Allegheny College
Department of Computer Science

Senior Thesis

Estimating the Political Stance of Twitter Users by their Tweets

by

Zachary Leonardo

ALLEGHENY COLLEGE

COMPUTER SCIENCE

Project Supervisor: **Janyl Jumadinova**
Co-Supervisor: **Aravind Mohan**

April 9, 2020

Abstract

This paper describes an approach to build a tool to estimate the political orientation of any public, Twitter account. The tool, written in Python, accesses the Twitter API via the tweepy library, and pulls recent tweets of a specified user. Using two trained supervised machine learning models, a subset of tweets containing only political subjectivity is extracted. The set is then sent through a sentiment analysis classification algorithm, where a polarity ratio will be determined based on a cross reference between overall sentiment and mention of active politicians. Using this ratio, each tweet can then be classified as either expressing a left, right, or undefined polarity. The accuracy of the tool can be evaluated on current politicians' social media accounts who embody a strong polarization. Additionally, the accuracy of both supervised learning models can be visualized and compared through multiple metrics in a classification report.

Acknowledgment

There are many people I would like to thank for all the support and guidance I've gotten throughout this beast of a project. This was not an easy accomplishment, and it wouldn't have been possible without the helpful encouragement and feedback I received. I need to first and foremost thank Professors Janyl Jumadinova and Aravind Mohan for not only being my readers on this project, but for guiding me thorough the many ups and downs I experienced along the way. On a similar note, I would like to thank Allegheny College and its Department of Computer Science for building the technical and personal skills that have molded me into the student I am today. A special thank you for Jordan Durci for the insightful peer reviews of my multiple drafts and keeping me on track through the many deadlines.

A loving thank you to Alex Butler, Andrew Everitt, and Simon Burrows for being my support group of friends, roommates, and fellow computer science students who were going through the same challenges as I was. On a similar note, I would also like to thank the rest of my roommates and friends for the constant support and respect I've been given over the past four years. College would never be the same without any of you, and it deeply saddens me that our last semester together was cut short.

A huge thank you to my wonderful girlfriend Hannah Kitchen for always encouraging me and raising my spirits throughout this project and school year as a whole. You have been so incredibly supportive, I wish we were able to celebrate finally being de-comped together. A final thank you to my loving family for always being there for me and helping me to become the person I want to be. Your undeniable love and affection has given me the power to not only finish this project, but to continually better myself, and for that I am forever grateful.

Abbreviations

API	application programming interface
NLP	natural language processing
CI	continuous integration
SVM	support vector machines
CRF	conditional random fields
SA	sentiment analysis
ABSA	abstract based sentiment analysis
NB	naive bayes
TFIDF	term frequency - inverse document frequency

Glossary

RESTful API	A RESTful API is an application program interface (API) that uses HTTP requests to get, put, post and delete data.

I would like to dedicate this project to my cat,
Bruno

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The aforementioned approach to build a command line tool that estimates a person's political affiliation encompasses a wide range of scientific theories and technical components. This project implements the complex ideas of social media metadata and textual sentiment analysis techniques to achieve the most accurate stance detection system possible within the time constraints of the senior thesis project. The final product will hopefully be recognized as an innovative way to improve the quality of life and personal amiability within online political discussion. Each aspect of the project including its background components, overall motivation, current state of the art, and end goals will be discussed in this section.

There has been no greater innovation that has had a larger impact on human connectivity than the ongoing establishment of social media in our ever-changing society. The freedom of expression that social media allows for is unmatched to any other place on earth. While our rights as citizens grant us similar expressions of thought, the interconnectivity of the internet allows these thoughts to be communicated far and wide. This also opens the door for the thoughts and opinions of anyone with an account to be heard by an audience, even if it's for the wrong reasons. Consequently, as more and more people become connected to a social media platform, the accumulation of user meta-data follows suit. This continuous process is what has developed the seemingly endless pool of personal opinions and information, begging to be utilized, that is today's user metadata.

Outside of basic peer to peer communication, social media has transformed the ways its users can interact with companies, organizations, and the government for a variety of reasons. Many of which would make sense to an everyday user, such as following news pages, or scrolling past an advertisement. However, there is a near unlimited influx of user data being collected from every interaction the user has with the platform. This data can then be sold to advertisers, or stored and analyzed without explicitly revealing so to the user. The companies of these applications want to understand their user's feelings towards certain products and services, or find out what hobbies they like or politics they follow. The intent is to refine overall user experience by offering personalized advertisements, and new innovative features to improve relevancy.

Even though this data is being collected with the purpose of monetizing social content, most large scale social media platforms will allow authorized access to any

user with a valid reason. This access can come in the forms of past data sets, real-time analytics, and even API's for application development. These tools can be taken advantage of by a wide set of techniques to examine a user's metadata, and extract their preferences. In this respect, Twitter has become a popular micro-blogging platform which provides users with a textual based way to express their thoughts and opinions. The platform offers an API that allows developers to access real time data, or publish live content. With this feature in mind, it comes at no surprise that Twitter is a valuable tool for public data research. It is for these reasons that Twitter was the chosen platform for analysis in this project.

Twitter is a rich source of cheap and plentiful public data. The platform serves roughly 139 million users everyday, 69 million of which are from the United States. For some demographics, 37% of all Twitter users are aged 18 to 29, in fact 38% of all Americans in this age range use Twitter [5]. Even with the obvious attention from a younger crowd, there are still a wide variety of active Twitter users from older age groups. Surprisingly, the total amount of users in the United States only make up 21% of Twitter's total user pool globally. This means there is a nearly unlimited amount of data being created every second, which is an invaluable resource for social data science.

Twitter is also a powerful information communication tool as well. For every session a user has on the platform they will interact with thousands of other users, whether they know it or not. Depending on how influential the user is, or how many followers they have, a piece of information can be spread to a countless number of other users. This includes those outside of the original poster's network as well. Twitter's capacity for information spreading has since attracted the attention of media companies, online advertisers, and political groups, all who wish to capitalize on the platform's enormous potential for social networking. A daily Twitter user will utilize the platform as a medium for consuming news, grazing entertaining content, reviewing new products, and sometimes engaging in political discussion. The ability for someone or some entity to express these opinions, and share it with a far-reaching audience is extremely important when it comes to information spreading in an advertising, political, or social science perspective.

It is also crucial to understand the downfall of Twitter's powerful information spreading abilities. The rise of disinformation has tainted the quality of online discourse on every popular social media platform since their inception. Now more commonly referred to as "fake news", disinformation is essentially the spread of false content with the intent to deceive, mislead, or manipulate. Separate from misinformation, which is still false but spread unintentionally, disinformation in the digital age is used as a weapon for political fear-mongering to widen social fissures and subvert democracy. Even when major social media companies are constantly under pressure to take action against these nefarious tactics, the spread of disinformation is supplemented by each platform's low barriers to communication and follower models. In order to properly address the issue, social media platforms need to fundamentally reevaluate how they allow users to connect, and continue filtering out fake content to improve the quality of online information spreading.

Besides the multitude of possibilities for marketing and advertising, the dynamics of Twitter's following mechanism enables users to keep up with any other user without reciprocation. This is important to note when analyzing the tendency of users to seek

out people of similar socially significant qualities or ideals. This is know as the homophily tendency, and has already been researched in existing work involving Twitter [24]. It is the combination of this societal tendency and the current polarized state of American politics that influences political Twitter discourse and amplifies division. Recent work has proposed the possible implications of Twitter influencing political extremism by encapsulating users in "echo chambers" with constant exposure to similar thoughts and discussions [7]. While it is not the goal of the described tool, the motivation for this work comes from the prominent and influential role social media plays on the political discourse in the United States.

1.1 Motivation

The research surrounding the tool described in this paper comes at a polarizing time for politics in the United States. There is a widening gap between left and right leaning political ideologies and social media discourse provokes this division even more. Uncovering the reasons behind such strong partisan polarization has been the subject of much scholarly attention. As mentioned above, the mechanics of Twitter's following mechanism has been found to amplify division and encourage political extremism. A recent study also highlights observational evidence of the tendency for social media to encourage a willingness to apply dehumanizing metaphors to those of the opposing party [11]. Each side of the spectrum will either blame the other side, or some sort of top-down influence such as putting the blame on the President, or mainstream media. On many occasions, social media coupled with overblown political rhetoric, and sometimes disinformation, is to blame for the day to day radicalization of otherwise ordinary people. However, it is easy to blame social media for its role in polarizing politics, in fact most people don't realize the good that social media has brought to the political sphere.

It is universally believed that higher participation among citizens benefits democratic institutions. Political observers lament the decline in voter turnout across established democracies and view the trend as detrimental to the democratic process as a whole. However, with the advancement of social media and some institutional reform, there is an increased number and variety of access points that citizens can use to influence political outcome. These platforms offer new methods of peer-to-peer involvement among citizens who share political views and want to be active. Live political updates from media companies expose new legislation or diplomacy policies to public perception faster and at a larger volume than in the past. Additionally, many of the political activities offer greater policy content and focus than just simply voting or watching the prime time news. People are actually engaging in political discussion online, helping to raise political awareness and improve involvement in public interest groups, civic associations, or expand their activist influence.

Clearly the spread of information through social media channels has transformed the present day political landscape in both favorable and unfavorable ways. We are at a crossroads where social media must be utilized to improve democracy by involving all of the public in the process, or effectively risk making inequitable policy decisions for our society. Platforms such as Twitter allow innovative ways for voters to get into direct

contact with elected representatives, influential party members, and other political leaders. Furthermore, online exposure enables these political figures or media outlets to be held accountable for spreading misinformation, hypocrisy, or other qualities that affect their public opinion.

Perhaps the most significant benefit of social media in politics is the attainability of real time public analytics. With the large amount of political discussion producing politically charged public data, politicians and their teams of data scientists are able to gauge public opinion more efficiently. While this doesn't directly benefit the everyday user, this allows politicians and political groups to tailor their messages to certain demographics or audiences. A campaign might find one message appropriate for voters under 30 years old and not as effective for those over 60 years old, and so they will customize their messages accordingly. With the addition of some analysis techniques, data can be collected surrounding certain words or topics allowing greater knowledge of where their voters stand on certain issues. Predictions can also be drawn from this kind of data as an alternative to traditional polling methods.

The political involvement in social media is an improvement to the present day political system as it works to connect citizens and their representatives, increase democratic participation, and supply politicians with powerful analytic tools to better their campaigns. These platforms still pose conventional problems with how political discussion is conducted online via fake content or a user's impertinent defamation of other users. Luckily, the possibility of filtering the negative aspects of social media platforms, like Twitter, by custom made software can circumvent the problems associated with online discourse and improve the quality of online politics. It is for this reason why the tool described in this paper came into fruition.

1.2 Applications of Sentiment Analysis

Twitter offers a myriad of helpful and open source software projects that have been created with either the purpose of improving a user's experience of certain aspects of the platform, or supplying them with robust data collection methods and visualizations. Twitter has been built on open source since the beginning of its development, and so the platform encourages open source as an aspect of its culture. The projects highlighted in Twitter's open source community implement a wide variety of data gathering and analysis techniques, but the most common method is sentiment analysis. Sentiment analysis is the computational process of identifying and categorizing opinions expressed in text in order to determine the writer's attitude. This type of analysis technique and Twitter conjoin nicely due to the nature of Twitter's data being mostly textual based. However, the idea behind sentiment analysis has been around for a long time, and has applications outside of the social media world.

Modern uses of sentiment analysis encompass any kind of mining of subjective information from the internet. As the culture of online shopping began to gain traction, so did the rise of online reviews and the need to analyze them through sentiment analysis. Similarly, any kind of online blogs, news articles, comments, and of course social media posts are subject for this kind of analysis. The textual evaluation is done using a variety of techniques including Natural Language Processing (NLP), statistics,

and machine learning methods. There are also different types of sentiment analysis systems that use various procedures to identify the sentiment towards certain aspects, or measure the subjectivity and objectivity of a text. More information about these techniques will be described in detail in the next chapter.

The applications of sentiment analysis technology are broad, powerful, and widely adopted by organizations all around the world. From a business perspective, sentiment analysis is extremely useful for filtering through the mass of information to find valuable customer opinions. By understanding their customer's conversations about their products or services, business can make data driven decisions for more effective, better-targeted actions. Applying sentiment analysis on the name of a brand or product with the intent to track negative or positive trends and address them accordingly is a process usually referred to as social listening. Moreover, these techniques can be applied to an entire industry or specific competitors as well, allowing for better managing of public relation strategies. Shifts in sentiment on social media have been show to correlate with shifts in the stock market, and so it makes sense for business to invest resources in monitoring and maintaining their brand online.

With real time access to actual customer opinions in a basic textual format, it should come at no surprise that Twitter has become an essential piece of brand supervising in the commercial sphere. There are plenty of comprehensive, social monitoring tools of all scales available for Twitter, some of which even work for multiple platforms at once. These tools work by tracking mentions to the user's brand across all social media channels. Twitter's mentioning mechanic allows any users to address specific accounts by their screen names, and subsequently updates the mentioned user via notification. When implementing these tools in a larger scale operation, a business has the ability to systematically track a customer's mention to their brand or product and respond by alerting the correct public relations team. Almost every enterprise level business will implement an automated customer support system in Twitter by tracking their mentions that appear negative, and responding with an automated help message.

Twitter and sentiment analysis have plenty of other applications in a data collection and research perspective as well. Since Twitter is a great place to collect real time data, custom data sets can be generated and analyzed with sentiment analysis for the research's uses. Specific use cases could include anything from tweets about cancer detection to sports commentary. Any kind of text classification tasks can be made into notable data sets to answer research questions about real life situations. Twitter also offers publicly available data sets of historic tweets for studying trends of sentiment from the past. These data sets are usually built around a certain keyword, a company's name for instance, or a over a specific series of time.

This capability to analyze people's sentiment over time has attracted plenty of attention in the political sphere as well. Political teams can perform research on voter sentiment at the time of an election, or the time leading up to it. There are plenty of scholarly research projects focused around using this historic data to predict the outcomes of elections, or examine the strengths and weaknesses of a campaign. Since all the data is produced by real people expressing their real opinions, the results of the prediction research is often quite accurate. Additionally, by targeting the reactions surrounding campaign statements or foreign policy news, politicians are able to better plan and strategize their campaign for the future. In particular, the Obama

administration used sentiment analysis and Twitter to gauge public opinion to policy announcements and campaign messages ahead of the 2012 presidential election.

Campaigns from both inside the United States and around the world apply research teams to analyze their Twitter voter base in other ways as well. Since political discussion is heavily based around common talking points and other modern policy issues, sentiment analysis on Twitter allows these teams to focus on specific conversations, and determine the public's sentiment towards them. Political campaigns will then take this information and adjust their public exposure to ensure they appeal to the most people possible. On a similar note, Twitter's mention mechanic can be used to calculate the public's awareness of a campaign by analyzing the number of positive and negative mentions they receive. Even if there are large number of negative mentions, research teams can confirm that their message is at least reaching a broad audience and creating media buzz in the process.

The powerful combination of Twitter and sentiment analysis provides rich analytic opportunities for mining subjective textual data from the internet. Modern commercial applications of this technology include brand and industry monitoring as a way to understand customer preferences and make data driven decisions based on them. Twitter acts as a source for rich public data, and when analyzed with sentiment analysis techniques it provides a base for answering real life research questions. Additionally, the ability to focus on tweets from certain points in time appeals to political research teams who use the data to make accurate predictions and analysis of elections. Campaigns will also use the information from voter opinions to adjust their public exposure, and measure the awareness of their campaign messages. There are really endless applications for sentiment analysis on Twitter, which is why the tool described in the paper takes advantage of this robust combination of technology.

1.3 Goals of the Project

There are several goals that the Twitter political stance detection tool described in this research hopes to achieve. Clearly the primary goal is to estimate the political orientation of a Twitter user by their tweets. However, there will be a considerable focus in creating a custom sentiment analysis algorithm with the highest level of accuracy possible. Ideally, this research will provide an end product that is not only appealing to political applications, but apply to advertising and data science tasks as well. Another major objective of the tool is focused on the design and delivery of the software itself. The tool will feature high quality software design with thorough documentation and robust test suits, all publicly available in a GitHub repository.

Twitter's micro-blogging tweet format allows the user to input a 140 character message with no restrictions on format or content. A usual tweet will contain 5KB of total data which is more than 40 times the amount of data that makes up the textual content [22]. While this text format is more fit for sentiment analysis work than a post with media, it is still tricky for Natural Language Processing systems. The challenges comes from cases of improper grammar, emoticons, and slang among others that complicate a tweet's content. Due to the erratic nature of a user's input into their tweets, the text must be parsed through and cleaned before the sentiment

analysis system can be applied.

The largest challenge will be in developing the actual political orientation of a given user. To solve this problem, the tool will extract all the political entities within the cleaned tweet, assign a polarity to the entities by cross referencing with words of far leaning stances. Then using a political tendency metric, classify each user into one of four categories, Left, Right, Center, or Undefined. A detailed description of the political sentiment metric is described in the methods section.

Continued tweaking of the classification algorithm should eventually yield a final product that is useful for both political research scientists and marketing agencies. Assuming a data scientist would want to target a certain political audience for research purposes, they could use the tool to apply a filter for a user's political standing and adjust their data collection accordingly. Similarly, if a marketing agency finds that their message resonates with a certain political standing more than the other, they would use the tool as a metric for reaching that specific audience. Since the final tool will be accessed through a command line interface, the source code would need to be adapted to suit the particular needs of the researcher, but the logic behind the algorithm would stay the same.

The reason behind creating an entire tool that helps encapsulate the process of estimation instead of just releasing the algorithm itself, is so that it will appeal to the everyday Twitter user as well as researchers. Political domineering is a common problem for political discussion among online Twitter threads. People have strong willed opinions about politics, and will overbearingly try to subjugate others into believing what they have to say, as opposed to them forming opinions on their own. As a solution, this tool can act as a screening device for anyone questioning the standing of a particular user and exposing their biases in real time. Of course in order to use the tool, the user would have to have a basic understanding of command line applications, and how to clone repositories. However, the end goal is complete a working and accurate estimation system within the time constraints, and if there is extra time perhaps another user friendly interface like a web application could be implemented. Additionally, the repository will feature thorough documentation explaining required dependencies, exact instructions on how to set up and use the tool, and other pertinent information.

In order to gain quality attention from researchers and everyday Twitter users, the software must be built with quality in mind. Just like other open source projects, the intent is for the tool to be improved upon by anyone who is willing to contribute. With a logical and easy to understand design, developers will be more likely to expand the existing features or add new ones. To help control complexity, semantic and styling guidelines will be enforced through Travis CI, a continuous integration tool. Pull requests will also need to be approved before merging, as well as passing the Travis CI checks. The goal here is to mimic a real life software development environment and produce a high quality end product.

To ensure that the functionality of the tool, and any other additions made through open source contributions fit into the system without without errors, a test suite will be a major feature of final tool. The objective for testing is to prove that every line of the source code does what it is intended to do, as well as evaluating the accuracy of the methods provided in an automated fashion. Since the tool will be written in

Python, it makes the most sense to build the test suite using Pytest library. It is not unrealistic to expect full coverage on a test suite, where test cases will execute every line of source code available. In fact, a fully covered test suite is a sign of good code design and robustness in testing. The percentage of covered code will be displayed front and center in the documentation, indicating the confidence in the tool's functionality, and attracting more attention from developers willing to contribute.

Overall, there are a significant set of goals put forth for the tool described in this research. The most obvious being the ability to estimate a Twitter user's political orientation with a focus on accuracy for the sentiment analysis algorithm. With a high level of accuracy, the tool will hopefully attract attention from political researchers, and data scientists for applications in political and advertising endeavors. Additionally, everyday Twitter users would be able to use the tool as protection against political biases online. Finally, in an effort to get as close to following high quality software development as possible, all source code will follow a logical design with thorough documentation and robust test cases.

Chapter 2

Related Work

As development on the tool described in this paper gets underway, it is crucially important to understand the latest sentiment analysis techniques produced by innovative research in the field. As a response to the growing availability of informal, subjective textual data such as blog posts or product reviews online, the field of sentiment analysis has developed into several specific techniques aimed at solving the different challenges associated with analyzing emotionally charged texts. The various approaches outlined in this chapter have been recognized by multiple researchers in computer science, computational linguistics, data mining, psychology, and even sociology disciplines.

Even with new ground breaking achievements in sentiment analysis technology, the discipline is not a new idea by any means. The original idea of sentiment analysis for text has been around since the 1950s. People have always been interested in the opinion of other's, and often look for them for everything from conducting business to just making every day decisions. However, the internet is the single largest reason for the expansion of the idea into multiple fields and applications. It is now possible to find the opinions of millions of people for just about any topic possible. As mentioned in the previous chapter, social media platforms play a powerful part in supplying this data, especially when acting as an increasingly popular forum for discussion and a source of information. Online blogs supply another significant portion of textual data with recent statistics showing that over 77% of all internet users read blogs with over 5.8 million new blog posts being created each day [5]. The increased accessibility for opinionated text, and a realized importance for its uses, has expanded the subject of study away from a traditionally fact-centric view and allowed for new, adaptable sentiment-aware implementations.

As the science of sentiment analysis continues to innovate, more and more new research studies are designed to take on the latest and emerging challenges. Even though the tool designed in this research embodies a specific technique that has been covered by other researchers numerous times, it is important to understand general sentiment analysis and the theories behind it. Additionally, each technique has its advantages and disadvantages for the tasks it's suited to solve. Therefor by making the connection to the Twitter political polarity tool, a rationale for why the chosen method performed in the tool can be produced.

2.1 General Sentiment Analysis

Personal feelings often seem more primitive than thoughts, yet they constitute a significant portion of fundamental communication between people. Emotions, opinions, and their expression in language not only impact all our daily lives, but have been the subject of research in linguistics and other social sciences for a long time. A book written by two prominent linguists, JR Martin and PR White, suggest that the expression of emotional states in language can be institutionalized into two major categories [12]. The first is the expression of judgement towards other people, and the second is categorization of appreciation. Together, judgement and appreciation capture how we convey our feelings and opinions thorough language, which is the object of study in sentiment analysis. To break it down even more, opinions have the ability to be expressed subjectively or objectively. Sentiment analysis systems focus more on subjective information which is linguistically derived as the expression of belief, emotion, evaluation, or attitude from language [23]. This is in contrast to objective language, which is rooted in facts used to present events or describe the state of the world. This subjective information can be further categorized into two modes either explicit or implicit [13]. An explicit sentence will directly express an opinion such as, *"Today's weather is beautiful"*. On the contrary, an implicit sentence will imply an opinion without directly stating it, *"The package arrived a week late"*. Both are possible to analyze, but a majority of research focuses on explicit since it is naturally easier to extract.

The basic idea of sentiment analysis, as the name implies, sprouted from the fundamental task of analyzing what has come to be know as sentiment. The goal of the majority of sentiment analysis techniques is to measure the subjectivity of expression for an overall message, and classify it as either, positive, negative, or neutral opinions. The separation of these classifications can be thought of as an interpreted range, where there is a distinction between polarity and strength. For example, a person might feel strongly about a product being satisfactory, not particularly good or bad, or weakly about a product being very good [13]. This contrasts to the closely related idea of studying emotion and emotive terms in an attempt to classify them as angry, surprised, fearful etc. While these two approaches generally work in tandem, they are still fundamentally different.

Despite being the baseline task of all sentiment techniques, the process of applying a sentiment classification still proposes its own challenges when it comes to slang or sarcasm. Some recent evaluation metrics even propose both polarities existing together in a mixed message [2]. The analysis performed on these messages can become even more complicated with the inclusion of supervised learning techniques. A majority of popular machine learning systems such as Support vector machines (SVM), Conditional random fields (CRF), and linear regression are mostly utilized in evaluation procedures, but have applications to analysis as well [15, 2]. While these supervised approaches are widely adopted, the biggest drawback is their domain dependent nature [1]. This means that the systems were very likely to perform poorly outside the scope of their original data. On account of this and other flaws, various other approaches have since been investigated to allow for more complex analysis tasks, aspect-based sentiment analysis is one of them.

2.2 Aspect-based Sentiment Analysis

Building off the successes of modern day sentiment analysis systems, aspect-based sentiment analysis provides a more fine tuned sentiment detection system for researchers. The core idea behind aspect-based sentiment analysis, or ABSA in short, is to focus the analysis on certain aspects of textual entities, and define the polarity expressed about each aspect [1]. The textual entities mentioned here simply refer to certain elements of a sentence, commonly proper and improper nouns that a sentiment can be expressed upon. For example, given the sentence *"The food at the local bar was lousy"*, we can identify two distinct aspects: *food* and *local bar*. The sentiment in regard to both of these specific entities would be considered negative, however modern ABSA systems can provide a varying polarity score for each entity as well. This form of sentiment analysis possesses important applications in the commercial domain, due to the increase of user written reviews, and the need to summarize them. Often times, the most important part of determining sentiment is in the expression of a target, such as a person, object, concept, or really anything else that can be described. It is for this reason that ABSA has been gaining increasing importance in recent years.

Similar to traditional sentiment analysis, many implementations of ABSA systems are extremely complex. This encourages the use of automated approaches to solve research specific tasks. A recent study uses a combination of machine learning algorithms such as SVMs and CRFs, along with having bag-of-words, lemmas, bigrams after verbs, and punctuation to create an improved ABSA system [3]. While this implementation achieves good results on aspect category, and aspect polarity detection, the addition of training results with machine learning still proposes the same domain dependent problem as before.

Additional pioneering in ABSA followed the release of the last SemEval competition in 2016, an ongoing competition for evaluating computational semantic analysis systems. A new approach was introduced which utilizes a convolutional neural network for both aspect extraction and aspect analysis [4]. The study aimed to improve granularity at aspect level, and achieved best results in the competition. The deep learning approach was also able to capture both syntactic and semantic features of a text without dealing with feature engineering based on domain knowledge of the data, improving on methods mentioned in the previous section.

While the inclusion of these syntactic recognition features has helped pave the development for the next emerging sentiment analysis task, the challenge associated with building the polarity estimation tool will not utilize a machine learning or deep learning approach. Since data will be extracted from Twitter in real time for every execution of the tool, there will not be an annotated data set that can be used to train a supervised learning model. Additionally, the tool aims to estimate where a person stands in relation to political leaning, not necessarily their sentiment towards certain aspects. To solve the challenge presented, the algorithm will use an customize version of an emerging sentiment approach analysis known as stance detection.

2.3 Stance Detection

Similar to the methods of aspect based sentiment analysis, the goal behind stance detection focuses on detecting the position of the user on some specific spectrum with respect to target entities [1]. The desire for this type of identifying technology arose mainly for implementation in the political sphere, specifically through online political discourse. As we know, Twitter offers a diverse set of data containing personal opinions and specifically polarized political ideologies. It makes sense why the platform, rather the tweets it's users produce, has been the target for most political stance recognition studies.

Similar to approaches taken with other sentiment analysis techniques, and in an effort to improving accuracy above past stance recognition techniques, there are a wide variety of machine learning systems employed across each research method brought forth. In a recent study, researchers implemented three python sentiment analysis libraries on a set of tweets, and compared the results by two machine learning algorithms. One being a Naive Bayes classifier and the other a SVM classifier [6]. In the interest of gauging public opinion about highly contested issues, another recent study proposed a semi-supervised framework using a retweet based label algorithm with high adaptability and accuracy to gauge user stances on certain topics [20]. Much like the pitfalls of other supervised learning methodologies, these studies work from an annotated training data set of downloaded tweets, which is irrelevant to task proposed by the Twitter PoliStand tool. However, their relevancy to political analysis is still important.

More applications of this technology can be employed to predict or explain real political outcomes by studying the sentiment and discourse of online discussions. The results of a sentiment analysis study on both Donald Trump and Hillary Clinton's speeches form the 2016 presidential election, and are leveraged via machine learning methods to explain the final outcome of the election [8]. Making predictions in this way provides plenty of opportunities for political gain, and so the accuracy of the stance detection systems implemented carries significant value. In the 2016 SemEval competition, each team was tasked with creating the most accurate system for detecting Twitter users stances with respect to chosen targets [1]. For the system that was able to outperform all others, researches implemented a linear SVM that leverages word and character n-grams [14]. As you can tell, accuracy is very important to stance detection research, and has received plenty of recent academic interest. A significant factor of most tweet stance detection research focuses on the accuracy of supervised learning systems using annotated data sets, but similar projects using live Twitter data do exist as well. The next task moving forward for the niche field of Twitter stance detection involves gathering real-time data in a clean form for immediate analysis. This is where the PoliStand tool makes its fit.

2.4 Twitter and Ideology Detection

There are a significant set of related research projects involving stance detection analysis that aim to gauge a user's political ideology. The tool proposed in this research will be built with a hybrid stance detection system that uses the sentiment

polarity towards certain political entities within tweets. In the essence of time and efficiency, not every tweet needs to be analyzed. Instead, only the tweets containing these political entities will be passed into the system. A specific user's tweets will be pulled, cleaned, and analyzed for immediate output. Luckily, there are a handful of similar research papers employing systems with real, unedited tweet data, however often provided through a data set.

In one of these studies, researchers defined a political tendency metric that takes into account the polarity of entities related to political parties mentioned in the tweets of a user [18]. While the methods of the proposed tool are similar to those deployed in this study, the tweets are pulled from a data set of 6000 political tweets in Spanish. Another study uses a seven point scale of political tendency in order to identify politically moderate users between neutral and extreme leaning [19]. The researchers identified different stances by mapping them to political keywords within tweets, similar to the method of the proposed tool, however tweets were provided by a novel data set with stances self reported through surveys. In another study, researchers studied the political orientation of tweets regarding the 2017 Turkish constitutional referendum by two methods and compared them. Both utilized an SVM classifier one of which was focused on semantic features, while the other trained on hashtags [25]. This study used an unedited data set of tweets where noise and irrelevancy had to be accounted for, equivalent to the data the proposed tool will collect. This is significant considering their results reported higher performance for systems targeting semantic features over hashtags, which is why the proposed tool will target politically charged words over hashtags.

Chapter 3

Method of Approach

As I continued to research stance detection systems that used tweets to classify a user, I noticed a few similarities between the designs brought forward by various experts in the field. Almost all the systems described some evaluation of various machine learning models combined with a sentiment algorithm for classification. By basing the design of the Polarized tool off the general format of these related systems, a modular, three step design was implemented. Figure 3.1 shows an overview of the Polarized tool's system for solving the classification problem. The system consists of three main modules; training, applying user data, and sentiment classification. The first module is training module, which takes two sample data set of labeled tweets, combines them into a single set, and trains either a Multinomial Naive Bayes or linear SVM to predict the political relevancy of each tweet, depending on the user's choice. Once trained, the second module produces a data set of unlabeled tweets collected in real time from the user of choice, and passes it into the chosen model for labeling. The output, a new data set of only the tweets labeled as political, is then passed on to the third module, the sentiment classification. The final module produces a polarity rating of the user based on a variety of sentiment tasks including noun extraction and sentiment polarity applied on each political tweet. More information about the details of this module is included later on. A final evaluation of the user's polarity classification will then be returned based on the overall average of each tweet's rating.

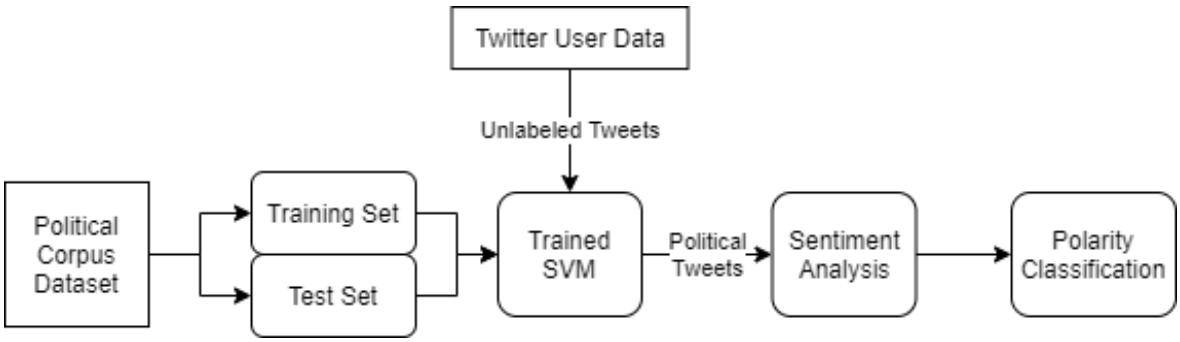

Figure 3.1: System Overview

3.1 Accessories of the Method

When it comes to the technical implementation of the aforementioned system, there a plethora of tools to choose from. As new, open source software continues to expand and innovate, the selection becomes even more difficult. While some tools such as Twitter and its development API are required due to the scope of the project, the specific language and libraries chosen were determined by multiple factors. Clearly, the software's ability to perform its necessary task was the first and most obvious step in narrowing down the list, but it was not enough. The learnability of a language or library by means of available documentation or tutorials was a key component of each selection. Considering the limited amount of time allotted to complete the project, there wasn't much room to experiment with new software unless necessary, and so my previous experience working with a product also played a large role in my selection. It is combination of these reasons, and more, as to why I chose to implement the project in the Python programming language.

After selecting a language, it was necessary to find a way to easily connect my system to the Twitter API. While it is possible to interact with a RESTful API with Python by making requests to a specific URL and parsing the data returned, there is a lot of information and programming expertise required for actual implementation. Luckily, Python offers an unmatched selection of open source libraries that help abstract away the complexity of this interaction. Additionally, I had no prior experience working with the Twitter API, and so a library was a necessity. In this regard, the tweepy library contains all the necessary requirements, and boasts a large selection of other features [21]. While other fine options such as python-twitter or twython were considered, the final decision was made due to the easy to follow documentation and tutorials given on the tweepy website. By using tweepy, the Polarized tool is able to easily search for Twitter users, and return a data set of their most recent 200 tweets, which was the rate limited by the free-tier Twitter development account.

In order to perform the tasks required by the training module, a machine learning model would need to be implemented. Like always, there are a variety of options when it comes to libraries that allow for this data analysis in Python, but no free option comes close to the popularity of scikit-learn [17]. This library helps simplify effective predictive data analysis in an easy to learn, accessible context. In addition, the level of maintained documentation and tutorials available on their website is unmatched, and so the choice was clear. By using multiple supervised learning methods provided by scikit-learn, the Polarized tool is able to not only train an algorithm to determine a tweet's political relevancy, but compare the accuracy of various models with ease.

After training and applying the user data to the selected model, the political tweets must be analyzed and classified based on their political sentiment and other factors. There were plenty of ways to go about developing this, and the choice of software wasn't as important as the design of the algorithm itself. Since all the sentiment analysis tools I could choose from essentially do the same thing, the final decision was made based on solely on my prior experience. Built on the successful Natural Language Toolkit (NLTK), the TextBlob library allows for textual sentiment analysis processing, and encapsulates the common sentiment analysis tasks described in Chapter 2 [9]. While the process behind TextBlob's analysis is fairly simple, it does exactly what it needs

to do in the context of the classification problem, which is covered in detail in section 3.4. By leveraging my previous experience with TextBlob, I was able to simplify the sentiment analysis, and accurately connect it with the overall classification module.

As far as data collection goes, the aforementioned system relies on two sets of data as inputs. Both data sets contain tweet text data that has been pre-processed and cleaned as to avoid any unnecessary formatting issues. As with most supervised learning techniques, the model must be trained with already labeled data. Creating a data set as such would have taken way more time than possible, and so other options needed to be applied. The Twitter Political Corpus provided two data sets containing over 2000 tweets each, and were hand labeled specifically if their topic was political or not [10]. For the first set, randomly selected tweets were chosen between June 1, 2009 and December 31, 2009. Figure 3.2 shows the distribution of politically labeled tweets within this first set. Similarly, the second set of tweets was randomly selected as well, but from a subset of tweets that contained at least one political keyword in each tweet. Figure 3.3 shows the same distribution of politically labeled tweets but within the second set. When comparing both figures, there is a clear majority of non-political tweets in the first set, and a majority of political tweets in the second set. Due to these trends, I started to encounter accuracy problems when training the model because conventional algorithms are often biased towards the majority class. However, by combining both sets into one combined set, I was able to balance the data and restore high prediction accuracy for both classes. The combined set then had a total of 2,290 non-political tweets and 1,714 political tweets, which was far more balanced than the original two by themselves.

In order to evaluate our newly trained model to a real time example, the Polarized tool collects a new test set of tweets from a specified Twitter user. The exact methodology behind this module is explained in Section 3.3. Essentially, these tweets are pre-processed and cleaned using a simple regular expression, and then appended on to the combined Corpus data sets for predictive analysis.

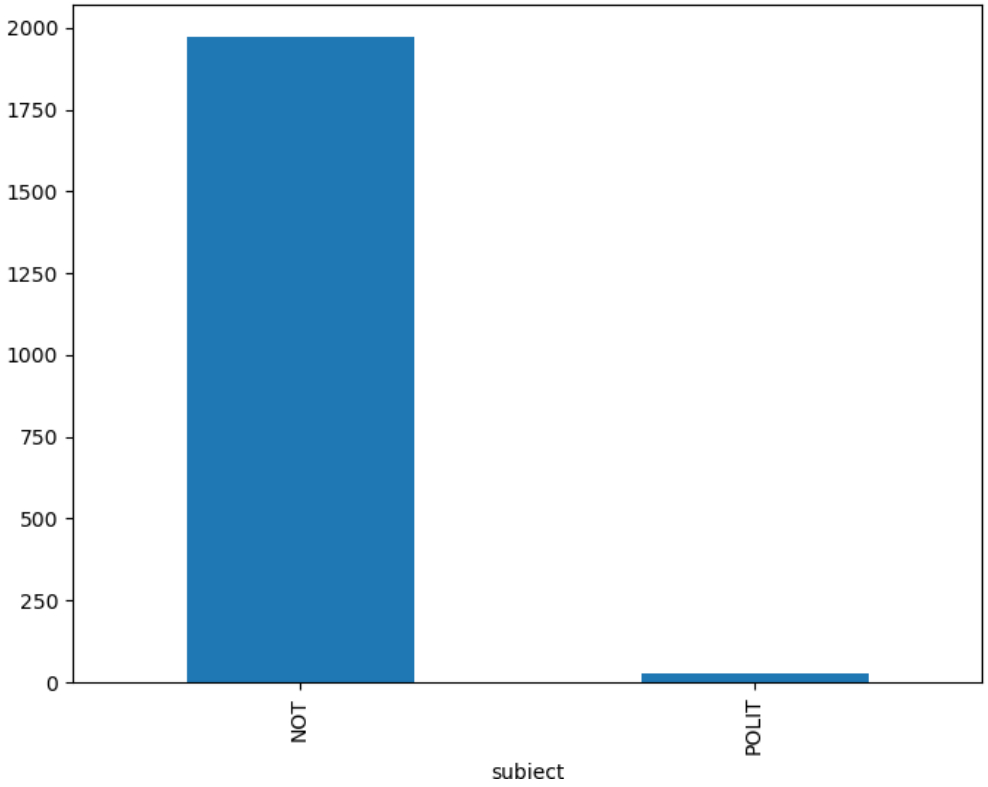

Figure 3.2: General Tweets Data

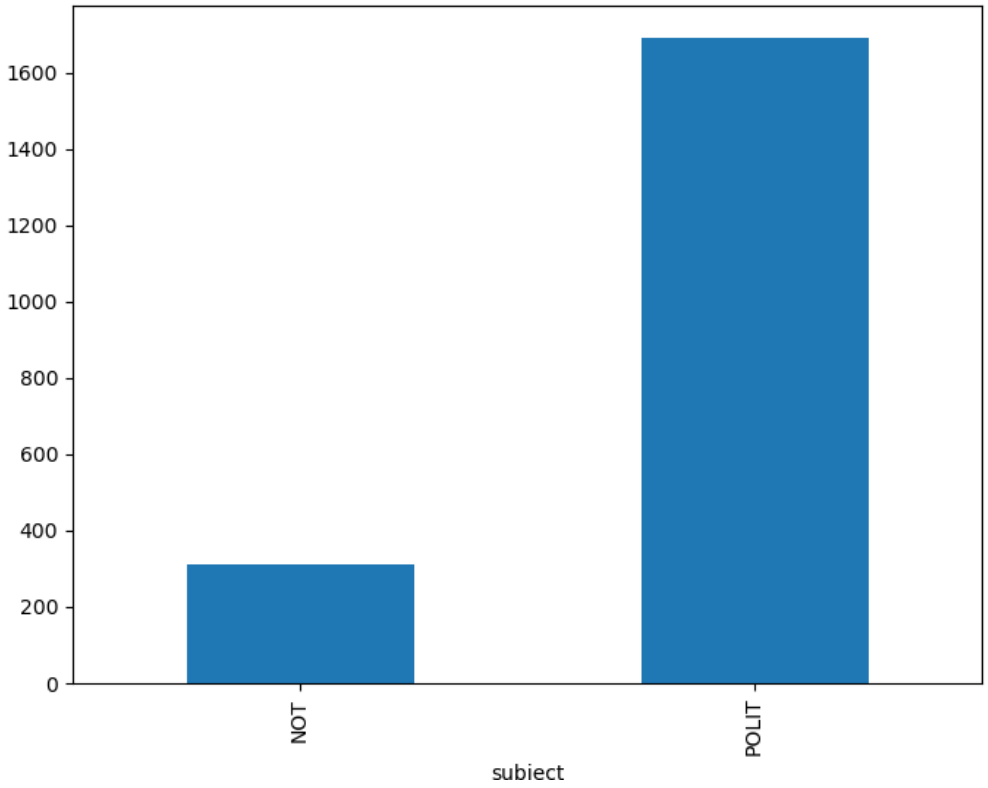

Figure 3.3: Keyword Tweets Data

3.2 Training the Supervised Model

As development on the first module began, it was important to first choose a simple starting model that wouldn't over complicate the task, and then move forward from there. Luckily the scikit-learn documentation was helpful in quickly explaining how to implement these models with a few lines of code. Basing my implementation on similar classification projects and a few other reasons, I decided on a Multinomial Naive Bayes for the first model.

The Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with a "naive" assumption of conditional independence between every pair of features given the value of the target variable. As one of the two classic Naive Bayes variants, the multinomial distribution is parametrized by vectors for each class, and the probability of a feature appearing in sample of a certain class, or in my case the subject. The parameters θ_y are estimated by by a smoothed version of maximum likelihood, which is essentially relative frequency counting.

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Above is the exact algorithm given by scikit-learn [17]. In this algorithm, N_{yi} is the number of times feature i appears in a sample of class y in the training set, and N_y is the total count of all features for class y. While the algorithm itself might be confusing, it is actually simpler than most supervised models, and the code provided by scikit-learn abstracted away much of its complexity.

While a basic Naive Bayes model is a good starting point, there are other options that allow for higher levels of prediction accuracy. Based on the comparative analysis from similar research projects, a linear SVM was frequently rated as the most accurate model for classification tasks. Similar to Naive Bayes, the Support Vector Machines are a set of supervised learning methods used in classification, regressions, and outlier detection. There are a myriad of advantages to using SVM's, but since the algorithm it encapsulates is more complex than that of the Naive Bayes, the results of predictions are frequently more accurate. It is important to note that SVM's do not directly provide probability estimates, but are instead calculated using five-fold cross validation. Just like with the Multinomial Naive Bayes model, scikit-learn allows for quick and easy implementation of a linear SVM with various optional parameters, and so I chose that as the second model for comparative results.

It became apparent after I attempted to fit both models with the original tweet data that textual data required special preparation before it could be used for predictive modeling. Most models expect numerical feature vectors with a fixed size rather than raw text documents with variable length. Each tweet needed to be parsed to remove words in a process called tokenization. The remaining words then needed to be encoded as integers or floating point values for use as input in a process called feature extraction or vectorization. I used the common but effective approach of the bag-of-words model to extract said features form the text. This model is simple in that the presence or frequency of words is taken into consideration, but the order in which they occur is not. Each word is assigned a unique number, and then each tweet can be encoded as

a fixed length vector with the length of the vocabulary of known words. The value in each position in the vector is then filled with the frequency of each word in the encoded text. The degree in which each word is presented in the encoded text can be tweaked through optional parameters passed into the algorithm. Luckily, the scikit-learn library offers easy to use tools to perform both tokenization and feature extraction through the CountVectorizer class or preferably, TfidfVectorizer class. A simple word count is very basic, but doesn't carry enough meaning in the encoded vectors, which is why I chose the more popular method of a TfidfVectorizer. Specifically for each tweet in the data set a measure known as Term Frequency, Inverse Document Frequency (TF-IDF) is calculated. This numerical statistic is intended to reflect how important a word is to the tweet in the overall set. Listing 3.1 below shows the implementation with scikit-learn to calculate a TFIDF vector for each tweet in the data set.

```
# Features = tweet
x = combined_tweets["tweet"]
# Target = subject
y = combined_tweets["subject"]

tfidf = TfidfVectorizer(sublinear_tf=True, min_df=3, norm='l2',
    encoding='latin-1', ngram_range=(1, 2), stop_words='english')

features = tfidf.fit_transform(combined_tweets.tweet).toarray()
print(features.shape)

X_train, X_test, y_train, y_test = sklearn.model_selection.
    train_test_split(x, y, test_size = 0.2)
X_train_tfidf = tfidf.fit_transform(X_train)
```

Listing 3.1: TFIDF Vectorization

As you can see in the example above, the tweet of each entry is set as the feature for labeling, and the target or class is set as the subject of the tweet, which in this case was either political or not. The TfidVectorizer object provided by scikit-learn was then initialized with special parameters. The first, sublinear_df, is set to True to use a logarithmic form for frequency. The min_df parameter, which is arguably the most important, is the minimum number of tweets a word must be present in to be kept, and is currently set to 3. Norm is set to 12 to ensure all feature vectors have a euclidian norm of 1. The ngram_range is set to (1, 2) to included both unigrams and bigrams. Finally, stop_words is set to "english" to remove all common English pronouns and reduce the noise in the feature set. After initializing and customizing the TfidfVectorizer, it is first fitted with all the raw tweet data, and transformed to return the result of (4004, 3011) via the print statement. This means that each of the 4004 tweets are represented by 3011 features, representing the TFIDF score for different unigrams and bigrams found in the tweets. The data set was then randomly split into testing and training subsets using provided scikit-learn functionality. The training subset is set to 80% of the original set, and is used to train the algorithm while the test subset contains the remaining 20%, and is used for evaluating the accuracy of predictions. By fitting the TfidfVectorizer with our training data, it is then ready to be passed into the Multinomial Naive Bayes and Linear SVM classification models.

Listing 3.2 below demonstrates the fitting and prediction of both models and their prediction result of a sample string.

```
# MultinomialNB
clf = MultinomialNB().fit(X_train_tfidf, y_train)
print(clf.predict(tfidf.transform([string])))

# Linear SVM with regulization parameter
clf2 = svm.SVC(kernel="linear", C=2)
clf2.fit(X_train_tfidf, y_train)
print(clf2.predict(tfidf.transform([string])))
```

Listing 3.2: Fitting Both Models and Predicting

As defined in the code example above, there are two classification models being fitted with our vectorized textual data. The first is our basic Multinomial Naive Bayes, and the second is an SVM defined with the SVC (Support Vector Classification) class. By setting the optional kernel parameter to "linear", this specifies the algorithm as a linear SVM, (x, x'), which is the simplest way of determining relationships in the data. The regularization parameter C is set to 2 in order to prevent the model vector growing arbitrarily for negligible reductions in the error, essentially minimizing the total cost of the optimization function. For testing purposes, a sample string is defined before fitting the models. This string could be anything classified as either a non-political subject such as, "I just went fishing", or the opposite, "Don't vote for this candidate". By using the predict method to pass this string into each classification model, a prediction of either 'POLIT' or 'NOT' is returned. Using this system, sample tweets collected from a Twitter user can be analyzed in the same way, and separated into a single data set of strictly political tweets.

3.3 Collecting and Applying User Data

After competing the first module, I now had a way to predict the political subjectivity of any string that is passed into the model. As I mentioned above, the second module consists of determining a Twitter user to investigate, collecting their most recent tweets, and applying the labeling prediction to each tweet. All integration with Twitter was done using the tweepy library, which allowed for easy data extraction with minimal lines of code. In order to integrate with the Twitter API and tweepy, a Twitter development account had to be created. Once approved, I needed to register the application to get the four necessary client keys, which I stored in a simple JSON file named "client.json". Since these keys are sensitive, this step must be replicated for anyone who wishes to use the tool. By using tweepy's OAuthHandler() and set_access_token() methods, as well as simple JSON parsing, authentication to the Twitter API can be made fairly quickly.

Once authenticated, the tool will prompt the user to specify a specific Twitter user's name or handle to investigate. Every Twitter account has three main data types associated with it, a name, screen name, and user ID. The name is the first form of identity for each user, and is the most common way of searching for someone. Similarly,

the screen name is a secondary form of identity but is unique to the user, begins with the @ symbol, and is usually referred to as the user's handle. Both the name and screen name are completely customizable by the user, and can be changed at any time. However, the user ID is an unchangeable string of numbers that is assigned to each user when they create an account, and is hidden away unless you request it through the API. The three types of identity per user creates a confusing dynamic, but allows for more accurate searching considering the sizeable amount of users on the platform. The name and screen name act as a way of referencing a specific person, even if they have matching names. The user ID is also useful from a programming perspective as it will always map to a user, and integers are usually more versatile than strings.

By putting all of this information together, I figured it would be best to allow the tool's user to accept both name and screen names when searching, and iterate through the top 20 results. By using tweepy's search_users() function and the user's input as a query, the tool will return the name and handle for each result, allowing the user to select which account they are searching for. Once selected, the tool will return that user's unique ID. The goal of this method was to work similarly to the people search provided on the Twitter app.

With a specific Twitter user selected for investigation, the next step involves gathering tweets from their timeline. Listing 3.3 below demonstrates the simple query for tweets, and how each one is parsed.

```
# call twitter api to fetch tweets
fetched_tweets = self.api.user_timeline(user_id=user_id, count=count)

# parsing tweets one by one
for tweet in fetched_tweets:
    # empty dictionary to store required params of a tweet
    parsed_tweet = {}

    # saving text of tweet
    parsed_tweet['text'] = self.clean_tweet(tweet.text)
    # label set to 'NOT' by default
    parsed_tweet['label'] = 'NOT'
```

Listing 3.3: Fetching Tweets and Parsing

As you can see from the code above, the tweepy library makes this process extremely simplified. By leveraging the user_timeline() method, which takes the user's ID and the count of tweets to return as parameters, every tweet and retweet from the user's timeline can be saved into a list. Since I needed each tweet to contain not only the text, but also a label and eventually a classification, it made the most sense to utilize a Python dictionary for every tweet object. By iterating through each tweet returned, a cleaned version of the textual data is saved to the 'text' key, while the default 'NOT' is saved to the 'label' key. The clean_tweet() method is provided and described in Listing 3.4 below.

```
# Regular expression to remove punctuation, links, and RT's
```

```
return ' '.join(re.sub("(RT)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ",
    tweet).split())
```

Listing 3.4: Tweet Cleaning Expression

The process of cleaning or processing the text within a tweet is fairly straight forward, given an understanding of regular expressions. By utilizing the default re library provided with Python, I was able locate and replace unwanted symbols, punctuation, links, and the 'RT', which represents when a text is retweeted, with a single line of code. The expression within the first parameter of the sub() method is the regular expression itself, while the second parameter indicates white space as the replacement. The process of cleaning is necessary when working with tweet data due to the possibility of unexpected formatting within the text. It is also necessary when working with the term frequency vectorizer described in the previous section.

With a data set of cleaned user tweets freshly collected, the next step is to pass each tweet into the supervised model for labeling. I decided in giving the user a choice between the two models implemented to reduce the execution time, and provide more evaluative material. Based on the user's choice, each tweet object is passed one by one into the predicting and labeling method provided in Listing 3.5 below.

```
# labels tweet as political or not, returns subject for both models
# tweet returned with label, option 0 = NB, option 1 = SVM

if option == 0:
    tweet['label'] = self.clf_NB.predict(self.tfidf.transform([tweet['
    text']]))
else:
    tweet['label'] = self.clf_SVM.predict(self.tfidf.transform([tweet[
    'text']]))

return tweet
```

Listing 3.5: Predicting Subjectivity and Labeling

The code provided in Listing 3.5 demonstrates how the 'label' key of each tweet is saved based on the prediction of its subjectivity. As I mentioned in the previous section, both learning models will either return 'POLIT' for a tweet with political subjectivity, or 'NOT' if it is unable to classify it as such. By using the predict() method on the classifier trained previously, as well as transforming each tweet's textual data back into vectorized form, the predicted output is returned. Once every tweet has been passed through the prediction, and labeled accordingly, all tweets with a political subjectivity can extracted into a separate list for further classification.

3.4 Sentiment Analysis and Classification

There are a plethora of methods for classifying a text with sentiment analysis other than the methodology I chose to take. Many of related works I read while research used a variety of sentiment tools such as tokenization, lemmatization, and supervised

learning classification models. While it would be possible for my classification algorithm to implement a learning model, similar to the first module, I was unable to find a free data set that could meet my needs. After searching and downloading a bunch of various data sets, I decided on a set containing the Twitter handles of all of the current representatives in the House of Representatives, their corresponding party, and 200 sample tweets from each [16]. The methodology for this algorithm was to first determine if there were any mentions to active congressional politicians within the tweet, and if there were, classify the tweet based on the overall polarity. To expand, a mention in this context is when a Twitter user specifies another user within a tweet by including their Twitter handle. Additionally, I needed to clean the data set by removing the tweet column and deleting all duplicate entries. I now had a set of handles for each representative, and so I needed to add all 100 senators with their corresponding handles as well. After obtaining a list of every senator's handle, I manually entered their corresponding party into the list. By combining both the representative and senator sets together, I transformed the data set object into a list of dictionaries where each entry contained a handle and corresponding party of all current congressional politicians.

The next step was to iterate through every tweet inside the list of politically labeled tweets and pass the text through the classification algorithm. As I mentioned above, the Textblob library offers a variety of features, however I only needed a way to extract the nouns within the text and determine it's overall polarity. Listing 3.6 below shows the first part of the implemented sentiment classification.

```
# create TextBlob object of passed tweet text
blob = TextBlob(tweet)
# attempt spelling correction
blob.correct()
# float to hold polarity of tweet
polarity = blob.polarity
# list of nouns within tweet
noun_list = blob.noun_phrases
```

Listing 3.6: Generating Polarity and Nouns

The code provided in Listing 3.6 demonstrates usage of three key TextBlob methods. By first creating a TextBlob object using the tweet's text, we can perform all the necessary methods on the given text. The first at which is a novel attempt at spelling correction, which is about 70% accurate, according to the documentation. While not necessary, correctly spelled words help improve the accuracy of the sentiment analysis. A polarity score for the text is generated and saved as a float in the range [-1.0, 1.0]. This score represents an interpretation of the emotion expressed where -1 is negative, 0 is neutral, and 1 is positive. Then, a list of all the nouns from the tweet are extracted using the noun_phrases method. If a certain politician is mentioned in the tweet, their handle will be recognized and appended to this list. Both the polarity score and nouns list are passed into the classifying algorithm depicted in Listing 3.7 below.

```
# -1 = left leaning, 0 = neutral, 1 = right leaning
tweet_ratio = 0
for noun in nouns:
  for entry in self.politicians:
    if noun in entry['Handle']:
      party = entry['Party']
      if polarity > 0 and party == 'Democrat':
        tweet_ratio = -1
        return tweet_ratio
      elif polarity > 0 and party == 'Republican':
        tweet_ratio = 1
        return tweet_ratio
      elif polarity < 0 and party == 'Democrat':
        tweet_ratio = 1
        return tweet_ratio
      elif polarity < 0 and party == 'Republican':
        tweet_ratio = -1
        return tweet_ratio
    else:
      return 0
```

Listing 3.7: Classifying Based on Polarity and Mentions

As I briefly mentioned above, the goal for this algorithm was to determine a stance for the given tweet based on the text's polarity if the tweet contained a mention to a politician. As you can see from Listing 3.7, the tweet_ratio variable holds the political stance of the tweet, where -1 is left leaning, 0 is neutral, and 1 is right leaning. This variable is initialized to 0, and if no politicians handles' are found within the text, a neutral stance is returned. Regardless, every noun within the nouns list is checked against the set of politicians handles'. In the off chance there is a match, the political party of that politician is pulled from the set. Then by analyzing both the polarity value and party, a judgement can be made about the tweet's stance. For example, if a tweet has a negative polarity and mentions a Democratic politician, it is classified as right leaning and receives a score of 1. Similarly, if the tweet has a positive polarity and mentions a Democratic politician, it is classified as left leaning.

The result of this classification method is then returned to the set of political tweet objects we collected from the inspected user. Since the tweet is a dictionary data type, the 'classification' key is set to the returned value. By creating a two separate lists for left and right leaning tweets, the percentage of both right leaning and left leaning tweets is determined and printed out for the user. Based on these percentages, an overall stance of the Twitter user can be made. While this stance is the final output of the Polarized tool, a selection can be made to display more analytics including which tweets are classified as which. This and other visualizations are introduced and explored in Chapter 4.

Chapter 4

Experimental Results

Immediately upon complete implementation of all three modules described in the previous chapter, experiments were able to be conducted. This chapter will discuss the various test performed with the Polarized tool and report their findings. I will also be evaluating a few metrics of the tool including the accuracy of tweet labeling and sentiment classification modules. A discussion will follow each evaluation to explore the results and their validity. It is important to note that these experiments are replicable through public use of the tool, however since they rely on time-oriented tweets, your results may be different from mine.

4.1 Experiments

The goal of the Polarized tool is two fold; estimate the political orientation of a given Twitter user, and perform analysis as accurately as possible. With these goals in mind, experiments conducted with the tool needed to be performed on Twitter accounts with an already observable bias. Due to the open ended nature of the tool, any Twitter account could be used to test the tool, however it is very unlikely that a random user being investigated will explicitly reveal their true political standing. In order to reference the estimated results of the tool with the actual political standing of a user, we would have to know that user's standing ahead of time. When proposed with this problem, I decided on two methods of testing. The first would focus on a select number of active politician's twitter accounts. By using the tweets of a politician, their standing is already represented by the party they support, to a certain degree. While this method is great for proving the accuracy of easily identifiable classification tasks, I wanted the tool to be accurate for less predicable use as well. This led me into the second method of testing, which was performed using the personal Twitter accounts of my peers.

As the first method of testing began, I decided it would be most beneficial to chose politicians who are not only adamant about their political views, but to test from both sides of the spectrum. In an effort to create interesting results, I thought I would chose two politicians who are actively in the spotlight as well. For the first test, I choose Donald Trump, the current President of the United States, as the right leaning subject. Not only is Donald Trump active on Twitter, but he often references other politicians

via Twitter handle, which is the backbone of how the sentiment classification operates. For the left side, I chose Bernie Sanders, a Democratic senator from Vermont, and a running candidate for the upcoming 2020 Presidential election. Similar to Donald Trump, Bernie Sanders is also very active on Twitter but is far more left leaning in policies he preaches and represents. Please note that I remained as neutral and unbiased as possible throughout experimentation. While I did test the tool on other politicians, I figured these two would be the most recognizable, and embody a prime example from each side of the spectrum.

Sample terminal output from the first method of experimentation on President Donald Trump's Twitter account is provided in Figure 4.1 below. From this figure you can see the steps the tool goes thorough as well as what it query's from the user. Although its not shown in this example, the user search method will return user names and handles for each search result. Additionally a choice between which learning model is also given to the user.

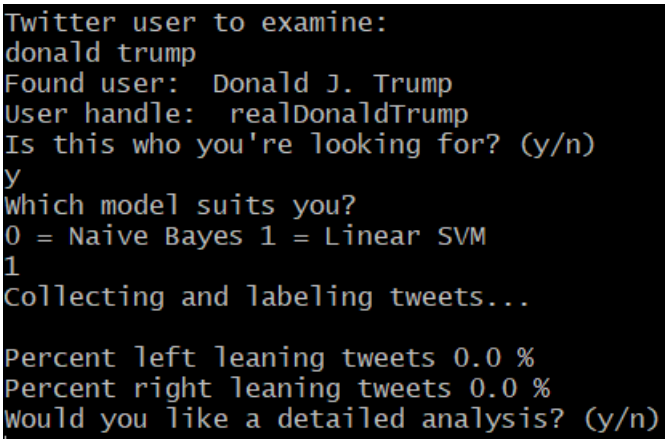

Figure 4.1: Donald Trump Estimated Standing

As you can see from the result of the first test, the tool was able to correctly search and return the most recommended user for the given query. For this test and the following ones, I chose to use the Linear SVM for tweet labeling, since it has been proven to be more accurate. After collecting and labeling each of Donald Trump's most recent tweets, the final result of the tool didn't manage to classify any of the tweets as right or left leaning. Upon further analysis, some tweets did managed to be labeled as political, but each of them was classified as having an undefined leaning. While this isn't the exciting result I was hoping for, I will evaluate possible reasons for why this happened later in Section 4.2.

Despite the lackluster results, it was time to evaluate the left leaning subject, Bernie Sanders. Sample terminal output from experimentation on Bernie Sander's Twitter account is provided in Figure 4.2 below. As I mentioned in the previous example, a linear SVM was the model of choice for this test as well.

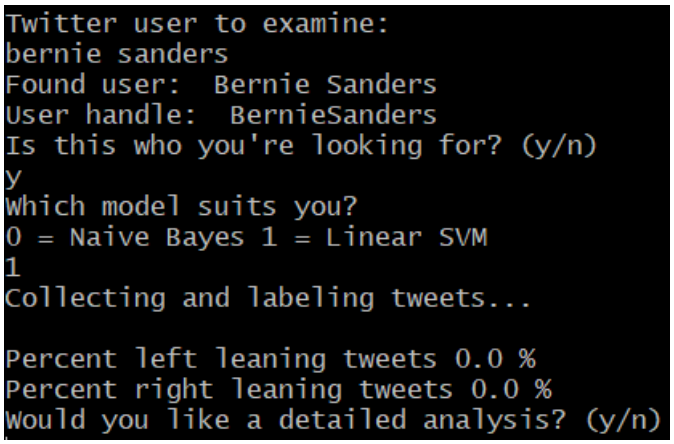

Figure 4.2: Bernie Sanders Estimated Standing

As you can see from the result of this test, the tool was able to find the correct Twitter account, and collect all of Bernie Sander's most recent tweets. Unfortunately, we had the same result from the previous test. While the tool is successful in labeling tweets as political, the classification algorithm didn't manage to classify any right or left leaning tweets. Instead, every political tweet was given an undefined leaning.

At this point, it seems there is a problem with the classification algorithm's implementation. Just to be sure, I figured I would continue with the second method of testing. For this test I entered in the Twitter account of one of my peers, who's name and account will remain concealed. This subject was chose due to their political participation on Twitter, and their consent to being tested on. Unfortunately, the tool returned the same result as previous tests. There were a handful of tweets labeled as having political subjectivity on their timeline, however it was unable to classify any of them as left or right leaning. While the results of these series of tests were an unfortunate end to a large amount of work, there is still much to learn from what went wrong.

4.2 Evaluation

As a system with multiple connected parts, it is important to evaluate each part individually. This means evaluating the accuracy of the training and labeling algorithm, and the sentiment classification. As I mentioned in the previous chapter, the sample tweets provided by the Twitter political corpus are split into two separate sets. Of these two sets, a majority of entries are put into the training set which is used to train the learning algorithm. The remaining entries are put into the test set which is used for evaluating the accuracy of the learning algorithm. By using the trained model, and a set of test entries, it is possible to determine what percentage of classes are estimated correctly, thus providing an idea of how well the learning model performs. In the context of the Polarized tool, all sample tweet inputs in the test set are either pre-labeled as 'POLIT' or 'NOT'. So by sending each tweet within the test set through the trained model, we get an estimated label to cross reference with the actual label. Following this process, the accuracy is then generated by the ratio of correct predictions

to the total number of input samples.

While it is possible to simply find the accuracy score of true positives predicted in the test set, accuracy alone doesn't give enough information to truly judge the model's performance. Thankfully, the scikit-learn library offers a variety of other evaluation metrics. To return the most metrics out of a single test set as possible, I decided to utilize the library's classification report visualizer. This report can be generated for each instance of our training algorithm, and displays the precision, recall, F1, and support scores for the model. In both Figure 4.3 and 4.4, you can see the terminal output for both models implemented; multinomial Naive Bayes, and linear SVM.

Classification report for chose model:

	precision	recall	f1-score	support
NOT	0.94	0.90	0.92	472
POLIT	0.86	0.92	0.89	329
accuracy			0.91	801
macro avg	0.90	0.91	0.90	801
weighted avg	0.91	0.91	0.91	801

Figure 4.3: Classification Report - Naive Bayes

Classification report for chose model:

	precision	recall	f1-score	support
NOT	0.90	0.88	0.89	460
POLIT	0.84	0.87	0.86	341
accuracy			0.88	801
macro avg	0.87	0.88	0.88	801
weighted avg	0.88	0.88	0.88	801

Figure 4.4: Classification Report - Linear SVM

Each classification report shows a representation of the main classification metrics on a per-class basis. This gives a deeper intuition of the classifier behavior over global accuracy which can mask functional weaknesses. The metrics are defined in terms of true and false positives, and true and false negatives. Positive and negative in this case is just a generic name for the class of label assigned, i.e political or not. A true positive in this context is when the estimated label is equal to the actual label. On the contrary, a false positive is when the the estimated label is negative but the actual label is positive and vice versa. Using this terminology, we can define each of the metrics shown in the classification reports above.

The precision metric is the ability of a classifier not to label an instance positive that is actually negative, which is the ratio of true positives to the sum of true and false positives. This is essentially the correct percent of classification made by each model. As you can see from the first example, the multinomial Naive Bayes model has a precision score of 94% for classifying non-political tweets, and a score of 86%

for political tweets. The linear SVM however, had a precision score of 90% for non-political and 84% for political tweet classification. Both of these models reported high precision, but precision isn't everything.

The recall metric is similar to precision but has a slightly different. It is the ability of a classifier to find all positive instances. In other word, it means for all instances that were actually positive, what percent was classified correctly. According to the report, the multinomial Naive Bayes had a recall score of 90% and 92% for non-political and political tweets respectively. This means the model had a slightly harder time classifying non-political tweets over political ones. On the contrary, the linear SVM reported a recall score of 88% and 87% for non-political and political tweets respectively. While both of these model's recall numbers were still fairly accurate, we can further evaluate by combining precision and recall together.

The F1 score is a weighted harmonic mean of precision and recall. Generally, F1 scores are lower than accuracy scores as the embed precision and recall into their computation. It is the weighted average of the F1 score that is typically used to compare models over global accuracy. For the multinomial Naive Bayes, we have a recorded F1 score of 92% and 89% for non-political and political tweets respectively with a weighted average of 91%. For the linear SVM, an F1 score of 89% and 86% for non-political and political tweets respectively gives us a weighted average of 88%. Since the average for the multinomial Naive Bayes is 3 percentage points higher than that of the linear SVM, it is initially perceived that the Naive Bayes preforms better classification, but there is a deeper reason behind this outcome.

When examining the support metric, which is the number of actual occurrences of the class within the test set, we notice an imbalance between the two labels. In both instances, there are over 100 more non-political samples than there are political. Imbalanced support in the training data usually indicates structural weakness in the reported scores of each classifier. As we know from the previous chapter, the combined Twitter political corpus data set has exactly 2,290 non-political tweets and 1,714 political tweets. This imbalance would indicate the need for stratified sampling or re-balancing, thus diminishing the minuscule difference between our reported F1 scores of each model.

While this outcome prevents us from supporting which model is more accurate than the other, it shouldn't be looked at as a failure. An evaluation of both models proves that either is a fine choice for the classification task at hand. Both reported high accuracy, and it is clear from testing that a supervised learning model can separate non-political tweets from political ones. The evaluation of the sentiment classification algorithm is unfortunately not in the same state.

When it comes to evaluating the sentiment classification for political leaning, it becomes apparent that the system implemented is broken. It was my initial expectation to utilize a confusion matrix of left leaning to right leaning tweets for each subject, however since I was unable to gather any of those I will instead share samples of tweets that are clearly polarized in a certain direction, but were classified as neutral or undefined by the current system. Hopefully, by supplying these examples it becomes clear how inaccurate the current algorithm is. The first of which is a tweet made by Donald Trump and is as follows.

> My Administration is helping U S auto workers by replacing the failed Obama Emissions Rule Impossible to satisfy

For some context, this tweet had already been pre-processed and cleaned of punctuation among other things, but it is still clearly polarized. While being correctly tagged as political, most likely due to keywords such as administration and Obama, it is improperly classified as neutral. There are a few key features of this statement that I believe make this more right leaning. The main reasons being the adjective failed describing an Obama era emissions rule. It is clear that the subject of the message is to instill a negative sentiment towards a democratic policy, and is thus right leaning. The next tweet comes from Bernie Sanders and is as follows.

> This is truly outrageous Congress needs to be working urgently to protect the health of the American people

This tweet is an interesting example because without context, many people would agree that it contains a neutral stance. I decided to include it to exemplify the limitation of sentiment analysis within a classification algorithm. While the subject of the tweet is about protecting the health of American people, its context is in lieu of a republican backed plan to stimulate corporations from economic collapse due to the COVID-19 pandemic. While it is easy for a human to read the tweet and pick up on its negative sentiment of right leaning ideas, a sentiment analysis system isn't smart enough to understand the complex nature of our political landscape. I will expand on this issue and others in the next section. For now, the final tweet comes from my peer and is the following.

> Please refuse service to any trump supporters

I am truly unsure of what context this tweet was written in response to, but it is clearly left leaning. The tool did correctly label it as political, mostly due to the keyword trump, but incorrectly classified it as a neutral standing. There is a very polarized and negative sentiment towards Donald Trump supporters, and is thus left leaning. Hopefully after seeing these examples, it becomes overwhelmingly obvious that there are deep rooted problems within the sentiment classification implementation within the Polarized tool. Some of these problems are related to specific technological limitations while others are more abstract.

4.3 Threats to Validity

While the Polarized tool was successful in some aspects of its task, but it should come at no surprise that there were numerous limitations and issues that threaten the study's validity. Among these threats, there are problems with data, tweet collection, sentiment analysis, and overall structural problems with the implementation. Not only that, but the task I chose to solve is rather difficult to comprehend through technology alone. I intend to discuss each of these topics in this section, and the reasons why they diminish the success of the tool. Additionally, I will attempt to conclude why the outcome of the study ended the way it did.

Starting from the first module of the implementation, which consisted of training the supervised model, problems began to arise with data collection. As I mentioned in the previous chapter, I needed a rather large set of tweets that were labeled as having a political subject or not to train the machine learning model. I figured it would take too long to collect tweets and hand annotate them myself, so I was left with searching the internet. As it turns out, it was quite difficult to find a free data set that met my needs. Ideally, I would use a set containing example tweets and a label for their political leaning, as to train the model to do the entire classification task. Unfortunately the only relevant data I could find was the Twitter Political Corpus [10]. While I'm grateful I was able to find such a data set, it definitely had its limits when it applied to my implementation.

The main issue of the Twitter Political Corpus set was the time frame from which the data was collected. Each tweet was randomly selected from Twitter's "spritzer" feed between June 1st, 2009 and December 31st, 2009. While I'm sure these tweets provided a good representation of politics on Twitter for half of 2009, that was over 10 years ago. The political landscape is ever evolving with each election cycle, and so the topics, policies, and politicians referred to in these tweets were outdated. Fortunately, some of the language and buzzwords used back then were still relevant, so it didn't have much of a significant impact on performance when processed by a supervised learning algorithm. However, I'm sure with relevant data collected from present day discussions I would have had better success in this task.

Along with problems with data collection, I had similar complications with the actual machine learning system. To start, I had mentioned some of the parameter tweaking for both models in the previous chapter. While I was developing this module, I had begun to notice the inaccuracy of some of the results I was getting. I would test my trained model by letting it predict the political subjectivity of sample strings I came up with myself. However, throughout the various tests I would get results for strings that obviously had the opposite subjectivity, to which I would attempt to fix the problem by tweaking parameters. In the end, I made the system as accurate as I could, but I came to the simple conclusion that supervised learning methods are just imperfect. Whether it was problems regarding training data, or the internal functionality of an algorithm, there never seemed to be a way to make a perfect machine learning prediction.

Moving on to the second module of implementation, which consisted of gathering and applying data from a user, there were a handful of technological limitations that hindered the success of the tool. The first of which was the rate limit of pulling tweets allowed by the free tier Twitter development account. I could only request the last 200 tweets from a user's timeline in one instance. Not only did this limit the time frame of study for each user, but it included replies to tweets, and retweets of other user's tweets. Some users tweet more frequently than others, and so the most recent 200 tweets on someones timeline could only date back a couple of days in certain scenarios. The addition of replies and retweets within the data set added another layer of complexity as well. It is nearly impossible to infer the user's sentiment of a reply or retweet without also analyzing the original tweet or the ones before it in a thread, and so I just treated them as if it was the user's own ideas.

The problems surrounding context continued since a significant amount of tweets include images or other types of media. Whatever media the user is attaching to

a tweet can represent a lot about what opinion they're trying to convey, however it would be an entire new challenge to analyze said media with the technology we have today. Furthermore, there were multiple issues with the way tweets were pre-processed. Most of the time, it seemed important parts of the tweet would be removed or a full 280 character tweet would be cut short. I believe this was mostly due to a combination of technological limitations of regular expression replacement, and what sentiment analysis libraries like TextBlob could handle. If I could have spent more time fixing the bugs of the tool, this is where I would start.

Continuing on, I experienced a myriad of problems with the sentiment classification module. The quality of the algorithm I produced definitely had theoretic flaws from the beginning. Not only was this a difficult classification task to solve, but my implementation had an extremely limited scope of effectiveness. At first, an algorithm that uses the sentiment expressed towards politicians to infer political stance doesn't seem two flawed. However it occurred to me later on that political orientation is a completely subjective matter. While it could true that tweeting negatively about a politician from the opposite party you represent proves your stance, conclusions can't be drawn from that alone. There are so many factors that affect a person's political stance, such as party, history or policy, that isn't capture by the implemented algorithm.

As far as technical limitations go, there is one critical issue with the current state of the tool that is responsible for the results it produced. Like I said in the previous chapter, I relied on TextBlob's noun extraction feature to deduce the subject of each tweet in the hope that it would be referring a politician's Twitter handle. However it became apparent after doing some tests that this feature of TextBlob was unable to recognize a handle as a proper noun. I tried this with multiple sentences, and each time it would either return a correct noun that wasn't a handle, or no noun at all. I believe it has to do with either the library of known words TextBlob is built upon, or some aspect of pre-processing where the handle is removed from the tweet. Unfortunately I didn't recognize this fundamental flaw until it was too late, and so I was stuck with the lackluster results I shared above.

Overall, there were plenty of theoretical and technological issues that threatened the validity of this study. While the Polarized tool was successful in labeling a tweet as having political subjectivity or not, the predictions were never perfect. I was limited by the quality of data provided for training the algorithm as well as multiple technology limits for gathering new tweets, and analyzing them. There were also the looming threat of algorithmic flaws within my stance classification procedure. Nonetheless, it doesn't mean that is project is completely useless, as my failures can act as a stepping stone for a more robust implementation in the future.

Chapter 5

Conclusions and Future Work

I have described my approach for developing a system to estimate the political stance of a Twitter user by use of supervised learning and sentiment analysis tasks. Moreover I have defined a metric to take into account the polarity of tweets containing political subjectivity and reference to active politicians. I was able to train a classifier based on a sample set of pre-labeled tweets and then systematically collect and append tweets from any given public Twitter user though the trained model. By individually cross referencing any mentions of active politicians with the overall sentiment for each tweet, a estimation of political orientation could be deduced.

The evaluation of this system was performed thorough multiple tests of Twitter accounts from politicians and my peers alike. To avoid personal bias, it was necessary to chose politicians who are not only active on the platform, but are also self declared as left or right leaning. In addition to these test, the accuracy of the two learning models were evaluated using a classification report. This allowed for an empirical comparison between each model's F1 scores. Additionally, a judgement could be made about the effectiveness of the sample training data through this report.

5.1 Summary of Results

The tool developed through this study was both successful and unsuccessful. By utilizing a supervised learning model, I have trained and built a system able to determine the political subjectivity of tweets pulled from an account in real time. Furthermore, both models supplied in this system are fairly accurate with an F1 score of 91% and 88%, given the data, technological, and project limitations. The final results of the entire system however wasn't significant considering the inability of the sentiment classification algorithm to determine which tweets were polarized.

While the overall results didn't express much innovation to a field that is ever evolving, I still consider it a significant step in solving the task at hand. As social media continues to expand, the volume of data available has become enormous along with its potential benefit to businesses and research. The ability to understand an audiences' opinions or beliefs based solely on textual expression has become a sought after technology for both advertising and political gain. It is not unlikely for a significant piece of technological innovation to originate from a failed experiment, and I fully believe

that with more work, the Polarized tool could become something great.

5.2 Future Work

I am clearly very interested in machine learning and sentiment analysis tasks among trends in social media. I can prove not only its significance to the evolution of social media as a whole, but to the way we analyze the data it produces as well. In the case of this specific project, I have several ideas on how to improve both the implementation and overall approach to identifying political stance in Twitter.

By referencing the numerous threats I discussed in the previous chapter, there are plenty of avenues for improvement given more time. I would first want to look into more powerful sentiment analysis libraries in order to get a better noun extraction procedure. I would also have to rethink the algorithm for determining polarity on a larger scale. Perhaps a similar approach to recognizing politicians in tweets could be implemented for recognizing policies since they are usually polarized topics as well. There has been significant research done in this area some of which take into account a user's connected network as well. This includes analyzing who a user follows, and interacts with on the platform in addition to the tweets they produce. I believe the ideal system would be able to combine the many factors of this task with a deep learning approach to create the most accurate prediction possible.

The difficulty of building an annotated corpus of tweets that can be used to evaluate and train alternative systems should be emphasized. There needs to be a great effort put forth to acquire and manually label tweets. The validity of this labeling would also need to be evaluated to correct any errors as well. In this context, I was pleased to find the corpus of sample data that I did, as it was a significant help to the time frame of this project.

On the portability of system, I think it would be beneficial to adapt the idea I created to other platforms and context. The mechanics of this tool would need to be functioning at high accuracy, and would need to include a full fledged test suite to ensure the functionality across a variety of circumstances. The end goal being an accurate, user friendly tool for both commercial or political research as well as for personal analysis of ideological discussion on social media.

Finally, this project has come very far from its conception. While it might not answer many significant questions, the development of this specific tool will be beneficial to its scientific community. Along with the code, I have also grown significantly along this journey. This area of research has always fascinated me, and I'm glad I have been given the chance to explore it in such an educational and intellectual process.

Bibliography

[1] Basile, P., Basile, V., Nissim, M., Novielli, N., Patti, V., et al. Sentiment analysis of microblogging data., 2018.

[2] Basile, V., Bolioli, A., Nissim, M., Patti, V., and Rosso, P. Overview of the evalita 2014 sentiment polarity classification task.

[3] Brun, C., Perez, J., and Roux, C. Xrce: Feedbacked ensemble modeling on syntactico-semantic knowledge for aspect based sentiment analysis.

[4] Do, H. H., Prasad, P., Maag, A., and Alsadoon, A. Deep learning for aspect-based sentiment analysis: a comparative review. *Expert Systems with Applications 118* (2019), 272–299.

[5] G., N. Revealing blogging statistics: The state of the industry in 2019, Mar 2019.

[6] Hasan, A., Moin, S., Karim, A., and Shamshirband, S. Machine learning-based sentiment analysis for twitter accounts. *Mathematical and Computational Applications 23*, 1 (2018).

[7] Hong, S., and Kim, S. H. Political polarization on twitter: Implications for the use of social media in digital governments. *Government Information Quarterly 33*, 4 (2016), 777 – 782.

[8] Liu, D., and Lei, L. The appeal to political sentiment: An analysis of donald trump's and hillary clinton's speech themes and discourse strategies in the 2016 us presidential election. *Discourse, Context & Media 25* (2018), 143 – 152.

[9] Loria, S. Textblob: Simplified text processing, 2020.

[10] Marchetti-Bowick, M., and Chambers, N. Learning for microblogs with distant supervision: Political forecasting with twitter. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (2012), Association for Computational Linguistics, pp. 603–612.

[11] Martherus, J. L., Martinez, A. G., Piff, P. K., and Theodoridis, A. G. Party animals? extreme partisan polarization and dehumanization. *Political Behavior* (2019), 1–24.

[12] Martin, J. R., and White, P. R. *The language of evaluation*, vol. 2. Springer, 2003.

[13] Mejova, Y. Sentiment analysis: An overview. *University of Iowa, Computer Science Department* (2009).

[14] Mohammad, S. M., Sobhani, P., and Kiritchenko, S. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT) 17*, 3 (2017), 26.

[15] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., and Stoyanov, V. SemEval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (San Diego, California, June 2016), Association for Computational Linguistics, pp. 1–18.

[16] Pastor, K. Democrat vs. republican tweets, May 2018.

[17] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[18] Pla, F., and Hurtado, L.-F. Political tendency identification in twitter using sentiment analysis techniques. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical Papers* (2014), pp. 183–192.

[19] Preoţiuc-Pietro, D., Liu, Y., Hopkins, D., and Ungar, L. Beyond binary labels: political ideology prediction of twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2017), pp. 729–740.

[20] Rajadesingan, A., and Liu, H. Identifying users with opposing opinions in twitter debates. In *International conference on social computing, behavioral-cultural modeling, and prediction* (2014), Springer, pp. 153–160.

[21] Roesslein, J. Tweepy, 2020.

[22] Russel, M. A. *Mining the social web: data mining Facebook, Twitter, Linkedin, Google , Github, and more.* OReilly, 2014.

[23] Taboada, M. Sentiment analysis: an overview from linguistics. *Annual Review of Linguistics 2* (2016), 325–347.

[24] Xu, S., and Zhou, A. Hashtag homophily in twitter network: Examining a controversial cause-related marketing campaign. *Computers in Human Behavior 102* (2020), 87 – 96.

[25] Yilmaz, K. E., and Abul, O. Inferring political alignments of twitter users: A case study on 2017 turkish constitutional referendum. *arXiv preprint arXiv:1809.05699* (2018).