do this so that the content of the project has some contemporary relevance.

4.3 The Politicians

This section is an in-depth description of the selected politicians. Who they are and what is essential to know about them. Each politician has a table with information related to the retrieved data. Note that the age and political position stated in the tables are at the time of the period of the data. The tweets are retrieved from different periods and lengths because it varies how much activity and popularity they have on the platform. This means that for some of the politicians, there is enough content in a shorter time period, while for others, it needs to be over a longer period to get the required number of tweets over $200K$. The number of tweets is both from the politicians' personal and work accounts.

4.3.1 Alexandria Ocasio-Cortez

Twitter accounts:	@AOC, @RepAOC
Twitter followers:	13.4M, 774K (as of April 2023)
About the retrieved data	
Total number of tweets:	588k
Period:	April – May 2022
Political position:	Member of the House of Representatives from New York's 14th district
State:	New York
Age:	32

Note: Information and picture are from Wikipedia, 2023a.

Alexandria Ocasio-Cortez (also known as AOC) is an activistic far-left democrat. She is a part of "The Squad" which is a progressive group of democrats in the House of Representatives[1]. She has the record of being the youngest woman elected to Congress. Her first term started in 2018 when she was 29 years old. In 2011, she got her bachelor's degree in international relations and economics from Boston University. During her college time, she worked as an intern for Ted Kennedy (former senator from Massachusetts). Prior to her position as House representative, she worked as a bartender, waitress, organizer for Bernie Sanders presidential campaign, and as a public activistic speaker for healthcare and environmentalism. Ocasio-Cortez is a part of the political organization Democratic Socialists of America and promotes the idea that the government should help the poor and criminals by making education affordable, cheaper housing, and healthcare for all. She wants to grant U.S. citizenship to all immigrants who contribute to community-supporting work. In addition, she advocates for a public economic policy to address climate change, and women's and pro-LGBT+ rights (Ocasio-Cortez, 2023; Wikipedia, 2023a).

We choose Ocasio-Cortez because she is skilled at using social media to spread her politics and has managed to create a brand around herself and the abbreviation AOC. Fox News and Fox Business (pro-Republican media) have mentioned Ocasio-Cortez on average 75 times per day from February 25 to April 2019 proving that she is a central political figure for both left a right-wing Americans. Fox portrays her as a radical socialist. Therefore, we expect people from both political wings to express their opinions about her on Twitter. She creates a lot of attention around herself in the media by doing activist actions, such as at the Met Gala 2021, when she was dressed in a white dress with the clear statement "Tax the rich". In January 2022, she appeared in a film about COVID-19 which we believe is a time with a lot of attention (Wikipedia, 2023a), therefore we avoid this period and choose to retrieve data for April and May.

[1]The US Congress consists of two chambers: the House of Representatives and the Senate. The Senate is the upper chamber which is smaller and has more restricted power than the lower chamber. The Senate consists of 100 seats with two representing senators of each state, serving a six-year term (Wikipedia, 2023u). In the House, one representative from each congressional of the 435 districts occupies a seat. The districts are distributed over the states based on the population of the state where a state is entitled to at least one district (Wikipedia, 2023t).

4.3.2 Kamala Harris

Twitter accounts:	@KamalaHarris, @VP
Twitter followers:	20M, 14.1M (as of April 2023)
About the retrieved data	
Total number of tweets:	554k
Period:	July – August 2022
Political position:	The 49th Vice President of the U.S.
State:	California
Age:	57

Note: Information and picture are from LosAngelesSentinel, 2018; Wikipedia, 2023f.

Kamala Devi Harris is part of the Democratic Party and is the first female to be Vice President of the U.S. She is also the first African-American and Asian-American vice president. Previously, from 2011 to 2017 she was the attorney general of California[2], and in the years from 2017 to 2021 she was a U.S. senator representing California. Harris holds a degree in political science and economics from Howard University and a Juris Doctor (degree for practicing law). Harris is an advocate for single-payer healthcare, legalizing cannabis on the federal level, citizenship for undocumented immigrants, abortion and LGBTQ+ and women's rights. She is against assault weapons, the death penalty, and tax cuts (Wikipedia, 2023f, 2023o). Harris has been rated as the most liberal U.S. Senator by the non-partisan website GovTrack, but left-wing activists criticize that some of her past actions as a prosecutor were acted in favor of right-wing. Harris was a candidate in the 2020 Democratic presidential election but withdrew from the race which can be impacted by her right-wing past. What is seen as right-winged actions was that she threatened to put the parents of truant children in prison who were poor and non-white and she was against a proposed parole program about releasing prisoners earlier if they had served half of their sentences in which she argued that "prisons would lose an important labor pool" (Intercept, 2019).

We choose Harris because she is an influential person who has made recognized changes for the world as she has been on the Times 100 list several times (Wikipedia, 2023r). Therefore, she is a popular figure, which we believe makes her a suitable politician for this project. As she occupied her current position in 2021 and we believe there might be more tweets about her occupation around that time than what is the general picture of the everyday tweets, therefore we give it some time and retrieve data from the middle of 2022.

[2] A state attorney general ensures that laws of the state are enforced uniformly and adequately and carries the responsibilities of the whole office (Wikipedia, 2023c).

4.3.3 Lauren Boebert

Twitter accounts:	@laurenboebert, @RepBoebert
Twitter followers:	2.3M, 900K (as of April 2023)
About the retrieved data	
Total number of tweets:	734k
Period:	February – March 2022
Political position:	Member of the House of Representatives from Colorados's 3rd District
State:	Colorado
Age:	35

Note: Information and picture are from Wikipedia, 2023g.

Lauren Opal Boebert is a young far-right Republican. She never held public office prior to her congress election. She earned her high school diploma a month before her primary election in 2020. Prior to that, she worked as an assistant manager at McDonald's, natural gas product technician, and pipeline integrity coordinator, and she owned the gun-themed restaurant Shooters Grill, among others, in Colorado where staff carried guns at work. She is a member of the congressional caucus House Freedom Caucus which is considered the most conservative and farthest-right bloc within the Republican Party of Representatives. Boebert is a Christian and promotes the idea of Christian nationalism, which means that she is against the separation of state and church. She is against abortion, LGBTQ+ rights, tax increases, and sustainable energy initiatives. On the other hand, she is a strong advocate for gun rights, the Mexico-U.S. border wall, and a closer relationship between Israel and USA. Right before she won her first primary election in 2020, she was associated with several far-right groups and she has also embraced the political conspiracy movement QAnon[3] (Boebert, 2023; Wikipedia, 2023g).

We choose Boebert because she is young, right-wing and female. The tweets regarding Boebert are from a time when she was serving her first term as a congresswoman and was at the beginning of her re-election campaign. Usually retrieving tweets from an election period indicates that there potentially can be a bias in the dataset. This is because the subjects regarding her can be skewed toward the campaign and the political agenda she is working towards. But for Boebert as a young politician only in her now (2023) second period in the House of Representatives, we consider it preferably that it is a normal and relatively calm period in her career. It is, in a way, always election season for politicians because it always is a priority to communicate their political message clearly and look good in their eyes. Because Boebert was publically speaking of Twitter as being propaganda in January 2022 (Wikipedia, 2023g), the first month of the dataset to be February. By this time she is well known and many people have an opinion about her because she has been in Congress for almost a full period. This means that we can expect the tweets to not only be created by her supporters.

[3]QAnon is an American far-right political movement and conspiracy theory at its core about a group of satanic and cannibalistic pedophiles that are operating a global child sex trafficking. This group is against Donald Trump and the movement started during Trump's term in office. Among others, Democratic politicians have been called members of this group (Wikipedia, 2023p).

4.3.4 Majorie Taylor Greene

Twitter accounts:	@mtgreenee, @RepMTG
Twitter followers:	696K, 2.1M (as of April 2023)
About the retrieved data	
Total number of tweets:	612k
Period:	January – March 2022
Political position:	Member of the House Representatives from Georgia's 14th District
State:	Georgia
Age:	42

Note: Information and picture are from Wikipedia, 2023i.

Majorie Taylor Greene (also known as MTG) is a far-right Republican and conspiracy theorist. She has served as a Representative for Georgia's 14th congressional district since 2021. She supports Donald Trump and she was endorsed by Trump and Matt Gaetz for her candidacy. Like Lauren Boebert, she is also a member of the Freedom Caucus and identifies herself as a Christian nationalist. Greene holds a bachelor's degree in business administration from the University of Georgia. Before she became engaged in politics, she worked as a coach and co-founded CrossFit Passion. She got into politics in 2016, and she wrote articles for a conspiracy news website and a pro-police fake news website in the following years. Greene promotes antisemitic, white supremacists, securing borders, conspiracy theories like QAnon and government involvement in mass shootings, and she praised the Russian president Vladimir Putin during the Russian invasion of Ukraine in 2022. She is against gun control, abortion, LGBTQ+ rights, and rejects climate change.

There have been several public incidents with Greene and Alexandra Ocasio-Cortez. One of the more recent was in May 2021 when Greene loudly asked Ocasio-Cortez why she supported Antifa (left-wing anti-fascist and anti-racist political movement) and Black Lives Matter which Greene marked as terrorists (Wikipedia, 2023b, 2023i). Greene uses Twitter frequently to share her view of things and she has several times been time-limited suspended from Twitter for sharing misinformation like in regards to COVID-19 and vaccines which she is against (Wikipedia, 2023s).

We choose Greene because she is popular, active on Twitter, and matches the age group of politicians over 45 years old. When choosing politicians, we want to include those that are extreme in their beliefs and not bi-partisan to make the picture as clear as possible between the ones far-most to the right (Republicans) and the ones far-most to the left (Democrats), and Greene is one that cannot be doubted in her belief. The period of data retrieval for Greene is during her reelection campaign. We believe that it is difficult to find a recent time when Greene has not been in the media. In the years before, she did campaign for her election, so we cannot avoid a time of her not doing a campaign unless we go back even further.

4.3.5 Pete Buttigieg

Twitter accounts:	@PeteButtigieg, @SecretaryPete
Twitter followers:	3.7M, 530K (as of April 2023)
About the retrieved data	
Total number of tweets:	273k
Period:	March – July 2022
Political position:	The 19th Secretary of Transportation in the U.S.
State:	Indiana
Age:	40

Note: Information and picture are from Wikipedia, 2023n.

Peter Paul Montgomery Buttigieg is part of the Democratic Party and has served as the 19th U.S. Secretary of Transportation since 2021. Previously, he was mayor of Indiana for two terms from 2012 to 2020 where he also did military service in Afghanistan. He graduated from Harvard College with a degree in history and literature and the University of Oxford with a degree in philosophy, politics, and economics in 2004 and 2007, respectively. He ran for president in 2020 and has been one of the first openly homosexual men to launch a major presidential campaign. As a top-tier candidate, he chose to drop out of the race because he believed there were better-suited candidates (TheNewYorkTimes, 2020). After that, he endorsed Joe Biden. Buttigieg was Biden's nominee for Secretary of Transportation after Biden secured the presidential post. The nomination was confirmed in 2021. Of political positions, he supports abortion, LGBTQ+ rights, healthcare, and climate change. Buttigieg came out in 2015. Later that year, he got re-elected as mayor for the coming term (Campaign, 2021). As Secretary of Transportation, Buttigieg addressed the African American Mayors Association in 2021 where he argued about how misguided investments in the federal transport and infrastructure policy have been a contributing factor to racial inequity. Racial inequity has been a topic that Buttigieg had mentioned in many interviews because he found it related to his work at the department (Wikipedia, 2023n).

We choose Buttigieg because he is: 1. well-known, 2. young (under 45 years old), 3. a Democrat, and 4. active on Twitter. It is difficult to find any other politicians that pass these four criteria. Therefore, we choose Buttigieg but his sexual orientation can make him a biased choice of a politician as we expect homosexual men will be talked more about their appearance than heterosexual men. The period data for Buttigieg should be a period between 2021 and now. We think that people would talk more about Buttigieg right at the beginning of his occupation of the current position than after a while when it is more settled. This is why we do not use tweets from beginning of 2021. In August 2021, Buttigieg was on paternity leave for two months with pay for which he got criticized (ABCNews, 2021). Therefore, we gave give it time and chose to use data in the middle of 2022.

4.3.6 Cory Booker

Twitter accounts:	@CoryBooker, @SenBooker
Twitter followers:	4.1M, 200K (as of April 2023)
About the retrieved data	
Total number of tweets:	280k
Period:	April – October 2020
Political position:	U.S Senator from New Jersey
State:	New Jersey
Age:	50

Note: Information and picture are from Wikipedia, 2023d.

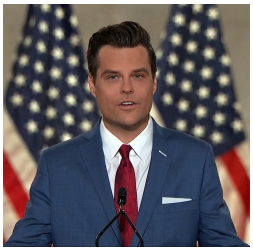
Cory Anthony Booker is a liberal and progressive Democrat who serves as a U.S. Senator from New Jersey since 2013 and is now in his second term. He is the second African-American to be elected to Senate after Barack Obama. Previously, he was mayor of Newark also for two terms from 2006 to 2013. He graduated from Stanford University with a bachelor's degree in political science and master's degree in sociology. After that, he got a Juris Doctor from Yale Law School and where he gave free legal advice to low-income residents in the area.

In relation to political positions, Booker has worked for supporting affordable housing which was doubled during his first term as mayor of Newark. And then he got reelected for his second term. He supports LGBTQ+ and women's rights, same-sex marriage, single-payer healthcare, climate change, and legalizing cannabis in the U.S. on the federal level. He also addresses wealth inequality in his politics, especially in regards to the racial wealth gap, and pursues restructuring national immigration policy (Wikipedia, 2023d). Booker was a candidate in the 2020 presidential campaign but withdrew because of a lack of money to sustain a campaign that can secure a Democratic win (Naylor, 2020). He shares many of the same political views as Kamala Harris and showed his support when she was a nominee for vice president by calling her his "sister" and told that he believed she was one of the most qualified people to be such a nominee in an interview back in 2020 as well (MSNBC, 2020).

We choose Booker because he is an experienced Democrat who is active on Twitter. We have trouble finding a Democrat who is popular, active enough on the platform, and also belongs to the age group of politicians over 45 years old. Compared to the other selected politicians, Booker might be the least known because he has not been covered as much in the media as others. This can also be observed in the number of retrieved tweets as he has only $280K$ which covers over seven months, while this number is what you get for other politicians in a month. In continuation of this, we choose to retrieve data back in the middle of 2020 because we believe he was still someway in the spotlight there as that was after he did his presidential campaign. That is why we choose to retrieve data back then than in a more recent year.

4.3.7 Matt Gaetz

Twitter accounts:	@mattgaetz, @RepMattGaetz
Twitter followers:	2M, 2.3K (as of April 2023)
About the retrieved data	
Total number of tweets:	341k
Period:	January – March 2019
Political position:	Member of the House of Representatives from Florida's 1st district
State:	Florida
Age:	36



Note: Information and picture are from Wikipedia, 2023k.

Matthew Louis Gaetz II comes from a family of politicians with both a father and grandfather in significant political positions. He is a far-right Republican, a close ally and an early supporter of Donald Trump. Gaetz has a bachelor's degree in interdisciplinary sciences from Florida State University and got a Juris Doctor degree from William & Mary Law School in 2007. The first time he was elected to Congress was in 2016 but prior to that he practiced law at a private law firm for a short period and served in the Florida House of Representatives. Like Lauren Boebert and Majorie Taylor Greene, Gaetz is also a member of the Freedom Caucus and additionally also a member of the Republican Study Committee (a group of conservative members of the Republican party). He advocates for tax cuts and for legalization to openly carry weapons saying gun rights are "granted not by the government but by God" (Wikipedia, 2023k). He also advocates for a wall between the US and Mexico, acknowledges climate change and has made a proposal on how to minimize the worst of global warming.

There has been a number of less serious conflicts where Gaetz has violated the House ethics by for example paying less than the legal minimum for office rent or address a witness right before a trial to either test or threaten them.

We choose Gaetz because he is a leader of a new generation of the Republican party with his "Alexandria Ocasio-Cortez-level media savviness" and "influential political connections" (BBC-News, 2021). He is young, talented and popular. The data related to Gaetz is the oldest of all the politicians in this project as the tweets mentioning him are from the first months of 2019. We came to the decision that we had to use data from a period that far back because that is before the police (and the public) got the tip of him being part of sex trafficking in 2020. In 2021 it became publicly known that Gaetz was subject to an FBI investigation on his being engaged in a sexual orgy with underage prostitutes (Wikipedia, 2023k). Even though the event happened a long time ago, to be sure that it was not a dominant topic in the dataset - thereby biasing the models trained on this data, we had to resort to older tweets than those used for the other politicians.

4.3.8 Ted Cruz

Twitter accounts:	@tedcruz, @SenTedCruz
Twitter followers:	5.9M, 3.3M (as of April 2023)
About the retrieved data	
Total number of tweets:	631k
Period:	January – February 2022
Political position:	U.S. Senator from Texas
State:	Texas
Age:	51

Note: Information and picture are from Wikipedia, 2023q.

Rafael Edward Cruz (Ted Cruz) is an experienced and highly influential Republican politician. He studied public policy at Princeton University and in 1995 graduated from Harvard Law School with a Juris Doctor degree. He has been a senator since 2013. Prior to this position, he was solicitor general of Texas[4]. He was a domestic policy adviser for George W. Bush's presidential campaign in 1999. In private, he has been an associate in a law firm, and his latest position before the senator was at a very big law firm called Morgan, Lewis & Bockius where he represented corporate clients. He competed against Donald Trump as the Republican nominee for president in 2016 and has been an open critic of him. Now Cruz is an important ally of Trump in the Senate. Cruz is a part of several senate committees, most importantly the Committee on Commerce, Science and Transportation of which he is ranking member (most senior member), and the Committee on Judiciary of which he is ranking member of the Subcommittee on The Constitution. He is conservative and supports gun rights, the death penalty and wants a significantly smaller government, and advocates for free trade. On the other hand, he is against outsourcing of jobs from the US to other countries and rejects the scientific consequences of climate change, and wrote President Trump in 2017 with other senators a letter that he should withdraw from the Paris Agreement (Wikipedia, 2023q).

We choose Cruz because he is an experienced and powerful politician with harsh, retorque against both democrats and fellow republicans in the senate. He has had a lot of media attention although there have been several public frenzies against him. One major frenzy happened in February 2021 when he and his family went to Mexico during a water and electricity crisis. It is important for us to avoid including tweets from periods with many public heads as we expect the tweets to deal with these subjects to a great extent. The data regarding Cruz is from the very beginning of 2022 to avoid some of the publicity because, in May of that year, there was big media coverage about his big loan of money to run his election campaigns.

[4]The government lawyer appointed to represent the federal government before the United States Supreme Court.

4.4 Notes and Considerations

In primo February of 2023, Twitter announced that it will change the use of APIs on its platform from free to paid subscription-based. This has an impact on the project because at that time we were not sure of which subjects to investigate. Within a few days, we chose the eight politicians, did some quick research, and then retrieved data. Overall, this change in API access was a disappointment because it made us unable to upscale or combine the dataset with tweets from different time periods to make it more representative of what is generally being tweeted about and not just one specific period. That could possibly have improved the quality of the results.

We experienced some interruptions when we retrieved data which happens, for instance, when there was busy traffic on Twitter the, cause too many requests were made at a time during the days up until the change in the API access. For us this meant that we had some waste of API calls. In the case of retrieving data for Ted Cruz, we ran into a broken connection error that prevented us from getting tweets about his personal account for a whole month.

Note that the stated number of followers for each politician on Twitter is from this year. As we do not have access to check the number of followers back in time which is what we believe is indicative of activity and popularity, it is worth noting that a politician might not have been that much tweeted about back then, but as we set the limit of a minimum number of tweets, we are sure to have data to work on.

4.5 The fastText Traning Set

In addition to the data for the politicians which is about 4M tweets, we have another data set of 2M tweets that in combination are used for training the fastText model. The additional tweets are about randomly chosen popular celebrities, both females and males, who we expect have received many comments related to their appearance. The celebrities are actors, artists, and models. An example of such a celebrity is the singer Rihanna who has 108M followers on Twitter and is ranked as the fourth hottest female celebrity in 2023 (TheTrendSpotter, 2023). We extend the fastText training set because we believe it will make the model better trained on appearance-related language and therefore will return better similarities when SAFT calculates the appearance scores. After preprocessing, we end with 5.3M tweets, which is the number of tweets fastText is trained on.

5 Tweet Preprocessing

Data preprocessing is the step of cleaning and preparing the data for the analysis part. We only clean the `text` column as it holds the message of a tweet. The text data contains lots of noise and uninformative words/parts that need to be handled. These handlings will reduce the extent of the problem which will improve the performance of the models used in the experiment (Haddi et al., 2013).

In this project, two different kinds of preprocessing are made for the tweets. We refer to these as steps in two parts. The first part is for LDA and faxtText to find appearance-based tweets and the second part is for the VADER sentiment analysis. For both parts, we start with removing duplicated tweets and tweets where more than one of the eight chosen politicians is mentioned. The reason for the latter is that we want the topic model or embedding model in the experiment to detect the appearance-based language in tweets that only involves one of the eight politicians. Neither of the models knows who is the subject of the tweet, so removing these tweets allows us to be able to distinguish them between the eight politicians because we want to assign a tweet to only one politician. But notice that other Twitter users (hence other politicians) besides the selected politicians can still be mentioned in the tweets. We count every time a politician is mentioned because it gives us insights into the relationships between the politicians in the period the data is extracted from K. If two politicians are mentioned too frequently together, there is possibly a bias in the dataset because maybe a publicly debated incident happened with them during that time.

5.1 Preprocessing for SAFT

Below is a listing of how we preprocess the language in the tweets which occur in the order that we execute.

1. Mix of uppercase and lowercase letters
2. Mentions (Twitter account names)
3. URL
4. Placeholders (e.g., when linking to a video, the video pops up)
5. HTML reference characters (e.g., & < >)
6. Non-letter characters including digits
7. Repetitive characters that occur consecutively more than twice in a word
8. Contracted words (e.g., let's)
9. Tokenization
10. Stop words
11. Additional punctuations (e.g., for emojis)
12. Lemmatization

13. Words/parts of at one or two characters
14. Words that only appear once (aka. singletons)

5.1.1 Simple Preprocessing

As for noise, uninformative parts, and inconsistency in the tweets, we elaborate on how we handle these issues in this paragraph. First of all, we lowercase all the text. It is worth noting that users on the platform may not follow the standard capitalization rules, as they may use caps for emphasis. Emphasis can also be shown by adding repetitive characters in a word like `'uhhhhhh'`. In this case, we cut it down to the base form `'uhh'` by removing repeated characters, which helps to reduce the number of variations of the same word. We remove uninformative information such as the name of the Twitter accounts, URLs, placeholders, HTML reference characters, and non-letter characters such as numbers, dots, commas, emojis, etc.

Often, tweets are expressed informally, so contracted words occur like `'I'll'`, `'you're'` etc. We use the contractions library to decontract these words, which also identifies and decontracts misspelled words like `'Ill'` and `'youre'` which are contractions with missing apostrophes. Almost all decontracted words are considered stop words so they will be removed later anyway. The purpose of decontraction is to avoid having too many variations of the same word, which would require adding all these variations to the stop words list.

Finally, we tokenize the tweets by breaking sentences down into simpler pieces (words). We use some libraries from the package Natural Processing Toolkit (nltk) that is specifically made for working with natural language processing in Python. Words with at most two letters or characters are not descriptive, so we remove them. Initially, we do not consider making a function for this. However, after observing the results from the first and second iterations, we notice that one character such as a hashtag, numbers, and one to two letters words still occur. We do not want to add the whole alphabet, numbers, etc. to the stop words list to remove them, so we create a function instead. Part of this issue was due to the used tokenizer. At that time, we used the TweetTokenizer from nltk which tokenizes words but keeps emojis and hashtags together as a token. As we want to tokenize everything, we use another tokenizer called WordPunctTokenizer from nltk instead.

After the first iteration, we additionally observe that there are many singletons (words that only occur once), so we make a function to remove these as well. Some of the words that fall into this category are misspelled words and wrong abbreviations which are words that will affect the fastText model negatively. This function is executed last in the preprocessing process after lemmatization (we elaborate on lemmatization below).

5.1.2 Stop Words Removal

Stop words are words that are filtered out of a text during natural language processing because they are insignificant. They are the most common words in the English language and occur frequently in any text (Rajaraman & Ullman, 2011). In NLP, we train our models using only significant words, so stop words such as adjectives, nouns, and (descriptive) verbs are usually not considered stop words. The natural language toolkit (nltk) provides a list of stop words that we use for this project but we also add some more words. After decontracting words such as `'wasn't'` to `'was not'`, these two words are removed as they are stop words.

5.1.3 Lemmatization

Lemmatization is a process of grouping inflected forms of words to their base form, known as the lemma. It is very similar to the process of stemming which removes suffixes and prefixes of a word without consideration for the context or part of speech (POS) of the word. The stemmed version of the word `'cats'` is, for example, `'cat'` and `'leaves'` is `'leav'`. In the last example, there is no way of knowing from the stemmed word if it is the noun `'leaf'` or the verb `'leave'`. Knowing this will require context. Unlike stemming, lemmatization is about finding morphemes added or removed to a word whenever it is affixation or alternation. Because the process of lemmatization involves consideration of context and POS, it is possible to detect strong and weak conjugations (Wikipedia, 2022b). Therefore, lemmatization is more syntactically accurate than the process of stemming but because it requires the usage of a dictionary it is a heavier task. Lemmatization is an important step in preprocessing for our project as the word embedding model used for analysis will otherwise find different semantic implications for all forms of a word. A consequence can be that the result after topic modeling will be ambiguous or vague topics. We have chosen to lemmatize adjectives, nouns, and verbs.

We have written functions that handle each of these considerations in appendix C. Many of the functions are written in regular expressions and applied by using lambda functions.

5.1.4 Statistics on Preprocessing Steps

Preprocessing is an essential part of the project as many functions are made and used to get a clean dataset. To assure the quality of the ongoing preprocessing, to follow along the process, and to get an indication that the functions give reasonable output, we do statistics on the number of removed duplicated tweets, tweets with multiple mentioned politicians (of the eight), stop words, and singletons. The code for these functions can be found in appendices B and D.

5.2 Preprocessing for VADER Sentiment Analysis

Below is a listing of how we preprocess the tweets for sentiment analysis which occur in the order that we execute.

1. Mentions (Twitter account names)
2. URL
3. Placeholders (e.g., when linking to a video, the video pops up)
4. HTML reference characters (e.g., & < >)

VADER needs a text to be coherent (not tokenized) and in addition also uses emojis, punctuations, repeated letters, capitalizations, degree modifiers, and conjunctions to evaluate the sentiment of a text. The original tweet does therefore not require a lot of preprocessing. Even so, we do a little cleaning of the text to remove URLs and placeholders which we consider redundant for sentiment analysis.

5.3 Notes and Considerations

Besides nltk, there is also Gensim which is a library that also has functions for working with natural language processing. We have considered using Gensim's `simple_preprocessing` functions to do a part of the preprocessing. Nevertheless, we write functions for other preprocessing purposes, so we think that we will have more control by writing all the functions ourselves.

We also handle misspelled words which are done by ignoring all words with a frequency under five in the fastText model (minimum count of fastText). By doing that we expect fastText to filter out most misspells because we assume they will occur a few times in the whole data set. Words that are regularly misspelled are not handled. An experiment using an autocorrect library results in creating new complications, for example, is the word 'beto' (which refers to the democrat Beto O'Rourke) corrected to 'beta'. Therefore, we choose to not handle this problem further.

A category of words that we decide not to handle in the preprocessing is abbreviations such as acronyms and initialisms because they exist in many categories and subjects it is too time-consuming to explore which words should be included. We are aware of abbreviations and if one appears as a very frequent word or an important word in a topic cluster, it is added to the list of stop words.

6 Experiments and Results

6.1 Failed and Incomplete Approaches

Before we design the final algorithm for scoring how appearance-based the language in a tweet is (SAFT), we try to detect appearance-based tweets with some of the most popular approaches established in section 2 which are LDA, and clustering based on a fastText model. Although we do not succeed in reaching the goal with these approaches, they are still crucial for us and the stepping stone for developing SAFT. Notice that it is a requirement that the data has been preprocessed before using these models.

6.1.1 LDA

LDA is a statistical model of Gensim that can be used to label tweets with topics based on the words in them. To use it separately on the dataset of each politician we first create a dictionary that contains all the words in the tweets as values and with a unique id integer as key. Then we make a bag of words corpus which LDA uses. Each element in the corpus represents a tweet that contains the number of distinct words expressed as a tuple like this: tuple(token-id, token-count) where the token-id is the word's unique id integer and the token count is the frequency of the word in that particular tweet and not the whole corpus. For every word in a tweet the probability of the word belonging to a topic is found by: $p(word\ with\ topic) = p(topic|document) \cdot p(word|topic)$ after randomly assigning it to one of k topics (the parameter defined number of topics in the corpus). A tweet can be assigned to several topics.
As for important parameters of LDA, we specify the number of topics k to be 20 because we expect the set of a politician's tweets to contain many topics. The `chunksize` parameter defines a chunk of tweets LDA at once can calculate the probability of a word belonging to a topic. We set it to 100. Another important parameter is `alpha` which dictates how topics should be distributed across tweets. We set it to 'auto' such that LDA automatically estimates a value for each topic itself (Řehůřek, 2022). The rest of the parameters are randomly set to values close to the default. The code for the LDA model is in appendix E. The results of LDA are an interactive HTML file that can be found in the attached zip folder (specific folder: LDA results). By observing the LDA visualization results, we see that the created topics are not specifically related to appearance. It can be due to the preprocessing not being thorough enough but it might also be because appearance is a way of speaking about a topic rather than being an actual topic itself. For this reason, we find the results of LDA useless for further investigation and conclude that this model is not a suitable approach to our purpose.

It takes about three to five hours to train the model on a dataset of one politician. We train the model on the politicians separately and observe the topics on them one by one, but it would be interesting to see the result of training the model on the whole dataset of all the politicians. Perhaps this would show a clearer appearance topic when the dataset is larger but taking into account that LDA performs best on datasets that are not noisy and sparse (as mentioned in section 2), the $3.6M$ tweets might be too big for LDA's optimal performance. By using the university's high-performance computing resources (HPC) to run the model training, we are able to model the politician separately but when we run for a combined dataset of all the politicians, it is not possible with the time limit of three days that the university's HPC's permit. This means that the model does not run in linear time, so it is unknown approximately how long it requires to run

for the whole dataset. This means it uses resources that we do not have available.

6.1.2 Cluster Analysis on fastText Word Embeddings

Another approach we try in order to detect the appearance in tweets is inspired by the studies of Esposito et al. (2016) and Adnana et al. (2021) who successfully detect topics by using simple machine learning clustering algorithms on reduced word embedding models.

We implement fastText's Skip-Gram model to make word embeddings through Gensim models and specify the parameter `vector_size` to be 300 which is the number of features of the word vector. We set it to 300 because the bigger the number the more information will be captured and even though it is larger than the default size of 100, it is still within the range of what is most popular but the downside of having a large number is that training takes the model longer (fastText, 2022). The parameters `window` and `min_count` are set to default values. We train the model on the politicians separately with an approximate time of half an hour per politician. For visualizing the result, we use the t-SNE algorithm from the sklearn library to reduce the dimensionality from 300 to two. After that, we search for clusters, first by the k-means approach and then with the DBSCAN approach. The idea of these approaches is that words are clustered based on their similarities and high similarities between words means that the Euclidean distance between these is close in vector space.

The centroid-based clustering algorithm k-means works by randomly choosing k points as initial cluster centers and then assigning all points to the closest cluster center. After the assignment, the mean points in each cluster are calculated as the new cluster centers (aka. centroid). Adjustments happen as points can get assigned to other clusters because they have shorter Euclidean distances to the newly updated cluster means. This iterative process is repeated until no more points get reassigned. The clusters made by k-means are bad and not representing topics. DBSCAN is a density-based clustering algorithm that groups points that are close together into clusters and points that are outside any clusters will be considered as noise. This algorithm groups together data points that have dense neighborhoods – meaning points closely placed together. Points that stand alone in low-density areas will be marked as outliers. Unlike the k-means algorithm which requires a predetermined number of clusters, DBSCAN does not have this requirement. Instead, it relies on two parameters: the minimum points which set the minimum density of data points required for an area to be considered dense, and ϵ which sets the radius around each point that the algorithm examines (Wikipedia, 2022a; Williams, 2011). DBSCAN might make better clusters than k-means, but it requires some parameter tuning.

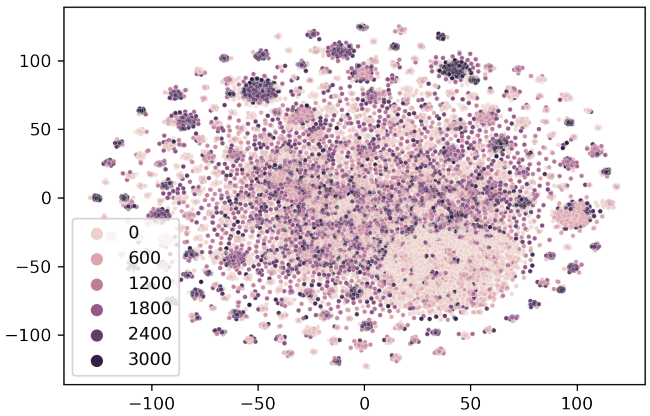


FIGURE 6.1: Visualization of the DBSCAN clusters.

We have set our parameters in DBSCAN as $\epsilon = 1$ (a result of running the k-NN algorithm) and minimum points = 4 (the default to set for 2-dimensional spaces). By observing the visualizations with six clusters specified in figure 6.1, we see that DBSCAN defines no distinct clusters which can mean that the dataset is not clean enough and contains a lot of noise that makes the word vector relations unclear. We could improve the result by cleaning the dataset even more but as we do not observe anything meaningful from this, we continue with the following approach.

6.2 Scoring Appearance with fastText (SAFT)

6.2.1 fastText for SAFT

After additional data retrieval of tweets mentioning popular celebrities on Twitter and improvements in preprocessing, we train the final fastText model on a combined preprocessed dataset with the celebrities and the selected politicians to extend the training set because we believe the model is better trained on the appearance-based language, the more text it gets on people who are famous for their appearance. The model is trained on approximately $5.3M$ tweets with parameters set as previously.

6.2.2 The Algorithm SAFT

SAFT is an algorithm that assigns an appearance score for a word collection (tokenized word list) based on each word's similarity score by using the word embedding model from `fastText`. We define a set of words that are related to appearance which can be seen in the code in appendix I. To find the appearance score of a tweet, SAFT uses the fastText function `similarity()` to calculate the similarity between the words in the tweet and the defined set of words. As mentioned in section 3, the similarity between words is the cosine distance between these vectors. The similarity score is given on a scale from 0 to 1 where 0 indicates 0% appearance-language and 1 indicates 100% appearance-language in a tweet.

The way SAFT functions is by returning a list of tuples for each tweet, where each tweet word is represented in a tuple with a similarity score. The information in a tuple is the tweet word as the first element and the highest similarity score to a word in the appearance-based word set as the second element. So, for instance, the tuple for a word like `'pretty'` is `('pretty', 1)`. The way it calculates the similarity score for a tweet word is by first checking if the word is contained in the appearance-based word set. If this is the case, it returns a tuple with the word and a maximum score of 1. If not, it loops through the appearance-based word set and calculates a similarity score for each of these words. Instead of storing these similarities, it will incrementally update a variable called "max word similarity" that contains the highest similarity calculated so far. A conditional statement is included in this loop that will break the looping process if the max word similarity variable has reached a score higher than 0.95. The reason for including this statement is to improve the runtime because we believe it is not worth to continue calculating if a score is over that value as it is close enough to 1.

The reasoning behind returning the max word similarity is: if a word has a high similarity with another word, it means that these words are often used in similar contexts or related to something similar. As the defined set consists of 150 words related to appearance, we return the similarity score to a word to avoid having to add other variations and synonyms of the word in the set. For instance, the word `'boob'` is in the set but there might exist other variations and synonyms in a tweet like `'boobie'`, `'tit'`, and `'breast'`. Ideally, the fastText model should be well-trained on this problem such that other ways of expressing the same word will have a high similarity score, and therefore, we omit to add these resulting in another improvement of the runtime. Initially, we programmed SAFT to return a dictionary for every tweet with tweet-word as key and highest similarity score as value but in case a tweet contains two identical words, only one is taken into account with the dictionary as this data structure does not allow duplicates. Therefore, we change it to returning a list of tuples instead. After SAFT has computed

the appearance score for every word in a tweet, we take three different approaches to detect the whole tweet's final appearance score.

In all three approaches, the number of words in a tweet is declared by the variable n and the similarity score of a word as s.

In **approach 1** as is the average of all the words appearance scores s.

$$as_1 = mean(s) = \frac{\sum s}{n} \tag{6.1}$$

In **approach 2** a threshold δ is set, so words with appearance scores over or equal to this threshold are divided by the number of words n in the tweet.

$$as_2 = \frac{\sum [s \geq \delta]}{n} \tag{6.2}$$

In continuation of approach 2, in **approach 3**, we find the share of words that are over the threshold.

$$as_3 = \frac{n_{[s \geq \delta]}}{n} \tag{6.3}$$

6.2.3 Considerations in Regards to the Appearance-Based Set of Words

We order the words in the set into categories to keep them organized. The categories are gender, race, color, age, to be, body, body parts, sexuality, makeup, and selected actions. The set consists of about 150 words. Initially, we created a set that was double the size of this one but as we experienced a longer runtime the more words there were, we reduced the set by removing synonyms.

The contexts in which these words occur determine the degree of how appearance-based a word is. Words that describe appearance can often also be used to describe other things. For instance, one can look cold if one does not show kindness or emotions but one can also be cold if one is freezing. In the first example, the degree of the sentence containing appearance-based language is higher than in the second example where the actual appearance-based language is not present because of the semantics. However, we do not take this matter into account with this algorithm, therefore it can be left for a further improvement in the future work. We try to be consistent with the choice of words by only including words that clearly describe or are related to appearance. This part is subjective because the definition of what is an appearance-based word may vary between persons or cultures. Therefore, the appearance-based set of words is only an example and should be developed further. The word classes are primarily adjectives and nouns as when talking about appearance, one's physical qualities (nouns) are described by using adjectives. What a person does, does not necessarily tell anything about their appearance. Nevertheless, we include a category for selected verbs because there are a few actions we believe can be associated with appearance such as if someone is 'twerking', one can appear to be sexy.

6.2.4 Example of SAFT Algorithm with Approach 1

To get a better understanding of the algorithm, we now illustrate it with an example. Imagine the following tweet: *"@XXX is pretty and drinks black coffee with rabbits"*. The tweet is preprocessed to: `['pretty', 'drink', 'black', 'coffee', 'rabbit']`. Now SAFT goes through this list of tokens. At first, it looks at `'pretty'` and checks if the word is in the set with appearance-based words. This is the case, therefore this tuple is returned `('pretty', 1)`. The second token `'drink'`

is checked. As this token is not in the set, similarity scores to the words in the appearance-based word set are calculated and the variable "max word similarity" is incrementally updated with the highest similarity score calculated so far. A tuple with the token and the highest similarity is returned. The third token `'black'` is checked. This token is in the set, so it returns a tuple of this token with a similarity score of 1. The fourth token `'coffee'` is checked. It is not in the set, so similarity scores with words in the set are calculated, and the same procedure as before. Likewise, for the fifth token `'rabbit'`. For calculating the overall tweet appearance score, SAFT calculates the average of all the maximum values. The following table 6.1 is a representation of this example.

	Max word similarity
pretty	1
drink	0.51
black	1
coffee	0.58
rabbit	0.41
Appearance score (average of maximum similarities)	**0.70**

Note: The similarity values in the table are actual numbers based on the `fastText` model trained on the dataset.

TABLE 6.1: SAFT algorithm example.

Additionally, when SAFT executes on the same example following approaches 2 and 3 with threshold $\delta = 0.5$, we get the appearance scores of 0.62 and 0.80, respectively.

6.3 Results of SAFT

In this section, the results are visualized in boxplots with belonging tables for all three approaches and gender distributions. The visualizations of the distributions for the other two groups; party and age with tables are in appendix L.

In figure 6.2 we observe the distribution of the appearance scores on all the politicians following the three approaches. Remember that the appearance score ranges from 0 to 1, where 0 (0%) is that no appearance-based language is present and 1 (100%) is the opposite.

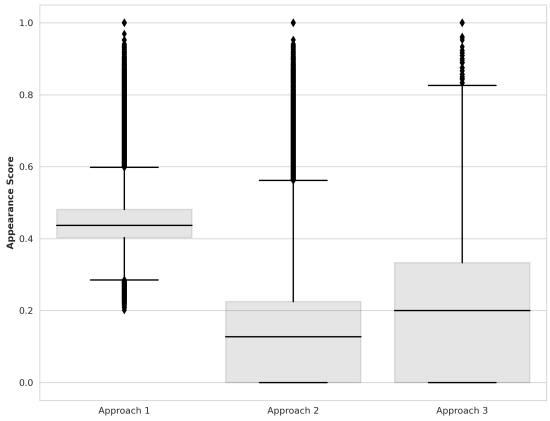


FIGURE 6.2: Distribution of the appearance scores on the whole dataset using the three approaches.

	Approach 1	Approach 2	Approach 3
count	3,636,467	3,636,467	3,636,467
mean	0.450	0.153	0.239
std	0.079	0.153	0.224
min	0.201	0.000	0.000
25%	0.403	0.000	0.000
50%	0.437	0.127	0.200
75%	0.481	0.225	0.333
max	1.000	1.000	1.000

TABLE 6.2: Values for boxplots in figure 6.2 (with threshold of $\delta = 0.5$ for approaches 2 and 3).

By including the values in table 6.2 in our observation, the average tweet appearance score in approach 1 is highest with an overall presence of appearance-based language at 45% where most tweets range between 40.3% to 48.1%. With these numbers, we see that for the majority of the tweets, most of the contents (more specifically words) have some contributing scores that pull the overall tweet score up because when reading general random posts on Twitter, we usually we do not observe that every second comment relates to appearance. This could be the case if a specific person is known for their appearance but this is not the case for the selected politicians. Therefore in our opinion, the results from approach 1 do not reflect reality because words that are not seen as relating to appearance with scores from 0.2 and below still contribute to the overall tweet score. This issue is taken into account in approaches 2 and 3 but one remark for approach 1 is that no information is lost as all scores of every word are included to compute the final tweet score.

The issue is handled in the other two approaches by including a threshold of $\delta = 0.5$ with the purpose of considering words with a score over or equal to this value when calculating the overall tweet score. This threshold is set at 0.5 because we observe that many words right under this value are not related to appearance and there are still many words over this threshold that do not relate to appearance (we will look at an attempt with a higher threshold in the next subsection) but after consideration, we decide to go with this threshold.

In approach 2, we observe that it is the one with the averagely lowest appearance scores with an overall presence of appearance-based language at 15.3% where the majority of the tweets range between 0% to 22.5%. Similarly in approach 3 with the same threshold, the tweets have an overall presence of appearance-based language at 23.9% where most tweets range 0% to 33.3%. The scores in the latter two approaches are closer to what we expect to see in reality compared to approach 1 because they evaluate words that are most dissimilar (under the threshold) to the defined appearance-based words in our set to have zero appearance. The difference between these two approaches is that we still use some information in approach 2 as it uses the similarities

of the words that are over the threshold, while in approach 3 it is the share of how many that are over the threshold. So, in approach 3, most information is lost because words that have a medium similarity score just right at the threshold at 0.5 and words with the highest possible similarity at 1.0 count as the same while in approach 2, we still differentiate these. From this reflection, we conclude to work with approach 2.

There are both advantages and disadvantages to all of the three approaches. Some of which are based on our expectations. The following table is an overview:

	Advantage	**Disadvantage**
Approach 1	No information lost	Far from expectation of reality
Approach 2	Close to expectation of reality	Some information is lost
Approach 3	Close to expectation of reality	Information is lost

TABLE 6.3: Advantages and disadvantages of the three approaches of finding the appearance score.

As mentioned, we consider approaches 2 and 3 to give scores that are closer to reality because of their aggressiveness in taking into account the number of appearance-based words (words over the threshold) present.

6.3.1 Setting the Threshold

In an experiment with threshold $\delta = 0.7$ with statistic values in figure 6.4 (and additional boxplot visualizations in appendix L), the overall presence of appearance-based language in the tweets with approaches 2 and 3 are 4.2% and 4.3%, respectively (statistics table L.16). In relation to the previous discussion about how rarely we actually see posts and comments about appearance except for famous people that are known for their appearance in reality, these numbers seem fair. As observing the boxplots for the politicians in L.3, outliers need to be considered because – if not it will look like there are no data (boxplots) for Greene, Booker, and Cruz because their points are too scattered from one another but when looking at the statistics below, Greene actually have an average appearance score that is a bit higher than Harris and it is even based on more tweets. Part of our findings will be based on visualizations but since these are not sufficient at this threshold of 0.7, we continue with the threshold of $\delta = 0.5$.

	Boebert	**Ocasio-Cortez**	**Gaetz**	**Greene**	**Harris**	**Buttigieg**	**Cruz**	**Booker**
count	663,961	541,527	326,774	528,632	496,240	252,740	592,748	233,653
mean	0.04499	0.04434	0.04369	0.04274	0.04240	0.04073	0.03934	0.03875
std	0.10937	0.10650	0.10728	0.10633	0.10398	0.10253	0.09974	0.09802
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75%	0.03474	0.04762	0.04339	0.00000	0.04762	0.04167	0.00000	0.00000
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE 6.4: Statistics on politicians with approach 2 and $\delta = 0.7$.

6.3.2 Results on Gender Distributions

The main part of our research questions is to find out if there is a difference in how appearance-based language is present in gender distribution. The boxplots of all three approaches are visualized in figures 6.3, 6.4, and 6.5 with corresponding table beside each plot. We observe that the females get on average higher appearance scores with all the approaches. Although the distributions of appearance scores between the genders are very close in the results of all three approaches, the consistent results we get amplify that there can be a difference. Additionally, we also explore if there is a difference in how appearance-based language is present in party and age group distributions. Visualizations for these two groups are in appendix L. These visualizations show that for the party distributions (democrats/republicans) there are no consistent results with the approaches because in approach 1, democrats have higher scores but in approaches 2 and 3, republicans have higher scores. Distributions for age groups (under 45 years/over 45 years) show consistent results in all three approaches with an indication that appearance-based language is more present in the tweets with the younger age group.

To verify that there is a difference in the distributions, we run statistical tests in the following subsection.

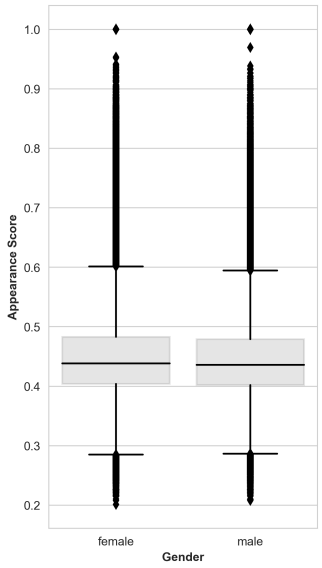


FIGURE 6.3: Distribution of appearance score for tweets with approach 1 and $\delta = 0.5$.

	Female	Male
count	2,230,443	1,406,024
mean	0.452	0.448
std	0.08	0.077
min	0.201	0.208
25%	0.403	0.402
50%	0.438	0.435
75%	0.483	0.479
max	1.0	1.0

TABLE 6.5: Values for boxplot 6.3.

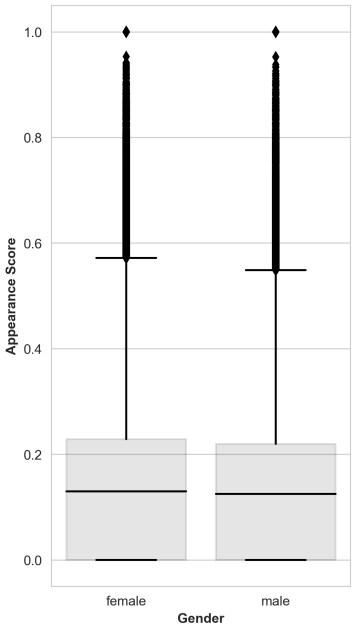


FIGURE 6.4: Distribution of appearance score for tweets with approach 2 and $\delta = 0.5$.

	Female	Male
count	2,230,443	1,406,024
mean	0.156	0.15
std	0.154	0.151
min	0.0	0.0
25%	0.0	0.0
50%	0.13	0.125
75%	0.229	0.219
max	1.0	1.0

TABLE 6.6: Values for boxplot 6.4.

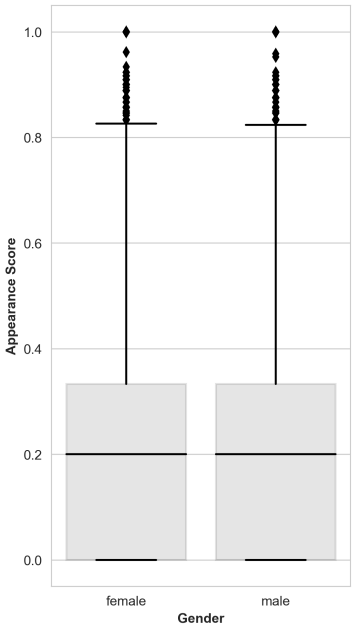


FIGURE 6.5: Distribution of appearance score for tweets with approach 3 and $\delta = 0.5$.

	Female	Male
count	2,230,443	1,406,024
mean	0.242	0.235
std	0.225	0.223
min	0.0	0.0
25%	0.0	0.0
50%	0.2	0.2
75%	0.333	0.333
max	1.0	1.0

TABLE 6.7: Values for boxplot 6.5.

6.3.3 Mann-Whitney U Test

This statistical test tells whether two distributions are identical and come from the same sample. It works by checking for randomly selected values of two sets of data X and Y, that the probability of X being greater than Y is equal to Y being greater than X. With this nonparametric test, the data does not have to be normally distributed (Wikipedia, 2023j). With a p-value that is less than or equal to the threshold of significance which is 0.05, we reject the null hypothesis (Wikipedia, 2023m).

The hypotheses are the following:

- H0: Both distributions are identical
- H1: The distributions are not identical

The results from the tests are collected in the table below:

	Gender	**Party**	**Age**
Approach 1	male < female	democrat > republican	under 45 > over 45
p-value	0.00	0.00	0.00
Approach 2	male < female	democrat < republican	under 45 > over 45
p-value	0.00	0.00	0.00
Approach 3	male < female	democrat < republican	under 45 > over 45
p-value	0.00	0.00	0.00

Note: The p-values might not be precisely 0.00 but rounded to two decimals.

TABLE 6.8: Mann Whitney U Test results on the three approaches.

In table 6.8, we observe consistent results for gender as males get lower appearance scores than females in all three approaches. The p-value is less than 0.00 which is the probability of the null hypothesis being true, and therefore, the two samples are not identical. The same applies to the age group which is also consistent with the results from all three SAFT approaches pointing out that the relatively younger politicians under 45 years are mentioned more in appearance-based tweets than the older ones. For the party group, we cannot conclude anything with certainty as with approach 1, democrats got higher appearance scores than Republicans but this is reversed with approaches 2 and 3. These Mann-Whitney U tests match our observation of the results in the boxplot and statistics visualizations, therefore, they are verified.

6.4 The Limit to be Considered Appearance-Based

As mentioned, we continue with approach 2 with threshold $\delta = 0.5$. Now that we have verified a difference in the distributions for gender, we can discuss further what is to be considered appearance-based. The point of this is to set a limit for what is appearance-based because we are interested in finding out how is the sentiment of appearance-based tweets. A limit for what should be considered appearance-based is arbitrary. Therefore, we declare two measures of what these limits can be:

1. Equal to or over the mean of appearance scores 15.3% (called lower limit)
2. Equal to or over 50% (called higher limit)

The reason for experimenting with the lower limit is that we want to take as many tweets into consideration as possible. More tweets do not mean that better results will be provided but part of this is also to see how the sentiment for most of the tweets is. The number of tweets over the lower limit is about $1.5M$ out of a total of $3.6M$. Based on our results from SAFT, for the lower limit, 43% of the tweets for female politicians and 41% of the tweets for male politicians are appearance-based. These numbers are quite high because they mean that about four out of 10 tweets are appearance-based. Comparing these numbers with what we expect to see in reality, we assess them to not correspond which is why we also have a higher limit to consider. The number of tweets over the higher limit is about $160K$ and for this one, 4.6% of the tweets for female politicians and 4.4% for male politicians are appearance-based. These numbers seem more realistic.

6.5 Sentiment Analysis with VADER

After executing the limited preprocessing (note this is not the same preprocessing as for SAFT), we use VADER to analyse the sentiment of the tweets. The compound score is used as it with only one number tells how and in which direction a tweet's sentiment is loaded.

Below we show the sentiment distribution on the two limits that we have set; 1) the lower limit equals to or over the mean of appearance scores 15.3%, and 2) the higher limit equals to or over 50%.

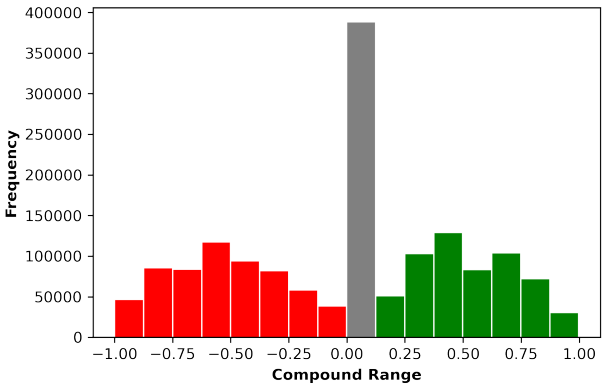


FIGURE 6.6: Lower limit: Sentiment of tweets with appearance scores equal to and over the mean of 15.3%.

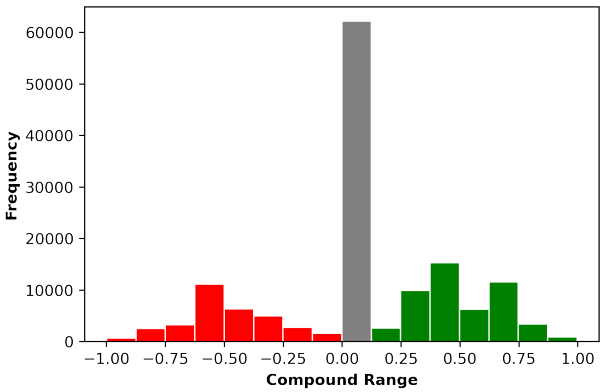


FIGURE 6.7: Higher limit: Sentiment of tweets with appearance scores equal to and over 50%.

Note: The color representation of the histogram: red is negative, grey is neutral, and green is positive

Figure 6.6 shows the sentiment distribution for all tweets following the lower limit with more descriptive statistics in appendix M table M.1. Overall, the sentiment of the tweets we consider is more in the positive direction. The distinction in the numbers of the negative and positive scores in regards to the genders is that for the females it only differs by one decimal from 38.1% to 38.0% with more to the negative side, whereas it differs for over one percent for the males from 37.7% to 38.8% with more to the positive side. This is also expressed in the overall average, as the female has an average that is -0.011, which is a bit more negative than the males at -0.002. Although the numbers are very close, we take into account that this is based on several hundred thousand tweets. Therefore this is an interesting indication that there is a slight difference in how loaded the language of the people tweeting about the politicians is.

	Negative	**Neutral**	**Positive**	**Total**	**Overall Average**
All	36,517	70,576	57,771	164,864	
Average	-0.48747	0.0000	0.48847		0.06321
%	22.1%	42.8%	35.0%	100%	
Female	22,883	44,628	35,423	102,934	
Average	-0.48484	0.0000	0.47968		0.0573
%	22.2%	43.4%	34.4%	100%	
Male	13,634	25,948	22,348	61,930	
Average	-0.49187	0.0000	0.50242		-0.00245
%	22.0%	41.9%	36.1%	100%	

TABLE 6.9: Sentiment statistic on tweets with appearance scores equal to or over the higher limit at 50% (compound score counts).

In the overall picture, the sentiment of the tweets is more in the positive direction but when looking closely into how it differs in regard to gender, the tweets about males lean more toward the positive side than females. Observing the sentiment distribution for tweets following the

higher limit in figure 6.7, there is a clearer picture of the presence of more positive tweets than negatives. With the corresponding statistics in the table 6.9, we observe that the differences in the numbers of negative and positive for both genders are that there are 22% negative tweets and 34% positive tweets, but males still have some percentages more positive tweets than females. This sentiment distribution is different from the one with the lower limit which is especially seen in the case of the females as it is reversed here with the majority in the positive direction. There are additionally in the higher limit even more neutral tweets as the number increased from 23.8% to 42.8%. All these differences tell us that many of the tweets with appearance scores in between the mean of 15.3% and 50% are more seen as negative or positive and excluding these from our evaluation, we see a big drop in the negative loaded tweets and a big increase in neutral loaded tweets. There is also a little drop in the positively loaded tweets but not a difference that is as remarkable as for the other two. So overall, we do see an indication that signals a difference in the language when talking in terms of gender differences with the males having slightly higher positive tweets than females.

7 Discussion

7.1 Indication of the Presence of Appearance-Based Language

With the successful implementation of SAFT, we now discuss what the results indicate. The purpose of this project is to compare distributions of tweets mentioning U.S. politicians in regard to appearance. Even though SAFT detects the presence of appearance in tweets, it does not necessarily mean that these should be marked appearance-based. Words in tweets can mistakenly receive high scores for instance if they either share high similarity scores with words in the appearance-based set (given the fastText model is not trained well enough) or they are homographs to words in the set. An example of a fictitious tweet that has a high appearance score but is not appearance-based can be: *"It is not nice to be in this disgusting hot climate"*[1]. To exactly determine if a tweet is appearance-based it depends on the context in which the appearance-based word appears. This is not considered in SAFT and it is a big challenge for future work of improvement. Consequently, we cannot say for sure that a high appearance score for a tweet means that it is appearance-based but we can say it is an indication of being appearance-based.

For the rest of this discussion, we assume that more appearance-based language present indicates more appearance about the politicians. So when female politicians receive a higher degree of appearance-based language than male politicians, it means that they are spoken more about their appearance.

7.2 Reflection on Results of Gender Differences

What we see from the results of the SAFT experiment is that although female politicians receive more comments for their appearance, it is the male politicians whose appearance is addressed most positively. The fact that the results show more appearance-based comments for female politicians than male politicians is not surprising since appearance is attributed as a higher value in American culture for women. The genders receive different social expectations where women tend to be judged based on their traits and men on their actions, and therefore, they value physical appearance higher for women than men (Jackson, 1992). It is for women strongly related to both intellectual competencies and social power. Therefore, Jackson believes that when people compliment the appearance of women, they are also complimenting their intellectual competence as politicians. Humans have always had a fascination with the physical appearance of oneself and one another which goes all the way back to ancient Greece Aristotle when some of the first known texts about the connection between appearance and human qualities were written. Rumsey and Harcourt (2005) suggest that people assign more positive personality traits to a person they classify as attractive regardless of gender.

In the American political world, men are superior to women both in the past and present.[2] It started as a male profession and has continued until today where women have entered the picture but still with more men than women. In continuation to the above, we associate good-looking people with attractive personalities, a good-looking politician will be perceived to have a good attitude and be trustworthy. For example, assume an author writes the following tweet

[1] This example is preprocessed to `['nice', 'disgusting', 'hot', 'climate']`. Three of the words are in the appearance-based word set which results in an overall high appearance score.

[2] In the 118th Congress beginning in January 2023, there are 75 men and 25 women in the Senate, and 301 men and 124 women in the House (StatistaResearchDepartment, 2023a, 2023b).

"Obama looks like a fucking champ on that stage". Besides the literal meaning of the tweet, it may also indicate that Obama is assigned a positive attribute in handling some areas of interest of the author.

In regards to our results, as there are more positive comments on the appearance of male politicians than on female politicians, it may indicate that people are generally more positive towards the professional presence they have, and therefore, there might be a higher probability of associating a politician with a male.

7.3 Concerns

7.3.1 Twitter as a Representation of the U.S Population

A study by Hughes and Wojcik (2019) shows that only 20% of adults in the U.S. use Twitter and less than half of these use it to discuss politics. They argue that *"Twitter users are younger and more likely to be Democrats compared with the general public and 10% of users created 80% tweets"*. This means that a large part of the population uses Twitter but it is still a very small representation of the U.S. population. Therefore, the results of our experiment show the way politicians are addressed on Twitter but it is questionable if it reflects the way they are discussed in other forums or in general. Looking at this stereotypical user of Twitter indicates that although the social media platform is popular for political discussions, it cannot stand alone in reflecting the political reality of that nation. We can compare Twitter with other social media platforms like Facebook to find similarities between the referred politicians and people's reactions and comments to strengthen the findings and enlighten which kind of users we are dealing with. As we are not looking further into the other data attributes that we have collected about the author's information, there are some interesting findings that can be investigated in an expansion of this study. Using social media for research on text data is powerful because it can tell a lot about the cultures of all kinds of people.

7.3.2 Word Similarity with fastText Word Embeddings

With the fastText model, we assume that synonyms of a word in the set of appearance-based words is represented with a high similarity score. This assumption is not necessarily true because high scores in the model exist between words that frequently appear in a similar context. For instance, in the table below the similarity score for the two antonyms good/bad is 0.63 and they are both in each others' list of top most similar words. This means that in vector space, these two words are among each other's nearest neighbors but this has nothing to do with the words being synonyms or antonyms.

Most similar to 'good'	Score
okgood	0.72
yeahgood	0.69
great	0.67
sogood	0.66
lolgood	0.65
best	0.63
bad	0.63
wowgreat	0.62
everyother	0.61
sure	0.61

Most similar to 'bad'	Score
terrible	0.67
good	0.63
horrible	0.61
horible	0.57
terible	0.56
best	0.56
badand	0.55
hmwonder	0.55
horribleperson	0.55
know	0.54

Apart from these two antonyms, it seems from the tables that most similar words are in fact synonyms. This indicates that it is reasonable for us to assume that the appearance score is higher for synonyms of words in the appearance-based set of words. On the other hand, when

we look at the similarity scores for closely related words, we get scores like these: man/boy: 0.19, man/guy: 0.49, woman/girl: 0.45, and boob/breast: 0.51. These scores seem a bit random because the score of man/boy is very different from the score of woman/girl. This gives us the impression that the model can be trained better than it is.

7.4 Improvements of the Results

7.4.1 The Dataset

There is a difference in the quantity of data between the two gender groups because female politicians have about $2.2M$ tweets and male politicians have about $1.4M$ tweets which is a difference of $0.8M$. The difference is great because it is over half of the size of the male politicians' dataset. If we have a more equal number of tweets in both datasets, it would be possible for us to get boxplots that are more descriptive and more comparable.

To get better word embeddings, we have done some improvements throughout the project for the fastText model. Initially, we only train the model on the dataset with the politicians of $3.6M$ tweets and use the same dataset as the test set which is due to the fact that we lose access to retrieve more data at the beginning of the project. But as we gain access again at the end, we choose to expand the training set with tweets about other public figures and celebrities in the U.S. which are people that are known for their appearance because of their professions as a model/actor/musician or maybe they have won some titles of being the prettiest, sexiest something. Presumably, they receive a large number of comments on their appearance. With this new training set, the fastText model has an expanded understanding of appearance-based language in general and is not limited to only politician-related tweets which we expect will result in similarity scores that reflect theory more like synonyms having higher similarity scores to each other.

By having a separate training set we will also take advantage of the fact that fastText is based on n-grams because then there is a high probability of comparing a word from the test set that does not exist in the training set.

7.4.2 Preprocessing

SAFT can also be improved with a better preprocessing of tweets. For example, the list of stop words could be extended with more frequently used words like `'say'`, `'make'`, `'think'`, `'really'` and `'know'` that are not useful and descriptive for this project. On the contrary, we can include more words in the vocabulary of fastText by handling some of the unmanaged issues described in considerations in section 5 like abbreviations and misspelled words. However, it is unknown how much value will be added if these handlings are to be made.

7.4.3 SAFT

The set of appearance-based words in SAFT can be improved. The choice of which words should be included is subjective and based on personal opinions so this can always be improved and discussed. We can improve it by having a larger set of words for securing more accurate appearance scores of words. An additional function that can be added to SAFT is to include weights for the words in the appearance-based word set. These weights determine the degree of appearance of a word because some words are more appearance descriptive. For instance, `'pretty'` is no doubt a word about appearance and therefore should have the highest weight, while a word like `'cold'` or a color does not necessarily always appear in appearance-based contexts and so these should a lower weight. The lower weights take away some of the possible misinterpretations SAFT can make when detecting similarities between words in a tweet and in the set of appearance-based words. The weights can range from 0 to 1, whereas the closer to 1, the higher weight which means the more appearance-based we have determined it to be, and the closer to 0, the less determined

it to be appearance-based. Determining what weight should be for a word is subjective because it is based on our own beliefs of how much and which words we perceive to be appearance-based. And SAFT is also not programmed in the most efficient way with adequate data structures, so this can also be looked into to improve the runtime.

7.5 Another Approach

There are still other approaches to solving the detection of appearance in tweets when classical topic modeling (LDA) fails. We propose SAFT but there are still other options. Another way is by classifying which tweets are appearance-based or not. Here we will have to label a training set of tweets as being appearance-based or not, and then use classification algorithms to use classify the rest of the tweets. The disadvantage of this is that the labeling process is time-consuming because it is done manually by ourselves and we cannot avoid including our own subjectivity when we assess which tweets are appearance-based or not. The purpose of this project is to find automated ways that we can use to find the degree of appearance present in tweets, and if we are to label a training set manually, it goes against the concept of the project. But on the other hand, the advantage of this approach is that we have more control because it involves the manual process and classification is a classic way of solving problems like this. Before developing SAFT, we looked into this approach but because of the mentioned disadvantage, the choice of which approach to use to detect appearance fell on SAFT which additionally is also because it seemed more creative and more educative to develop an algorithm ourselves.

7.6 Future Work

In summary, this means that the future work of this project includes the following:

1. Improve the training set for fastText with more data.
2. Add a function in SAFT that distinguishes between appearance-based words that are more and less appearance descriptive.
3. Do the same analysis with data from other social media platforms to find if there is a difference in the way the politicians are referred to.
4. Implement different approaches to the detection of appearance in tweets/texts and make a comparison.

8 Conclusion

In this project, we propose a solution for detecting tweets regarding physical appearance called Scoring Appearance with fastText (SAFT). It is first of all a method for scoring how much of appearance-based language is present in a set of tweets. The results from SAFT show that there is a difference in the number of appearance-related tweets based on gender, whereas female politicians receive more than male politicians. After experiments with three SAFT approaches, we select approach 2 with a set threshold of 0.5 which signals a (weak) gender-related difference. It is not possible to conclude anything of appearance in relation to which party a politician identifies with as the statistical test (Mann-Whitney U Test) shows inconsistencies across the three SAFT approaches. But, in relation to age, the results show that politicians belonging to the younger age group receive more tweets that are appearance-related.

VADER sentiment analysis is performed on tweets that have an overall appearance score of higher than 50% on the scale. We conclude that the sentiment of the tweets for both genders is primarily neutral at 42.8%, positive at 35% and then negative at 22.1% but the male politicians have a few higher percentage points of positive tweets than female politicians. Even though SAFT detects more appearance-based language for female politicians, the detected appearance for male politicians is more positive which indicates that people on Twitter speak differently about appearance based on the gender of the politicians, and the fact that male politicians get more positive tweets can be based on the tendency in American society that people generally associate the appearance of a politician with a man.

A Data Retrieval

This is inspired by the code from a GitHub repository (Foote, 2021).

```
import pandas as pd
import time
import tweepy

# Authenticitation with Twitter developer account bearer token
bearer_token = 'XXX'
client = tweepy.Client(bearer_token, wait_on_rate_limit=True)

# Information to retrieve
query = '@billieeilish -is:retweet lang:en'
start_time = '2023-01-01T00:00:00Z'
end_time = '2023-02-01T00:00:00Z'
user_fields = ['username', 'public_metrics', 'description', 'location']
tweet_fields = ['created_at', 'geo', 'public_metrics', 'text']

tweets = []
for api_call in tweepy.Paginator(client.search_all_tweets,
                                 query = query,
                                 user_fields = user_fields,
                                 tweet_fields = tweet_fields,
                                 expansions = 'author_id',
                                 start_time = start_time,
                                 end_time = end_time,
                                 max_results = 500):
    time.sleep(1)
    tweets.append(api_call)

# Create an empty list to hold dictionaries of results
result = []
user_information_dict = {}

# Make a dictionary of dictionaries with user information
for api_call in tweets:
    for user in api_call.includes['users']:
        user_information_dict[user.id] = {'username': user.username,
                                'followers': user.public_metrics['followers_count'],
                                'tweets': user.public_metrics['tweet_count'],
                                'description': user.description,
                                'location': user.location}
    # Put the data in a dictionary for each tweet
    for tweet in api_call.data:
        user_information = user_information_dict[tweet.author_id]
        result.append({'author_id': tweet.author_id,
                       'username': user_information['username'],
                       'author_followers': user_information['followers'],
                       'author_tweets': user_information['tweets'],
                       'author_description': user_information['description'],
                       'author_location': user_information['location'],
                       'text': tweet.text,
                       'created_at': tweet.created_at,
                       'retweets': tweet.public_metrics['retweet_count'],
                       'replies': tweet.public_metrics['reply_count'],
                       'likes': tweet.public_metrics['like_count'],
                       'quote_count': tweet.public_metrics['quote_count']})
```

```
# Pickle the retrieved data (the list with results)
pd.to_pickle(result,'billieeilish.2023.1-2.pickle' )
```

B Preprocessing

```
import nltk
import pandas as pd
import string
from collections import Counter
from nltk import word_tokenize, pos_tag
from nltk.corpus import stopwords
from preprocessing_functions import *
from preprocessing_statistics_functions import *

nltk.download('stopwords')
nltk.download('wordnet')

new_filename = 'all_preprocessed_4.pkl' # This is the name for the exported pickle

pickle = pd.read_pickle('all_3_not_preprocessed.pkl')
df_imported = pd.DataFrame(pickle)

# Remove duplicates
df_no_dub = df_imported.drop_duplicates('text').copy()

# Count whenever a politician is mentioned
# Create new columns and assign them to 0
df_no_dub['Ocasio-Cortez'] = 0
df_no_dub['Greene'] = 0
df_no_dub['Gaetz'] = 0
df_no_dub['Cruz'] = 0
df_no_dub['Harris'] = 0
df_no_dub['Buttigieg'] = 0
df_no_dub['Boebert'] = 0
df_no_dub['Booker'] = 0
# Go through the texts to check if they contain the politicians. If so, the value is set to 1
df_no_dub['Ocasio-Cortez'] = df_no_dub.text.apply(lambda x: count_aoc(x))
df_no_dub['Greene'] = df_no_dub.text.apply(lambda x: count_mtg(x))
df_no_dub['Gaetz'] = df_no_dub.text.apply(lambda x: count_gaetz(x))
df_no_dub['Cruz'] = df_no_dub.text.apply(lambda x: count_cruz(x))
df_no_dub['Harris'] = df_no_dub.text.apply(lambda x: count_harris(x))
df_no_dub['Buttigieg'] = df_no_dub.text.apply(lambda x: count_buttigieg(x))
df_no_dub['Boebert'] = df_no_dub.text.apply(lambda x: count_boebert(x))
df_no_dub['Booker'] = df_no_dub.text.apply(lambda x: count_booker(x))

# Create a dataframe that stores the frequency of each politician is mentioned
politician_mentioned = {
    'Politician': ['Ocasio-Cortez', 'Greene', 'Gaetz', 'Cruz', 'Harris', 'Buttigieg', 'Boebert', 'Booker'],
    'Mentioned': [df_no_dub['Ocasio-Cortez'].sum(), df_no_dub['Greene'].sum(), df_no_dub['Gaetz'].sum(),
                  df_no_dub['Cruz'].sum(), df_no_dub['Harris'].sum(), df_no_dub['Buttigieg'].sum(),
                  df_no_dub['Boebert'].sum(), df_no_dub['Booker'].sum()]}
politician_mentioned = pd.DataFrame(politician_mentioned)
politician_mentioned # For statistics
```

```
# The tweets with sums less than 2 are included in a new dataframe
# By creating a new column 'single_mentioned' that is 'True' if that text contains only one politician
# and 'False' otherwise. The True rows are added to the new dataframe
cols = ['Ocasio-Cortez', 'Greene', 'Gaetz', 'Cruz', 'Harris', 'Buttigieg', 'Boebert', 'Booker']
df_no_dub['sum_of_mentioned'] = df_no_dub[cols].sum(axis=1)

df_no_dub['single_mentioned'] = df_no_dub['sum_of_mentioned'].apply(lambda x: single_mention(x))
df = df_no_dub[df_no_dub['single_mentioned']].copy()

# Exclude Twitter messages
df['not_twitter_message'] = df.text.apply(lambda x: is_not_twitter_messages(x))
df = df[df['not_twitter_message']].copy()

# Preprocessing for Sentiment
df['text_for_sentiment'] = df.text
df.text_for_sentiment = df.text_for_sentiment.apply(lambda x: remove_mentions(x))
df.text_for_sentiment = df.text_for_sentiment.apply(lambda x: remove_url(x))
df.text_for_sentiment = df.text_for_sentiment.apply(lambda x: remove_placeholder(x))
df.text_for_sentiment = df.text_for_sentiment.apply(lambda x: remove_html_ref(x))

# Preprocessing for SAFT
df.text = df.text.str.lower()
df.text = df.text.apply(lambda x: remove_mentions(x))
df.text = df.text.apply(lambda x: remove_url(x))
df.text = df.text.apply(lambda x: remove_placeholder(x))
df.text = df.text.apply(lambda x: remove_html_ref(x))
df.text = df.text.apply(lambda x: remove_non_letter_char(x))
df.text = df.text.apply(lambda x: remove_rep_char(x))
df.text = df.text.apply(lambda x: decontract(x))
df.text = df.text.apply(lambda x: tokenize_text_punct(x)) # NOTE: It uses the nltk.word_tokenize

# Stop Words
stopwords = nltk.corpus.stopwords.words('english')

# Add new words to the nltk list of stopwords here:
new_stopwords = ['let', 'also', 'etc', 'nd', 'cannot', 'would', 'could', 'get', 'yes', 'though',
                 'repaoc', 'aoc', 'ocasio', 'cortez', 'ocasiocortez', 'alexandria', 'margie',
                 'marge', 'majorie', 'majorietg', 'cory', 'cbooker', 'lboebert', 'pbuttigieg',
                 'kharris', 'pete', 'kamala', 'harris', 'greene', 'greenee','boebert', 'buttigieg',
                 'cruz', 'gaetz', 'lauren', 'ted', 'repmtg', 'mtgreenee', 'mtgreene', 'mtg',
                 'matthew', 'matt', 'mgaetz', 'repmattgaez', 'mattgaetz', 'senbooker', 'corybooker',
                 'secretarypete', 'petebuttigieg', 'vp', 'kamalaharris', 'repboebert',
                 'laurenboebert', 'sentedcruz', 'tedcruz']
stopwords.extend(new_stopwords)

# Remove stopwords
df.text = df.text.apply(lambda x: [word for word in x if word not in (stopwords)])

# Remove additional punctuation (e.g., for emojis)
df.text = df.text.apply(lambda x: remove_punctuation(x))

# Lemmatization
df.text = df.text.apply(lambda x: lemmatize_words(x))

# Remove one and two character words
df.text = df.text.apply(lambda x: remove_one_and_two_char(x))
```

```
# Remove singletons
word_counter = Counter()
for word in df.text:
    word_counter.update(word)

singletons = set()
for word, count in word_counter.items():
    if count == 1:
        singletons.add(word)

df.text = df.text.apply(lambda x: [word for word in x if word not in (singletons)])

# A final remove of punctuation (as we experience some extra punctuations)
df.text = df.text.apply(lambda x: remove_punctuation(x))

# Remove redundant columns
df = df.drop(['Ocasio-Cortez', 'Greene', 'Gaetz', 'Cruz', 'Harris', 'Buttigieg', 'Boebert',
            'Booker', 'sum_of_mentioned', 'single_mentioned', 'not_twitter_message'], axis = 1)

# Remove empty tweets
df_export = df[df.text.apply(is_text_empty)].copy()

# Export to pickle
pd.to_pickle(df_export, new_filename)
```

C Preprocessing Functions

```
import contractions
import nltk
import re
import string
import tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import WordPunctTokenizer
from nltk import word_tokenize, pos_tag
from string import punctuation

def remove_mentions(text):
    text = re.sub(r'@\w+', '', text)
    return text

def remove_url(text):
    text = re.sub(r'https?:\/\/\S+', '', text)
    text = re.sub(r"www\.[a-z]?\.?(com)+|[a-z]+\.(com)", '', text)
    return text

def remove_placeholder(text):
    text = re.sub(r'{link}', '', text)
    text = re.sub(r"\[video\]", '', text)
    return text

def remove_html_ref(text):
    text = re.sub(r'&[a-z]+;', '', text)
    return text

def tokenize_text_punct(text):
    result = WordPunctTokenizer().tokenize(text)
    return result

def remove_non_letter_char(text):
    text = re.sub(r"[^a-z\s\(\-:\)\\\/\];='#]", '', text)
    return text

def remove_rep_char(text):
    text = re.sub(r'(.)\1{2,}', r'\1', text)
    return text

def remove_one_and_two_char(tokens):
    return_list = []
    for word in tokens:
        if len(word) > 2:
            return_list.append(word)
    return return_list

def decontract(text):
    expanded_text_list = []
    for word in text.split():
        expanded_text_list.append(contractions.fix(word))
    expanded_text = ' '.join(expanded_text_list)
    return expanded_text
```

```
def is_not_twitter_messages(text):
    boolean = True
    if 'account is temporarily unavailable because it violates the Twitter Media Policy' in text:
        boolean = False
    return boolean

punctuation_list = list(string.punctuation)
def remove_punctuation(word_list):
    return [w for w in word_list if w not in punctuation_list]

def lemmatize_words(tokens):
    lemmatizer = WordNetLemmatizer()
    lemma_tokens = []
    for word in tokens:
        word1 = lemmatizer.lemmatize(word)
        word2 = lemmatizer.lemmatize(word1, pos='v')
        lemma_tokens.append(lemmatizer.lemmatize(word2, pos='a'))
    return lemma_tokens

def is_text_empty(text):
    boolean = True
    if text == []:
        boolean = False
    return boolean
```

D Preprocessing Statistics Functions

```
def num_words(text):
    count = 0
    for i in text:
        count = count + len(i)
    return count

def count_aoc(text):
    value = 0
    acc1 = '@AOC'
    acc2 = '@RepAOC'
    if acc1 in text:
        value = 1
    if acc2 in text:
        value = 1
    return value

def count_mtg(text):
    value = 0
    acc1 = '@mtgreenee'
    acc2 = '@RepMTG'
    if acc1 in text:
        value = 1
    if acc2 in text:
        value = 1
    return value

def count_gaetz(text):
    value = 0
    acc1 = '@mattgaetz'
    acc2 = '@RepMattGaetz'
    if acc1 in text:
        value = 1
    if acc2 in text:
        value = 1
    return value

def count_cruz(text):
    value = 0
    acc1 = '@tedcruz'
    acc2 = '@SenTedCruz'
    if acc1 in text:
        value = 1
    if acc2 in text:
        value = 1
    return value

def count_harris(text):
    value = 0
    acc1 = '@KamalaHarris'
    acc2 = '@VP'
    if acc1 in text:
        value = 1
    if acc2 in text:
        value = 1
    return value
```

```
def count_buttigieg(text):
    value = 0
    acc1 = '@PeteButtigieg'
    acc2 = '@SecretaryPete'
    if acc1 in text:
        value = 1
    if acc2 in text:
        value = 1
    return value

def count_boebert(text):
    value = 0
    acc1 = '@laurenboebert'
    acc2 = '@RepBoebert'
    if acc1 in text:
        value = 1
    if acc2 in text:
        value = 1
    return value

def count_booker(text):
    value = 0
    acc1 = '@CoryBooker'
    acc2 = '@SenBooker'
    if acc1 in text:
        value = 1
    if acc2 in text:
        value = 1
    return value

def single_mention(number):
    boolean = True
    if number > 1:
        boolean = False
    return boolean
```

E LDA

```
import gensim
import gensim.corpora as corpora
import pandas as pd
import pyLDAvis
import pyLDAvis.gensim
from gensim.models import CoherenceModel
from gensim.test.utils import datapath
from gensim.utils import simple_preprocess

# Name of the html visualization file to save
export_visualization = "all_lda2.html"

# Import preprocessed pickle
pickle1 = pd.read_pickle('/home/jngu/all_2.pkl')
df = pd.DataFrame(pickle1)

# Create dictionary
word_id = corpora.Dictionary(df.text)

# Create corpus
tweets = df.text

# Create bag of word
corpus = [word_id.doc2bow(tweet) for tweet in tweets]

[[(word_id[id], freq) for id, freq in i] for i in corpus[:1]]

lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=word_id,
                                            num_topics=20,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)

# Save the model
temp_file = datapath("/home/jngu/thesis/lda_2/all_lda2_model")
lda_model.save(temp_file)

# Visualize the topics
pyLDAvis.enable_notebook()
visualization = pyLDAvis.gensim.prepare(lda_model, corpus, word_id)

# Save the visualization
pyLDAvis.save_html(vis, export_visualization)
```

F fastText Word Embedding

```
import pandas as pd
from gensim.models import FastText, Word2Vec

pickle = pd.read_pickle('all_preprocessed_4.pkl')
df = pd.DataFrame(pickle)
new_filename = 'XXX.pkl' # This is the name for the exported pickle

# fastText Modeling
fast_Text_model = FastText(df.text,
                           vector_size = 300,
                           window = 5,
                           min_count = 5,
                           sample = 1e-3,
                           workers = 4,
                           sg = 1)

# Save model
fast_Text_model.save("fast_text_model_all")
```

G DBSCAN

This code is in a Jupyter notebook (`.ipynb` file) in the attached `.zip` folder.

```
import numpy as np
import pandas as pd
import pickle
import seaborn as sns
import sklearn
from gensim.models import FastText
from matplotlib import pyplot as plt
from sklearn import manifold
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors

pickle = pd.read_pickle(r'C:\Users\tonem\Desktop\ITU\Speciale\Master Thesis Processed Data 2\aoc_2.pkl')
frames = [pd.DataFrame(pickle)]
df = pd.concat(frames)

# Load saved gensim fastText model
fast_Text_model = FastText.load("fast_text_model_2_aoc")

# Reduce dimentionality of fastText model to 2 dimentions
tsne = sklearn.manifold.TSNE(n_components = 2 , random_state = 0)
matrix_2d = tsne.fit_transform(matrix)
df_2d = pd.DataFrame(matrix_2d, columns = ['1','2'])

# Visual representation of the 2d vector space of word-vectors
plt.scatter(df_2d['1'], df_2d['2'], s=0.05)

#KNN to find optimal epsilon for DBSCAN
nbrs = NearestNeighbors(n_neighbors=5).fit(df_2d)
neigh_dist, neigh_ind = nbrs.kneighbors(df_2d)
# Sort by row
sort_neigh_dist = np.sort(neigh_dist, axis=0)

k_dist = sort_neigh_dist[:, 4]
plt.plot(k_dist)
plt.axhline(y=1, linewidth=1, linestyle='dashed', color='g')
plt.ylabel("k-NN distance")
plt.xlabel("Sorted observations (4th NN)")
plt.grid()

# min_samples = the number of dimentions/features * 2. So for Data_2D min_samples=4.
# eps = 1 as result by kNN above.
clusters = DBSCAN(eps=1, min_samples=4).fit(df_2d)

# Visualize 2d vector space for word-vectors with DBSCAN clusters
plot = sns.scatterplot(data=df_2d, x=df_2d['1'], y=df_2d['2'],hue=clusters.labels_, alpha=0.5, s=7)
```

H SAFT

```
import pandas as pd
import pickle
from gensim.models import FastText
from saft_functions import *

pickle = pd.read_pickle('all_preprocessed_4.pkl')
df = pd.DataFrame(pickle)
new_filename = 'XXX_SAFT.pkl' # This is the name for the exported pickle
fast_Text_model = FastText.load("XXX") # Load fastText model

df = df.drop(df.columns[[0,1,2,3,4,5,7,8,9,10,11]], axis=1) # Dropping redundant columns

# Create list of tuples (word/appearance score) from tweet-words
df['tweet_as_list'] = df.text.apply(lambda x: as_list(x))

# Create list of tuples (word/appearance score) from tweet-words,
# but only for words with appearance score above threshold
df['tweet_as_list_thr'] = df.tweet_as_list.apply(lambda x: as_list_thr(x))

# Assign appearance scores as approach 1
df['as_1'] = df.tweet_as_list.apply(lambda x: as_1_2(x))

# Assign appearance scores as approach 2
df['as_2'] = df.tweet_as_list_thr.apply(lambda x: as_1_2(x))

# Assign appearance scores as approach 3
df['as_3'] = df.tweet_as_list_thr.apply(lambda x: as_3(x))

# Export to pickle
pd.to_pickle(df, new_filename)
```

I SAFT Functions

```
from gensim.models import FastText

# Load fastText model
fast_Text_model = FastText.load('/home/jngu/algorithm_and_sentiment_iteration3/fast_text_model_all_300')

# Variables that are used in the SAFT functions
global appearance_based_words, threshold
appearance_based_words = set()
threshold = 0.5

# Set with appearance-based words
gender = {'boy', 'girl', 'man', 'woman', 'male', 'female', 'trans', 'nonbinary', 'masculine',
        'feminine'}

race = {'asian', 'hispanic', 'african', 'caucasian', 'nigger', 'negro'}

color = {'brown', 'white', 'black', 'red', 'blue', 'yellow', 'green', 'color', 'grey', 'light',
        'dark', 'nude', 'naked'}

age = {'elderly', 'old', 'young', 'infant', 'baby', 'child', 'kid', 'adolescent', 'adult',
        'teenager', 'youthful', 'mature', 'age'}

to_be = {'pretty', 'adorable', 'cute', 'attractive', 'beautiful', 'cute', 'elegant', 'gorgeous',
        'graceful', 'handsome', 'lovely', 'ravishing', 'stunning', 'charming', 'striking',
        'radiant', 'sophisticated', 'exquisite', 'fashionable', 'trendy', 'stylish', 'classy', 'nice',
        'good', 'hot', 'slutty', 'ugly', 'disgust', 'hideous', 'unattractive', 'childish', 'filthy',
        'cold', 'cheap'}

body = {'body', 'fat', 'overweight', 'skinny', 'slim', 'tall', 'short', 'atletic', 'chubby', 'slender',
        'muscular', 'obese', 'bald', 'blond', 'brunette', 'redhead', 'curly', 'bearded', 'fat',
        'petite', 'curvy', 'short', 'weak', 'strong', 'furry', 'hairy', 'flex'}

body_parts = {'hair', 'brow', 'nose', 'ear', 'lip', 'tongue', 'chin', 'breast', 'tit', 'clevage',
            'boob', 'nipple', 'stomach', 'arm', 'hand', 'finger', 'nail', 'beard', 'moustache',
            'tooth', 'hip', 'waist', 'butt', 'ass', 'skeleton', 'skin', 'booty', 'dick', 'penis',
            'cock', 'genital', 'vagina', 'cunt', 'pussy', 'pregnant', 'heel', 'foot'}

sexuality = {'milf', 'dilf', 'sex', 'gay', 'homosexual', 'homo'}

makeup = {'makeup', 'lipstick', 'jewelry', 'dress', 'shirt', 'suit', 'tie', 'belt', 'pants', 'skirt',
        'sock', 'shoe', 'accessory', 'glass', 'bikini', 'blazer'}

selected_actions = {'twerk', 'dance', 'shave', 'wax', 'sweat', 'spank', 'look', 'see', 'watch',
                    'observe'}

appearance_based_words.update(gender)
appearance_based_words.update(race)
appearance_based_words.update(color)
appearance_based_words.update(age)
appearance_based_words.update(to_be)
appearance_based_words.update(body)
appearance_based_words.update(body_parts)
appearance_based_words.update(sexuality)
appearance_based_words.update(makeup)
appearance_based_words.update(selected_actions)
```

```
# Max similarity between the tweet-word and the words in the appearance-based set.
# This function is used in as_list()
def word_as_tuple(word):
    max_word_similarity = 0
    if word in appearance_based_words:
        max_word_similarity = 1.0
        return (word, max_word_similarity)
    else:
        for appear_word in appearance_based_words:
            if max_word_similarity > 0.95:
                return (word, max_word_similarity)
            similarity = fast_Text_model.wv.similarity(appear_word, word)
            if similarity > max_word_similarity:
                max_word_similarity = similarity
        return (word, max_word_similarity)

# Returns a list of tuples (word/appearance score) from list of tokens (tweet-words)
# The function is used in approach 1, 2 and 3
def as_list(list_of_tokens):
    as_list = [word_as_tuple(x) for x in list_of_tokens]
    return as_list

# If a similarity value is less than the threshold the value is set to 0
# This function is used in as_list_thr()
def lowval_to_zero(as_tuple):
    word = as_tuple[0]
    similarity = as_tuple[1]
    if similarity < threshold:
        return (word, 0)
    else:
        return as_tuple

# For all tuples in as_list use lowval_to_zero() to set values under threshold to 0
# The function is used in approach 2 and 3
def as_list_thr(as_list):
    as_list_thr = [lowval_to_zero(x) for x in as_list]
    return as_list_thr

# Return the appearance score for a whole tweet from the sum of word appearence scores (similarities)
# The function is used in approach 1 and 2
def as_1_2(as_list):
    def sum_similarities(as_list):
        return sum(j for i, j in as_list)
    appearance_score = sum_similarities(as_list)/len(as_list)
    return appearance_score

# Return the share of words that are over the threshold
# The function is used in approach 3
def as_3(as_list):
    count = 0
    for i, j in as_list:
        if j > threshold:
            count = count+1
    appearance_score = count/len(as_list)
    return appearance_score
```

J Sentiment Analysis with VADER

```
# Run the out-commented lines first-time
#nltk.download('vader_lexicon')
#!pip install vaderSentiment

import nltk
import pandas as pd
from nltk.sentiment.vader import SentimentIntensityAnalyzer

pickle = pd.read_pickle('cruz_SAFT_4_improved_0.5.pkl')
df = pd.DataFrame(pickle)
new_filename = ('xxx_sentimented_compound.pkl')

# Do sentiment analysis on a text/tweet
def sentiment_function(text):
    analyzer = SentimentIntensityAnalyzer()
    new_text = analyzer.polarity_scores(text)
    return new_text

# Return the compound score
def compound(sentimented_dict):
    return sentimented_dict.get('compound')

df['sentimented'] = df.text_for_sentiment.apply(lambda x: sentiment_function(x))
df['compound'] = df.sentimented.apply(lambda x: compound(x))
pd.to_pickle(df, new_filename)
```

K Statistics on Preprocessing

Number of tweets when removing duplications:

Before	After	Difference	Percentage
4,038,121	3,860,762	177,359	4.39%

The number each politician is mentioned in the same tweets as another politician of the dataset:

Politician	Mention count
Ocasio-Cortez	596,619
Greene	606,912
Gaetz	373,265
Cruz	639,290
Harris	566,898
Buttigieg	273,556
Boebert	722,717
Booker	281,826

Number of tweets removing those containing multiple mentions:

Before	After	Difference	Percentage
3,860,762	3,673,143	18,619	4.86%

Number of words when removing stop words:

Before	After	Difference	Percentage
69,308,817	34,975,550	34,333,267	49.54%

Singletons:

Unique words	Number of singletons	Percentage	Unique words left
375,222	255,309	31.96%	119,913

Number of tweets when removing empty rows:

Before	After	Difference	Percentage
3,672,969	3,636,275	36,694	1.0%

L SAFT Results

Here is an overview of the SAFT results with the different approaches to the calculation of the appearance score.

L.1 All politicians compared

Approach 1:

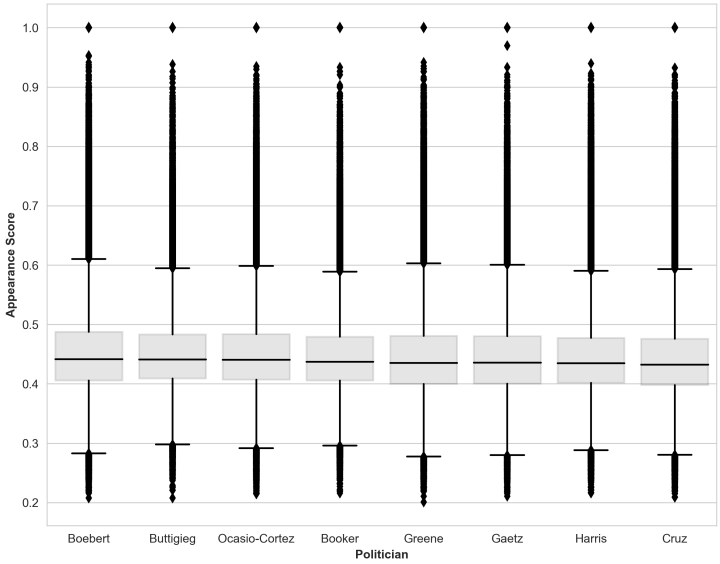


FIGURE L.1: Appearance score distribution.

	Boebert	Buttigieg	Ocasio-Cortez	Booker	Greene	Gaetz	Harris	Cruz
count	664,000	252,758	541,565	233,667	528,678	326,805	496,200	592,794
mean	0.455	0.455	0.453	0.45	0.449	0.448	0.448	0.445
std	0.081	0.076	0.078	0.075	0.081	0.08	0.078	0.077
min	0.208	0.208	0.215	0.216	0.201	0.21	0.216	0.209
25%	0.406	0.409	0.407	0.406	0.4	0.4	0.402	0.398
50%	0.441	0.441	0.44	0.437	0.435	0.436	0.435	0.432
75%	0.487	0.483	0.484	0.479	0.481	0.48	0.477	0.476
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE L.1: Values for boxplot L.1.

Approach 2:

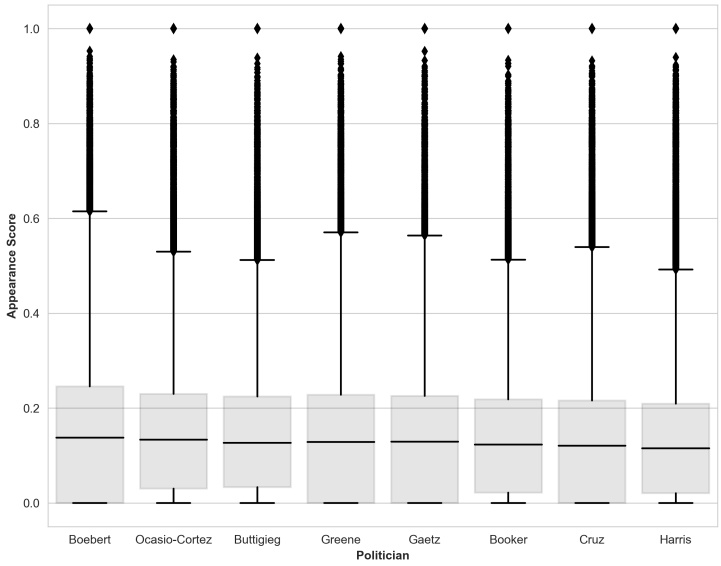


FIGURE L.2: Appearance scores with $\delta = 0.5$.

	Boebert	Ocasio-Cortez	Buttigieg	Gaetz	Greene	Booker	Cruz	Harris
count	664,000	541,565	252,758	326,805	528,678	233,667	592,794	496,200
mean	0.163	0.157	0.156	0.154	0.154	0.149	0.146	0.145
std	0.159	0.151	0.154	0.153	0.155	0.149	0.148	0.148
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.03	0.033	0.0	0.0	0.022	0.0	0.021
50%	0.137	0.133	0.127	0.129	0.129	0.123	0.121	0.115
75%	0.246	0.23	0.225	0.226	0.228	0.218	0.216	0.209
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE L.2: Values for boxplot L.2.

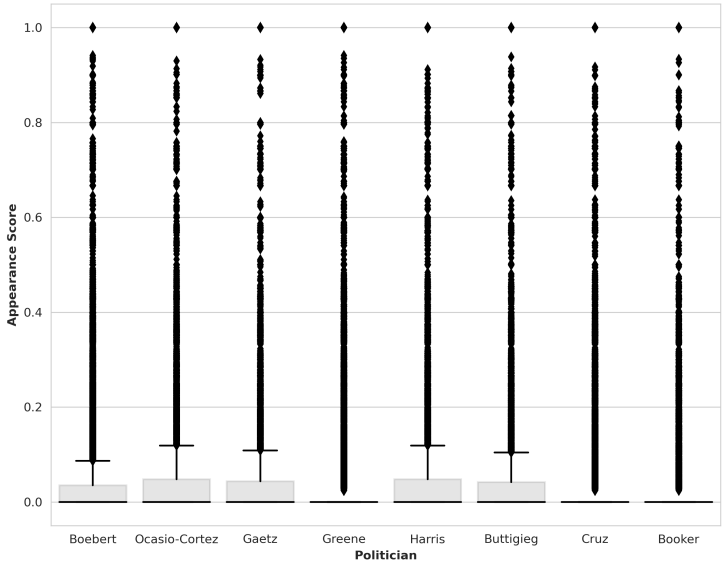


FIGURE L.3: Appearance scores with threshold $\delta = 0.7$. The corresponding statistics for this plot is in table L.3.

	Boebert	**Ocasio-Cortez**	**Gaetz**	**Greene**	**Harris**	**Buttigieg**	**Cruz**	**Booker**
count	663,961	541,527	326,774	528,632	496,240	252,740	592,748	232,653
mean	0.04499	0.04434	0.4369	0.04274	0.04240	0.04073	0.03934	0.03875
std	0.10937	0.10650	0.10728	0.10633	0.10398	0.10253	0.09974	0.09802
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75%	0.03474	0.04762	0.04339	0.00000	0.04762	0.04167	0.00000	0.00000
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE L.3: Values for boxplot L.3.

Approach 3:

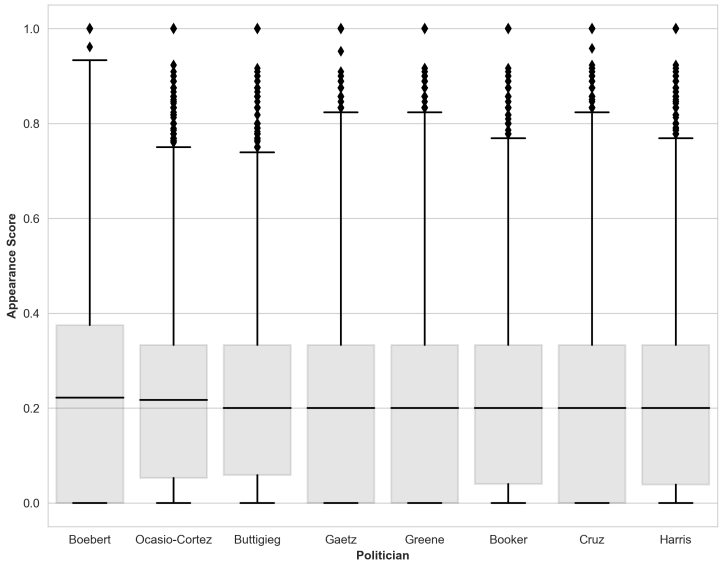


FIGURE L.4: Appearance scores with $\delta = 0.5$.

	Boebert	Ocasio-Cortez	Buttigieg	Gaetz	Greene	Booker	Cruz	Harris
count	664,000	541,565	252,758	326,805	528,678	233,667	592,794	496,200
mean	0.255	0.245	0.245	0.239	0.239	0.235	0.229	0.224
std	0.234	0.221	0.23	0.225	0.226	0.223	0.219	0.214
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.053	0.059	0.0	0.0	0.04	0.0	0.038
50%	0.222	0.217	0.2	0.2	0.2	0.2	0.2	0.2
75%	0.375	0.333	0.333	0.333	0.333	0.333	0.333	0.333
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE L.4: Values for boxplot L.4.

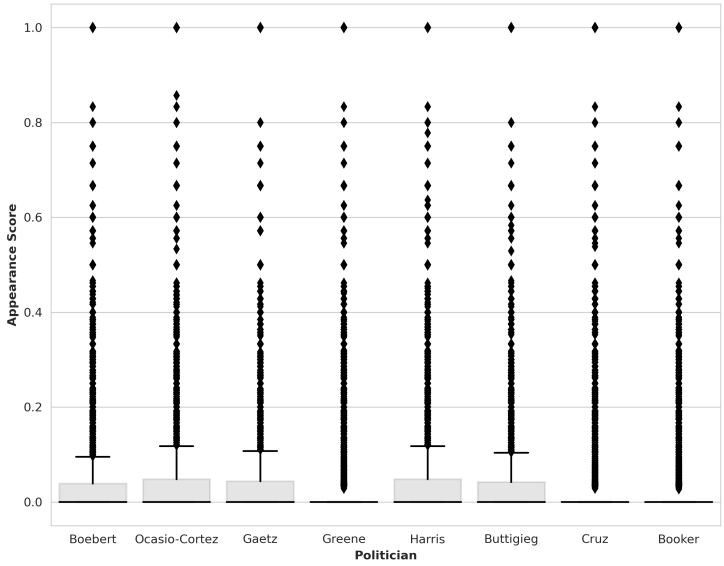


FIGURE L.5: Appearance scores with $\delta = 0.7$.

	Boebert	Ocasio-Cortez	Gaetz	Greene	Harris	Buttigieg	Cruz	Booker
count	663,961	541,527	326,774	528,632	496,240	252,740	592,748	233,653
mean	0.04582	0.04495	0.04462	0.04344	0.04302	0.04133	0.03981	0.03927
std	0.11157	0.10791	0.10992	0.10835	0.10536	0.10427	0.10102	0.09931
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75%	0.03846	0.04762	0.04348	0.00000	0.04762	0.04167	0.00000	0.00000
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE L.5: Values for boxplot L.5.

L.2 Party compared

Approach 1:

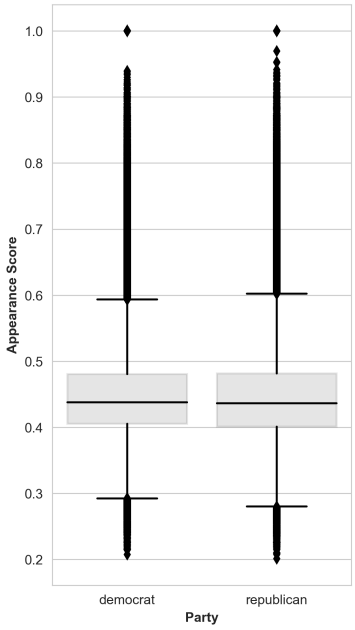


FIGURE L.6: Appearance scores.

	Democrat	Republican
count	1,524,190	2,112,277
mean	0.451	0.45
std	0.077	0.08
min	0.208	0.201
25%	0.405	0.401
50%	0.438	0.436
75%	0.48	0.482
max	1.0	1.0

TABLE L.6: Values for boxplot L.6.

Approach 2:

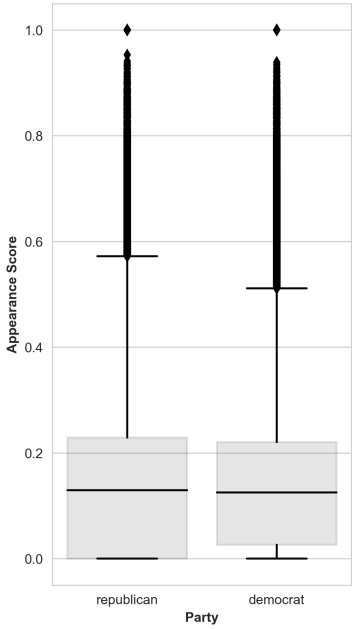


FIGURE L.7: Appearance scores with $\delta = 0.5$.

	Republican	Democrat
count	2,112,277	1,524,190
mean	0.155	0.152
std	0.154	0.15
min	0.0	0.0
25%	0.0	0.026
50%	0.129	0.125
75%	0.229	0.22
max	1.0	1.0

TABLE L.7: Values for boxplot L.7.

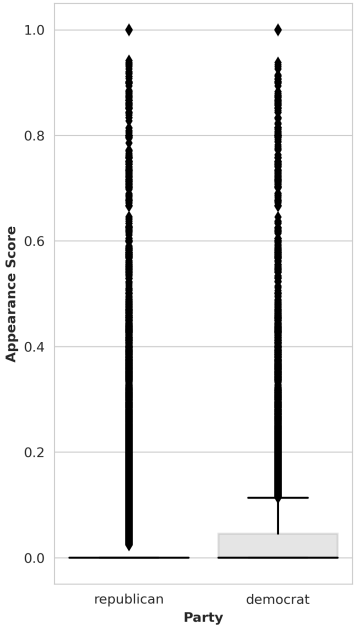


FIGURE L.8: Appearance scores with $\delta = 0.7$.

	Republican	Democrat
count	2,112,115	1,524,160
mean	0.04264	0.04225
std	0.10568	0.10378
min	0.0	0.0
25%	0.0	0.0
50%	0.0	0.0
75%	0.00000	0.04545
max	1.0	1.0

TABLE L.8: Values for boxplot L.8.

Approach 3:

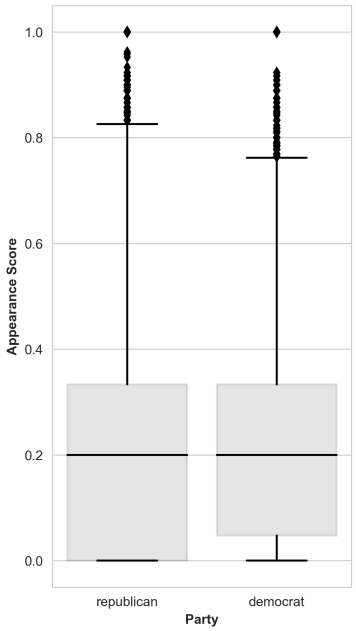


FIGURE L.9: Appearance scores with $\delta = 0.5$.

	Republican	Democrat
count	2,112,277	1,524,190
mean	0.155	0.152
std	0.154	0.15
min	0.0	0.0
25%	0.0	0.026
50%	0.129	0.125
75%	0.229	0.22
max	1.0	1.0

TABLE L.9: Values for boxplot L.9.

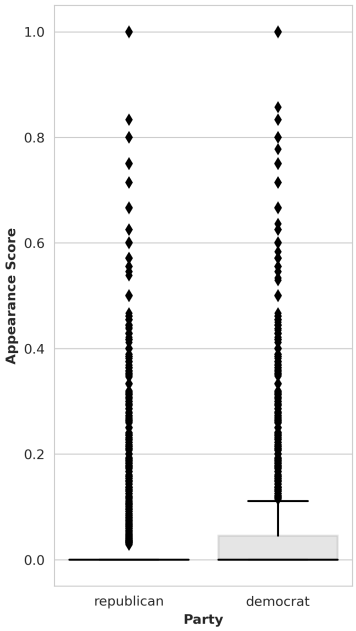


FIGURE L.10: Appearance scores with $\delta = 0.7$.

	Republican	Democrat
count	2,112,115	1,524,160
mean	0.04335	0.04285
std	0.10766	0.10522
min	0.0	0.0
25%	0.0	0.0
50%	0.0	0.0
75%	0.00000	0.04545
max	1.0	1.0

TABLE L.10: Values for boxplot L.10.

L.3 Age compared

Approach 1:

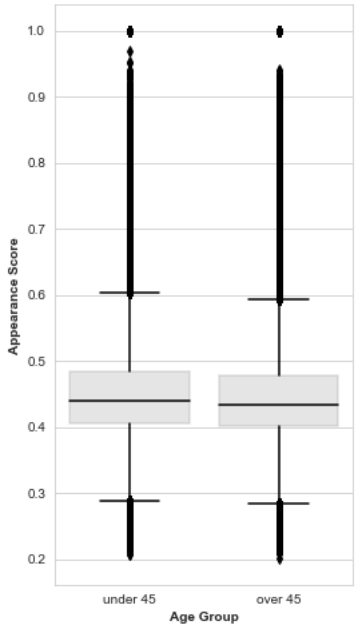


FIGURE L.11: Appearance scores.

	Under 45	Over 45
count	1,785,128	1,851,339
mean	0.453	0.447
std	0.08	0.078
min	0.208	0.201
25%	0.405	0.4
50%	0.44	0.434
75%	0.484	0.478
max	1.0	1.0

TABLE L.11: Values for boxplot L.11.

Approach 2:

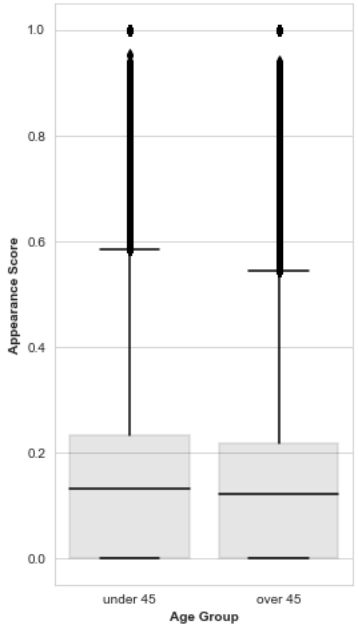


FIGURE L.12: Appearance scores with $\delta = 0.5$.

	Under 45	Over 45
count	1,785,128	1,851,339
mean	0.159	0.148
std	0.155	0.15
min	0.0	0.0
25%	0.0	0.0
50%	0.132	0.122
75%	0.234	0.218
max	1.0	1.0

TABLE L.12: Values for boxplot L.12.

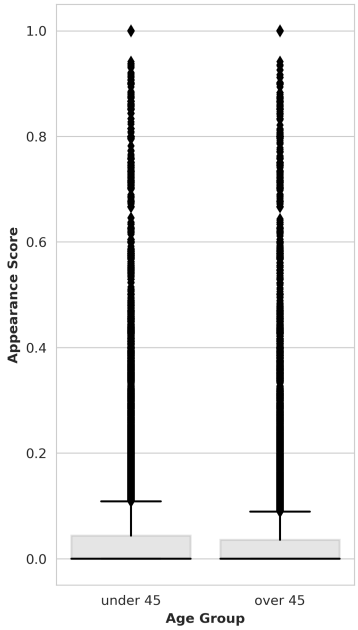


FIGURE L.13: Appearance scores with $\delta = 0.7$.

	Under 45	Over 45
count	1,785,002	1,851,273
mean	0.04395	0.04106
std	0.10718	0.10260
min	0.0	0.0
25%	0.0	0.0
50%	0.0	0.0
75%	0.04348	0.03560
max	1.0	1.0

TABLE L.13: Values for boxplot L.13.

Approach 3:

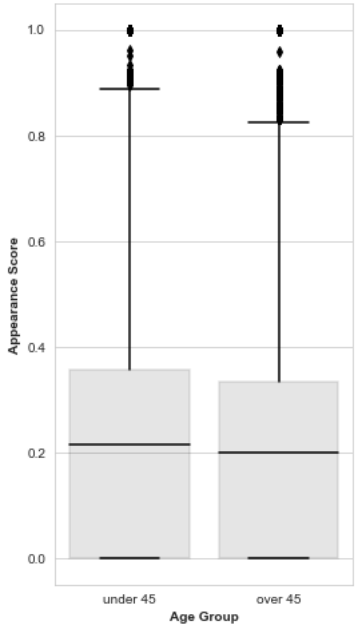


FIGURE L.14: Appearance scores with $\delta = 0.5$.

	Under 45	Over 45
count	1,785,128	1,851,339
mean	0.247	0.231
std	0.228	0.22
min	0.0	0.0
25%	0.0	0.0
50%	0.214	0.2
75%	0.357	0.333
max	1.0	1.0

TABLE L.14: Values for boxplot L.14.

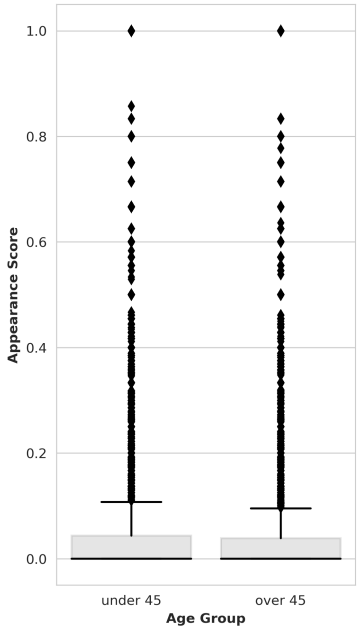


FIGURE L.15: Appearance scores with $\delta = 0.7$.

	Under 45	Over 45
count	1,785,002	1,851,273
mean	0.04470	0.04164
std	0.10916	0.10413
min	0.0	0.0
25%	0.0	0.0
50%	0.0	0.0
75%	0.04348	0.03846
max	1.0	1.0

TABLE L.15: Values for boxplot L.15.

L.3.1 Overview of approach 2 and 3 with $\delta = 0.7$

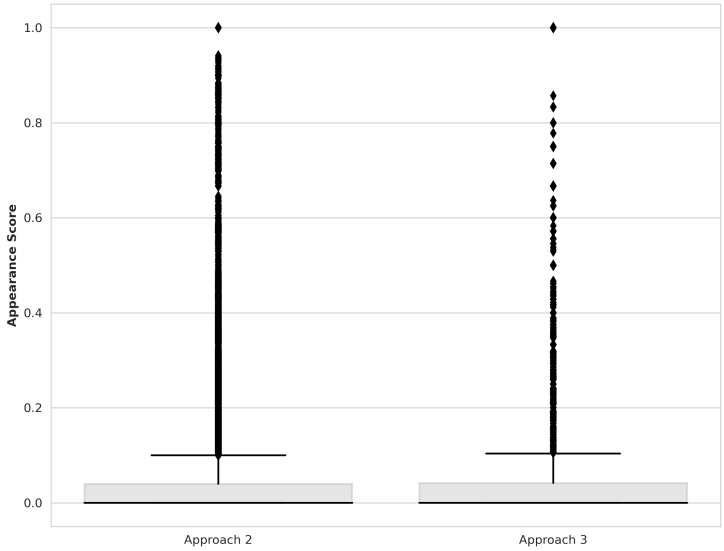


FIGURE L.16: Appearance scores with $\delta = 0.7$.

	Approach 2	Approach 3
count	3,636,275	3,636,275
mean	0.042	0.043
std	0.105	0.107
min	0.0	0.0
25%	0.0	0.0
50%	0.0	0.0
75%	0.040	0.042
max	1.0	1.0

TABLE L.16: Values for boxplot L.16.

M VADER Results

	Negative	Neutral	Positive	Total	Overall Average
All	584,755	366,215	590,163	1,541,133	
Average	-0.53043	0.0001	0.50492		-0.00788
%	37.9%	23.8%	38.3%	100%	
Female	365,829	229,848	364,678	960,355	
Average	-0.52763	0.0001	0.49992		-0.01113
%	38.1%	23.9%	38.0%	100%	
Male	218,926	136,367	225,485	580,778	
Average	-0.5351	0.0000	0.51301		-0.00251
%	37.7%	23.5%	38.8%	100%	

TABLE M.1: Sentiment statistic on tweets with appearance scores equal to or over the mean at 15.3%.

Bibliography

ABCNews. (2021). *Pete buttigieg defends paternity leave* [Accessed 11th April 2023]. https://abcnews.go.com/Politics/pete-buttigieg-defends-paternity-leave-supply-chain-issues/story?id=80670846

Adnana, M. M. J., Hemmje, M. L., & Kaufmannc, M. A. (2021). Social media mining to study social user group by visualizing tweet clusters using word2vec, pca and k-means.

Anspach, N. M. (2017). The new personal influence: How our facebook friends influence the news we read. *Political communication*, *34*(4), 590–606.

BBCNews. (2021). *Matt gaetz: Why this trump ally is fighting for his political life* [Accessed 21th May 2023]. https://www.bbc.com/news/world-us-canada-56608178

Boebert, L. (2023). *Biography* [Accessed 10th April 2023]. https://boebert.house.gov/about/biography

Campaign, H. R. (2021). *Secretary pete buttigieg makes history as first openly lgbtq, senate-confirmed person to lead a department* [Accessed 11th April 2023]. https://www.hrc.org/press-releases/secretary-pete-buttigieg-makes-history-as-first-openly-lgbtq-senate-confirmed-person-to-lead-a-department

Cassese, E. C., & Holman, M. R. (2018). Party and gender stereotypes in campaign attacks. *Political Behavior*, *40*, 785–807.

Chen, Y., Zhou, Y., Zhu, S., & Xu, H. (2012). Detecting offensive language in social media to protect adolescent online safety. *International Conference on Privacy, Security, Risk and Trust (PASSAT)*. https://doi.org/10.1109/SocialCom-PASSAT.2012.55

Dizikes, P. (2018). *Study: On twitter, false news travels faster than true stories* [Accessed 23th May 2023]. https://news.mit.edu/2018/study-twitter-false-news-travels-faster-true-stories-0308

Egger, R., & Yu, J. (2022). A topic todeling komparison between lda, nmf, top2vec, and bertopic to demystify twitter posts. *Frontiers in Sociology*, *7*.

fastText. (2022). *Word representations* [Acessed 24th April 2023]. https://fasttext.cc/docs/en/unsupervised-tutorial.html

Foote, J. (2021). *Twitter v2 full archive search* [Acessed 29th May 2023]. https://github.com/jdfoote/Intro-to-Programming-and-Data-Science/blob/fall2021/extra_topics/twitter_v2_example.ipynb

Gaydhani, A., Doma, V., Kendre, S., & Bhagwat, L. (2018). Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. *arXiv preprint arXiv:1809.08651*.

Haddi, E., Liu, X., & Shi, Y. (2013). The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, *17*, 26–32. https://doi.org/10.1016/j.procs.2013.05.005

Hong, L., & Davison, B. D. (2010). Empirical study of topic modeling in twitter. *Proceedings of the first workshop on social media analytics*, 80–88.

Hutto, C., & Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the international AAAI conference on web and social media*, *8*(1), 216–225.

Hutto, C., & Gilbert, E. (2022). *Vadersentiment* [Accessed 3rd May 2023]. https://github.com/cjhutto/vaderSentiment#citation-information

Intercept, T. (2019). *Kamala harris wants to be president. but what about her right-wing past?* [Accessed 28th May 2023]. https://theintercept.com/2019/01/31/kamala-harris-and-the-myth-of-a-progressive-cop/

İrsoy, O., Benton, A., & Stratos, K. (2020). Corrected cbow performs as well as skip-gram. *arXiv preprint arXiv:2012.15332*.

Jackson, L. A. (1992). Theoretical perspectives on gender-appearence relationship: The sociobiological and sociocultural perspectives. In *Physical appearance and gender: Sociobiological and sociocultural perspectives*. State University of New York Press.

LosAngelesSentinel. (2018). *Senator kamala harris won't take pac money* [Accessed 28th May 2023]. https://lasentinel.net/senator-kamala-harris-wont-take-pac-money.html

McCormick, C. (2016). Word2vec tutorial-the skip-gram model. *Apr-2016.[Online]. Available: http://mccormickml.co tutorial-the-skip-gram-model*.

MSNBC. (2020). *Cory booker: Kamala harris is my sister* [Accessed 28th May 2023]. https://www.msnbc.com/all-in/watch/cory-booker-kamala-harris-is-my-sister-90042949736

Naylor, B. (2020). *Cory booker drops out of presidential race* [Accessed 28th May 2023]. https://www.npr.org/2020/01/13/795874596/cory-booker-drops-out-of-presidential-race

Ocasio-Cortez, A. (2023). *Aoc's platform* [Accessed 20th May 2023]. https://www.ocasiocortez.com/issues

PrincetonUniversity. (2023). *Wordnet: A lexical database for english* [Accessed 22th May 2023]. https://wordnet.princeton.edu/

Rajaraman, A., & Ullman, J. D. (2011). Data mining. In *Mining of massive datasets* (pp. 1–17). Cambridge University Press. https://doi.org/10.1017/CBO9781139058452.002

Řehůřek, R. (2022). *Latent dirichlet allocation, gensim* [Acessed 19th April 2023]. https://radimrehurek.com/gensim/models/ldamodel.html

Rong, X. (2014). Word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

StatistaResearchDepartment. (2023a). *Representatives in the united states congress from 1975 to 2023, by gender* [Accessed 24th May 2023]. https://www.statista.com/statistics/198341/representatives-in-the-us-congress-by-gender-since-1975/

StatistaResearchDepartment. (2023b). *Senators in the united states congress from 1975 to 2023, by gender* [Accessed 24th May 2023]. https://www.statista.com/statistics/198423/senators-in-the-us-congress-by-gender-since-1975/

TheNewYorkTimes. (2020). *Pete buttigieg drops out of democratic presidential race* [Acessed 29th March 2023]. https://www.nytimes.com/2020/03/01/us/politics/pete-buttigieg-drops-out.html

TheTrendSpotter. (2023). *50 hottest female celebrities in the world today* [Accessed 12th May 2023]. https://www.thetrendspotter.net/hottest-female-celebrities/

Wikipedia. (2022a). *Dbscan* [Acessed 24th April 2023]. https://en.wikipedia.org/wiki/DBSCAN

Wikipedia. (2022b). *Lemmatisation* [Acessed 26th Febuary 2023]. https://en.wikipedia.org/wiki/Lemmatisation

Wikipedia. (2023a). *Alexandria ocasio-cortez* [Accessed 20th May 2023]. https://en.wikipedia.org/wiki/Alexandria_Ocasio-Cortez

Wikipedia. (2023b). *Antifa* [Accessed 28th May 2023]. https://en.wikipedia.org/wiki/Antifa_(United_States)

Wikipedia. (2023c). *Attorney general of california* [Accessed 28th May 2023]. https://en.wikipedia.org/wiki/Attorney_General_of_California

Wikipedia. (2023d). *Cory booker* [Accessed 25th May 2023]. https://en.wikipedia.org/wiki/Cory_Booker

Wikipedia. (2023e). *Cosine similarity* [Accessed 1st May 2023]. https://en.wikipedia.org/wiki/Cosine_similarity

Wikipedia. (2023f). *Kamala harris* [Accessed 26th May 2023]. https://en.wikipedia.org/wiki/Kamala_Harris

Wikipedia. (2023g). *Lauren boebert* [Accessed 10th April 2023]. https://en.wikipedia.org/wiki/Lauren_Boebert

Wikipedia. (2023h). *List of emoticons* [Accessed 2nd May 2023]. https://en.wikipedia.org/wiki/List_of_emoticons#Western

Wikipedia. (2023i). *Majorie taylor greene* [Accessed 25th May 2023]. https://en.wikipedia.org/wiki/Marjorie_Taylor_Greene
Wikipedia. (2023j). *Mann–whitney u test* [Acessed 6th May 2023]. https://en.wikipedia.org/wiki/Mann%E2%80%93Whitney_U_test
Wikipedia. (2023k). *Matt gaetz* [Accessed 21th May 2023]. https://en.wikipedia.org/wiki/Matt_Gaetz#cite_note-16
Wikipedia. (2023l). *N-gram* [Accessed 2nd May 2023]. https://en.wikipedia.org/wiki/N-gram
Wikipedia. (2023m). *P-value* [Acessed 6th May 2023]. https://en.wikipedia.org/wiki/P-value
Wikipedia. (2023n). *Pete buttigieg* [Accessed 29th March 2023]. https://en.wikipedia.org/wiki/Pete_Buttigieg
Wikipedia. (2023o). *Political positions of kamala harris* [Accessed 28th May 2023]. https://en.wikipedia.org/wiki/Political_positions_of_Kamala_Harris
Wikipedia. (2023p). *Qanon* [Acessed 29 May 2023]. https://en.wikipedia.org/wiki/QAnon
Wikipedia. (2023q). *Ted cruz* [Accessed 16th April 2023]. https://en.wikipedia.org/wiki/Ted_Cruz
Wikipedia. (2023r). *Times 100* [Accessed 28th May 2023]. https://en.wikipedia.org/wiki/Time_100
Wikipedia. (2023s). *Twitter suspensions* [Accessed 25th May 2023]. https://en.wikipedia.org/wiki/Twitter_suspensions#List_of_notable_suspensions
Wikipedia. (2023t). *United states house of representatives* [Accessed 20th May 2023]. https://en.wikipedia.org/wiki/United_States_House_of_Representatives
Wikipedia. (2023u). *United states senate* [Accessed 20th May 2023]. https://en.wikipedia.org/wiki/United_States_Senate
Williams, G. (2011). Data mining with rattle and r. Springer. https://doi.org/10.1007/978-1-4419-9890-3
Zirn, C., Glavaš, G., Nanni, F., Eichorts, J., & Stuckenschmidt, H. (2016). Classifying topics and detecting topic shifts in political manifestos [Online-Ressource]. In D. Širinić (Ed.), *Poltext 2016 : The international conference on the advances in computational analysis of political text : Proceedings of the conference* (pp. 88–93). University of Zagreb. https://madoc.bib.uni-mannheim.de/41552/