“The Use of Aspect-Based Sentiment Analysis on Political Classification & Potential Impacts”

Hunter Berry

Berry College – HON 451

Abstract: With natural language processing (NLP) and machine learning technologies improving at a rapid pace, aspect-based sentiment analysis (ABSA) has become a powerful tool for data analysts who wish to categorize data. Using social media posts from Twitter, this paper seeks to study the possible uses of ABSA to identify a user's political polarity. After analyzing the results, the paper offers a discussion on the possibility of misuse with this algorithm and similar technologies, alongside other possible implementations.

Keywords: sentiment analysis, machine learning, ai ethics, ideological identification, politics, aspect-based sentiment analysis.

Contents

1 - Introduction

Technology in the 21st century is a rapidly growing and ever-changing entity. Every day, scientists are finding new ways to push the boundaries of technologies and provide some kind of benefit to the world around us. Social media is one major aspect of this technology and something that is constantly growing and changing to give users around the world access to information sharing capabilities we once thought impossible. Although social media has been prevalent in many forms since the original days of “MySpace,” one of the most common forms of social media today is Twitter. Twitter is a major web and mobile application that allows its user to tweet out small, 280-character messages to their various followers and interact with each other’s messages through hashtags, replies/threads, retweets, likes and “favorites,” and much more.

With Twitter’s popularity steadily rising, many researchers have turned to it to use the platform for studies aimed at determining how to detect thoughts and opinions based on a tweet’s text alone. This paper seeks to build on these studies, exploring the use of tweets as predictive components that can be used to determine a given user’s political polarity. Using a number of factors, I have been able to utilize a plethora of existing software libraries and predictive models to build a framework capable of determining political polarity within a select subgroup with upwards of 75% accuracy. However, with this kind of technology comes a variety of concerns. While there are numerous benefits that can come from these technologies, like reducing the billions of dollars an average election season in the U.S. costs or detecting radical profiles and tweets, there are also a number of negatives, with the technology opening up the possibility for invasions of privacy, discrimination and racism, and even radicalization and radical recruitment. Thus, while we may find some benefits, there are clearly many dangers as well that must be dealt with and discussed if technology like this is to continue to grow. With these factors in mind, this paper seeks to show how the results identified could pose significant danger to society and offer some solutions to this problem aimed at providing additional security to Twitter users through new default settings and changes to the Twitter API.

Ultimately, then, the purpose of this paper is to explore the use of aspect-based sentiment analysis (from here on out referred to simply as ABSA) on Twitter data, the potential uses of this technology, and methods to prevent possible misuse. Thus, the breakdown of this paper is as follows: Section 1 will contain a brief literature review, focused on detailing the rapid growth of ABSA and similar technologies, specifically emphasizing their immense predictive capabilities. Section 2 will offer some discussion and definitions of the models and techniques used in this study, and Section 3 will elaborate more on these by offering a brief literature review. Section 4 details the approach and methodology used in a more theatrical and framework-based sense, whereas Section 5 discusses the actual implementation of these ideas. After, we move into Section 6, which discusses the results of the algorithms and approaches tested. Sections 7 and 8 respectively discuss the project’s limitations and room for future work and improvements. In Section 9, I provide a detailed discourse on the potential positive and negative uses of this technology and similar ones, discussing topics like discrimination, political radicalization, and media targeting, before then using these findings to advocate for increased security and privacy. Finally, Section 10 presents a brief conclusion of the paper and some final remarks.

2 - Definitions and Models

The purpose of this section is to offer a brief overview and description of the various models used within this paper, including Naïve Bayes and the probabilistic modeling approach, sentiment analysis (and the more detailed ABSA layer), vectorizers, and the machine learning models used within our hybrid approach, specifically the XGBoost Classifier and the Sci-Kit Learn Logistic Regressor.

2.1 - Naïve Bayes and Probabilistic Model

Naïve Bayes is a probabilistic model, where a classification is produced based on the probability of some number of events occurring. The model takes in two necessary features – a “feature,” which is an aspect of something that may or may not be present, and a set of labels in which the model is attempting to classify. Essentially, this boils down to the equation

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

where A is the label, and B is the feature in question. While this approach is great for single features, it must be taken a step further to incorporate models where there may be many features at once. To do so, I adopt the formula provided by TextBlob, a Python-based natural language processing toolkit. To incorporate multiple features, we also operate under the assumption that each feature is independent of each other (i.e., that the inclusion or exclusion of feature does not affect the probability of another feature occurring). With this in mind, TextBlob boils the equation down to

$$P(A|B^N) = \frac{P(A) * P(B^1|A) * \ldots * P(B^N|A)}{\sum(P(A) * P(B^1|A) * \ldots * P(B^N|A))}$$

which represents a normalized (sum of one) representation of the probability of each label being present given the respective features (Loper 2019). In layman terms, the equation simply represents the probability of a certain classification being given when provided a set of features. For a coded representation of the model, see the TextBlob library, and for a description of the model applied to this project specifically, see Section 4.1 – Naïve Bayes.

2.2 - Sentiment Analysis

Sentiment analysis, as described by data scientist Bing Liu, is “[the] field of study that aims to extract opinions and sentiments from natural language text using computational methods” (2020, 1-2). Essentially, in terms of natural language processing, sentiment analysis uses the terms and phrases in each sentence or set of words to determine the “sentiment” or opinion of the given set. Sentiment is determined by analyzing the set for phrases and terms that may imply a given sentiment. For instance, terms like “ecstatic” or “amazing” may imply a positive or “happy” sentiment, whereas terms like “terrible” or “miserable” may imply a negative or “sad” sentiment. Terms which cannot be related to a specific sentiment are often labeled as neutral and left out of calculations.

Although sentiment can be calculated in a plethora of different ways, the most common and straightforward approach is to simply take the difference between the number of positive and negative terms in a phrase and produce a score, where a score greater than 0 represents an overall “positive” sentiment, a score less than 0 represents an overall “negative” sentiment, and a score of 0 represents a neutral statement. For instance, the sentence “I love dogs” may produce a positive sentiment, the sentence “I hate cats” may produce a negative sentence, and the sentence “I love dogs but hate cats” may produce an overall neutral sentiment. A more in-depth analysis of specific positive and negative terms, as well as other calculation methods, can be viewed by analyzing something like the R library, which contains several statistical models and equations.

While sentiment analysis provides a good way at getting the overall view of a phrase, determine certain aspects of the phrase may be even more helpful. This is where aspect-based sentiment analysis (ABSA) comes into play. ABSA provides a more insightful view on a phrase because instead of calculating the overall sentiment of a phrase, it calculates the sentiment for single terms of phrases (Liu 2020, 90-91). For instance, if given the previous sentence "I love dogs but hate cats," ABSA would produce results showing that the sentiment towards dogs is positive and the sentiment towards cats are negative. Because of this detailed information, more insightful decisions and classifications can be made based on ABSA results when compared to ordinary SA results.

2.3 – Vectorizers and Bags of Words

Before it is possible to discuss the hybrid models and their various machine learning models, it is important to first discuss the data that goes into them. To get our data, we go through a process called "vectorization," which, as defined by the Sci-Kit Learn library as "[the] general process of turning a collection of text documents into numerical feature vectors" through tokenization, counting, and normalizing, which in this case refers to the process of removing words like "a" or "the" that occur too frequently and cannot help provide any valuable insights (Pedregosa et. al., 2011). This process creates what we call a "bag of words" or a "bag of n-grams," which represents a frequency chart containing all of the terms within the training data and the frequency with which they occur. This frequency chart can then be analyzed, allowing for comparisons to be made between sentences based on the terms used and the labels given to both sentences.

2.4 – XGBoost Classifier

The XGBoost Classifier is one of the two models that will be used within the hybrid approach in the project, which combines the results of vectorizer with the results of an ABSA model. The XGBoost Classifier is a type of gradient boosting algorithm, meaning that it takes an interactive approach. Unlike other machine learning techniques, which make predictions based on a single model or decision tree, gradient boosting techniques make lots of models, using the results of model one, both good and bad, to train model two, and so on until no further improvements in predictive accuracy can be made (“Introduction to Boosted Trees”). Because of this approach, many errors in previous models can be accounted for and weeded out, producing an optimized classification system. With strong results seen in other projects, which are detailed in Section 2, this algorithm will provide strong results. Further mathematical details and model implementation can be viewed in the XGBoost library (“Introduction to Boosted Trees”).

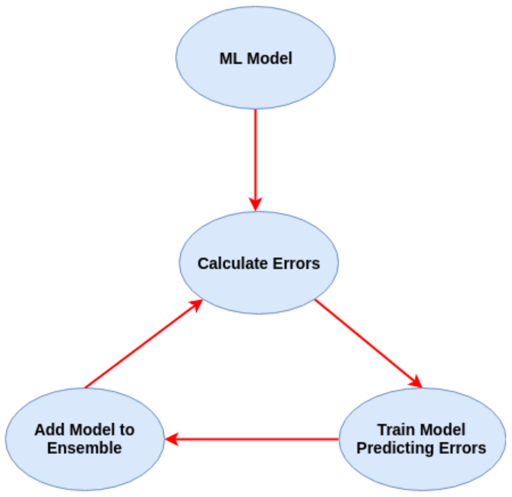

Figure 1: Graphical representation of the gradient boosting process, provided by George Seif in his article “A Beginner’s Guide to XGBoost” on Towards Data Science.

2.5 – Logistic Regressor

The logistic regressor is another type of probabilistic model, similar to the Naïve Bayes classifier. Much like its counterpart, the regression system works by taking in two labels and predicting the probability of a given label occurring given a number of features. Similar to a linear regression system, values and features are combined with weights and coefficients to predict the probability of a given class occurring, and these probabilities are in turn converted into binary classifications (the predictions) based on the sigmoid function modeled in Figure 2. Put simply, “[d]ata is fit into linear regression model, which then be acted upon by a logistic function predicting the target categorical dependent variable” (Swaminathan 2019). For a more in-depth mathematical analysis and description of learning methods implemented in logistic regressors, see Pedregosa et. al.'s Sci-Kit Learn implementation.

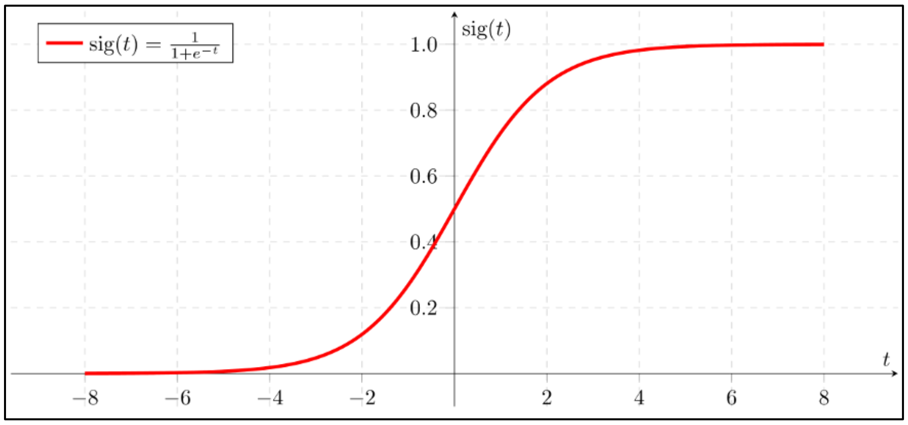

Figure 2: Representation of the sigmoid function that logistic regression works on. Provided by Saishruthi Swaminathan in their article "Logistic Regression – Detailed Overview" on TowardsDataScience.

3 - Related Works & Literature Review

Although a discussion has been provided of the techniques and models used within this project, a defense of them must also be present to show why these techniques were implemented instead of other models. Below follows a review of some data science literature that show strong performance for the techniques we have chosen to implement.

In terms of Naïve Bayes, a recent study applied probabilistic classification methods to a study of political classification to the 2016 United States presidential election. The study had the goal of using tweets from short-term events (town halls, debates, rallies, etc.) to determine an overall sentiment level, and in turn using overall sentiment levels to predict political party. The researchers in this scenario use a plethora of differently tuned Naïve Bayes classification systems, all of which returned with above 75% accuracy levels for their predictions (Ding et. al. 2017). It is important to note that these researchers had databases with much higher levels of data than other studies, like Bakliwal et. al's, and the others that will follow, with approximately 28,000 tweets available for training and testing purposes.

While Naïve Bayes shows some promise, then, we most also investigate sentiment analysis and some of its possibilities, especially given the fact that numerous other successful studies have heavily relied upon it. Recent research has shown that classification from sentiment is possible, and that sentiment analysis is an excellent way to derive not only general opinions, but political opinions as well. Indeed, in a recent study, scientists analyzed tweets surrounding the 2011 election in Ireland. Using both supervised machine learning techniques and lexicon-based techniques, the latter of which was implemented as a "bag of words" set, the researchers were able to correctly identify just over 60% of all tweets in their dataset of ~3,500 as containing either positive or negative sentiments towards a specific political party and/or its various members and leaders (Bakliwal et. al. 2013). While this accuracy is admittedly somewhat low, this study is one of the few (that I could find) that used sentiment as it relates to politics, instead of using general sentiment as a predictive indicator. Seeing this is something that my study plans to do as well, this is an important study for my research.

Although sentiment analysis has proved to be successful, it is also important to address the abilities of aspect-based sentiment analysis instead. Success has been seen in a number of studies, with a specific Twitter based study of particular interest. Zainuddin et. al. conducted their ABSA study using a variety of different extraction methods, but what was clear from their study was that ABSA produced extremely good classification results and accuracies. Each of their various implementations produced results with above 70% accuracies, higher than the accuracies computed when using simple sentiment analysis methods (Zainuddin et. al. 2018). Thus, the benefits of ABSA are clear, especially when applied to Twitter-based data.

Finally, it is also important to discuss some of the models used within this study and their predictive capabilities. Both our Logistic Regressor and XGBoost Classifier have proved to be extremely accurate in a number of different problems, including text classification. Indeed, XGBoost is the number one used model by scientists competing in Kaggle data competitions, and is simply generally known for having excellent performance, speed, and trainability (Dwivedi 2020, Morde 2019). Thus, because of the ease at which variables can be adjusted in training, and the strong results in other text classification problems, XGBoost should be a good algorithm for us to use in our approach. Similarly, the logistic regression has also proved to be extremely strong. It one study, logistic regression beat every other tested model/algorithm by at least more than 10% and has performed strong in other projects as well (Pranckevičius & Marcinkevičius, 2017). Since it is recognized as one of the best and "most useful" machine learning algorithms to date, we believe it will be a good algorithm to match up against the XGBoost during the testing process ("Logistic Regression For Machine Learning and Classification" 2019).

Overall, then, there is great potential for aspect-based sentiment analysis to build on these results and create new ways to determine political polarity from tweets. Taking combinations of the results from these studies, among others, like Hasan et. al's study on sentiment analysis with the recent Indonesian elections, we can determine a suitable approach and methodology that can incorporate both aspects of a lexicon-based approach, and those of machine learning.

4 - Approach & Methodology

The project began with an analysis of possible approaches. Ultimately, there are many ways we could have engaged in the classification system, but the two we decided to specifically study was a Naïve Bayes (NB) approach, and a hybrid ABSA-machine learning approach. Respective breakdowns will follow.

4.1 – Naïve Bayes

Naïve Bayes was chosen as the primary testing algorithm because it is a relatively simply algorithm to implement, and one that generally preforms well with limited data sources. Specially, Naïve Bayes has produced optimal results in previous studies that studied sentiment on Twitter (Hasan et. al. 2018, Ding et. al. 2017). Furthermore, it was chosen for its potential to incorporate states into a relatively simply system.

As a probability-based model, our Naïve Bayes model is calculating based on a few key areas. A more in-depth analysis of this can be viewed 2.1 - Naïve Bayes and Probabilistic Model. Building on the equations discussed in said section, we can apply the knowledge to our own problem and come up with the equation:

$$P(Party|Feature) = \frac{P(Feature|Party) * P(Party)}{P(Feature)}$$

It is important to stress this is the simplified version of the formula, not the version incorporating *n* features, although the transference of variables would nevertheless look similar. So, with this in mind, I can simplify our equation to words and say that the probability of a given term being associated with a given party (party given term) is equal to the probability of the term given the party multiplied by the likelihood that the term is going to occur, all divided among the probability of the party being chosen. That being said, for our Naïve Bayes approach, the data needed is simply the tweet and the party of the user in question.

4.2 – Hybrid Approaches

A hybrid approach was chosen for the second application because of its multi-faceted capabilities. The decision to implement this hybrid model was based on promising results brought about in numerous research studies, showing that hybrid models, while subject to somewhat noisy results, are great at “detect[ing] and measur[ing] the sentiment at the concept level" and are "less sensitive to changes in topic demand" (Dandrea et. al, 2015). Other studies cited by Liu in his discussion on supervised learning and machine learning techniques within the ABSA field also showed positive results.

Overall, the data needed by a hybrid model to perform well is much more complex in comparison to the Naïve Bayes approach. For my hybrid model, I first begin by preforming ABSA on a set of tweets, which is then pipelined into the main classification system. Before classification is done, all tweets in the database are sent through a vectorizer, which will break apart the database into a bag of words. These two sets of features will then be combined into a common dataset and fed into my two classification systems (the XGBoost Classifier and Logistic Regressor) and their results/predictions are produced accordingly.

In terms of the aspects that were chosen, a full list is viewable in *src/sentimentCacher.py* and in Appendix A. In general, I attempted to choose topics that were a) ongoing/popular discussion topics, b) controversial (commonly argued about), and c) clearly differentiable (i.e., a position is clearly linked to a certain party). In doing so, I generated eight core topics: economics, police, foreign policy & immigration, the president (at the time of this writing, Donald J. Trump, although aspects also included the election in general), military, abortion, and health/healthcare. Aspects were then generated for each topic. Aspects, either phrases or words, were derived from a number of sources, including analyzing popular topics on Twitter, Instagram, and Facebook through hashtags, an analysis of Google search queries, and simply reading news articles and position comparison websites on decisive issues. For example, the "health" topic consists of aspects like the coronavirus (or COVID-19), healthcare, PPE, and more, while the "police" topic consists of aspects relating to abuses of power, fatal shootings, racism, protecting and patrolling, and more. When things like PPE or COVID-19 also have other forms of mention (i.e., PPE => personal protective equipment), both terms/phrases are included to attempt to catch as many mentions as possible.

5 - Implementation

Ultimately, the project is developed in Python 3.8.1, with reliance on a number of libraries, all of which can be viewed in Table 1 and the project's *requirements.txt* file.

Library Name	Version	Purpose/Uses
Numpy	1.19.4	Use of custom arrays and dense models for data storage and model input.
Pandas	1.1.14	Use of library's DataFrame model for data storage, exporting said data to CSV files, and as model input.
Natural Language Processing Toolkit (NLTK)	3.5.0	Used alongside TextBlob for stemming, tokenization, noun phrase extraction, and other language-oriented transformations.
TextBlob	0.16.0	Used for Naïve Bayes Classifier implementation, as previous studies have found their implementation to produce the best accuracies (Hasan et. al., 2018).
Python Aspect-Based-Sentiment-Analysis	2.0.1	Provides implementation for aspect-based sentiment analysis through a number of different models and techniques.
XGBoost	1.2.1	Provides an implementation of the XGBoost model and classification system.
SciKit Learn	0.23.2	Provides an implementation of the Logistic Regressor model and classification system.
Tweepy	3.9.0	Used for interacting with Twitter developer account and API to pull tweets and account information.

Table 1: List of Python libraries used within research.

5.1 – General Pipeline

Overall, the project is organized into a pipeline such that users wishing to engage with the code do not have to engage in much editing or alternating from process to process, instead simply passing along the needed data from one end of the pipeline to the other. This pipeline and its various components are represented in Figure 3. To initialize the project, users can run the *src/twitterAPI.py* file, which interacts with the local text file containing a list of current Congress members and White House staff, alongside their associated political party. Republicans and conservatives are dictated with an R, whereas Democrats and liberals are dictated with a D. Members of third parties, alongside self-proclaimed independents, are associated with the party they most tend to vote with. Since web scraping for this information proved to be a timely and inefficient process, the text file was created manually and must be updated manually should users wish to engage in a more recent (i.e., post-2020) analysis of data.

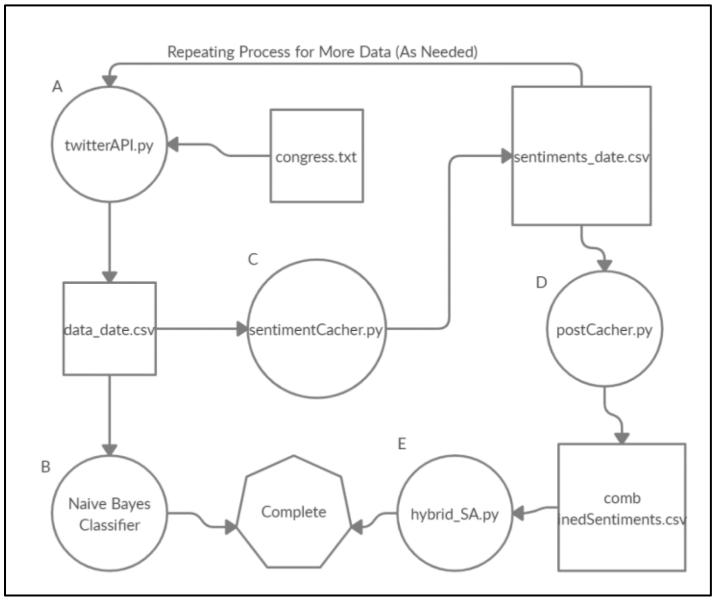

Figure 3: Program life-cycle flowchart with implementation. Squares represent data inputs and outputs, circles represent pipeline components and files, and hexagons represent finished states.

The *twitterAPI.py* file will pull the latest tweets from each of the users located in the text file. Although this method is tunable, a default number of five tweets will be pulled per user. While an increased number of tweets may help generate more training and testing data, it will mean that users will experience a halt in processing as rate-limits (limits on how much information can be pulled from an API in a given time frame) are reached. Although there is a work around to this problem (see Appendix C), it will regardless take additional time as a new process must be opened and a new cursor (API interaction tool) must be created. However, when not rate-limited, users have the option to engage in a number of additional methods on pulled tweets, such as processing general/overall sentiment, doing pre-processing, tokenizing, noun phrase extraction, and more. After the tweets are extracted, they are then sent to CSV files for use in other parts of the pipeline (see output and choices from Figure 3 'A').

As previously mentioned, the Naïve Bayes approach is somewhat fast and easy to implement. To begin with, data files from the *twitterAPI.py* file are converted into DataFrame objects from the Pandas library. Data is split into two sets, categorized as training and testing data, before using both sets to compile a Naïve Bayes classifier, which is imported from the NLTK and Textblob libraries. Predictions and probabilities are based on the frequency and probability charts created by Textblob during the training phase. The classifier then takes in a list of tuples (containing tweet and party, respectively) and makes its predictions before determining its accuracy (see output and choices from Figure 3 'B').

Things are somewhat more complex for our hybrid approach. After downloading the tweets and preprocessing the data, tweets must now be parsed during the ABSA process. A Python library is used to extract the aspects, which are all listed in the *sentimentCacher.py* file. Aspects are broken down into categories for visual purposes, but categorization does not play a role in the implementation nor the analysis of the data itself. Ultimately, the ABSA process takes a considerable amount of time to complete, taking up the most time within the process. On average, a normal-length tweet takes between forty-five seconds to a minute and a half to analyze. Since our databases comprised a number of tweets from March to October, it was important to divide up the process to speed the analysis along. Data was split into four sections, each section being run on a Windows 10 computer with 8 gigabytes of Random-Access Memory (RAM) and an Intel i5 processor. After each section of tweets is done being analyzed, the sentiments, along with information from the original process (specifically tweet and party) are loaded into a new CSV file (see output and choices from Figure 3 'C->A'). These files are then sent into a separate pipeline component (*postCacher.py*) where the files are concatenated for training and testing purposes and all rows with errors are weeded out (see output and choices from Figure 3 'C->D'). See the following section for a more in-depth explanation of the project's caching process.

Finally, the data is loaded into *hybrid_SA.py*, where the actual predictions are done. Tweets are passed through a bag-of-words vectorizer before this data is concatenated with the sentiment data into one large DataFrame. This data is then sent over to the respective classification systems (either the XGBoost Classifier or the Logistic Regressor) for training. After the training phase is completed, the testing dataset is similarly vectorized and then concatenated with the sentiment data, before being tested on the classifier and printing out the results (see output and choices from Figure 3 'E'). Code snippets of the training and testing can be viewed in Appendix D and Appendix E respectively.

5.2 – Caching Implementation

The caching process for the pipeline is, as previously mentioned, utilized to reduce the time it takes to complete the data processing of the pipeline. Users can cache any CSV file formatted in the same method as the data located in the project's folder (further detailed in the *twitterAPI.py* file). In terms of structure and operations, the caching batch will take in any spreadsheet given and complete sentiment analysis on it. The process itself is fairly simple, as the main goal/responsibility of the batch file is to simply ensure that sentiment analysis process is restarted in the case of any errors, which can occur due to issues with tweet parsing, the ABSA library running out of memory, and more. The exported CSV file is viewable at any point during the caching process, so users can see how much data has been processed at the given moment. Once the process is complete, users can use the file as needed, or use the postCache.py file to combine multiple cached documents into a single CSV for processing.

6 – Results

Overall, initial results for both the Naïve Bayes approach and the hybrid approach proved to be better than expected. This section offers a brief breakdown of the results generated and some insights gained by analyzing them.

6.1 – Naïve Bayes

The Naïve Bayes classification proved to be the most accurate of our approaches, although this classification system was trained and tested on smaller databases due to data chunking issues within the algorithm. However, with this in mind, when given ~2,500 rows of data and working on a 50% train, 50% test split, the algorithm predicted the correct party of around 80% of tweets. This implementation, as previously mentioned, only worked with word-based features, although future iterations may be re-worked to include states and location as well (see Section 8 – Future Work).

6.2 – Hybrid Approaches

As far as our hybrid-approach goes, similar results can be found. Tests were run on approximately 1,700 lines of testing data after drops were done to account for rows with missing data points and sentiments, and approximately 6,000 lines of training data when drops are once again accounted for. Tweets in this specific dataset were from the beginning of October, with the default number (5) of tweets being pulled for each respective user in the dataset of users to pull from (see Section 5 - Implementation). With a learning rate of .9, our XGBoost algorithm correctly classified 1,237 rows out of 1,677, equating to a 73.77% accuracy level. Our Logistic Regressor produced slightly better results, classifying 1,278 out of 1,677 right, equating to a 76.21% accuracy level.

With this in mind, there are some interesting insights that can be seen from the results. In terms of Republican predictions and results, both our logistic regressor and boosting regressor produced similar results. In fact, our boosting regressor only produced one more correct response for Republicans than our logistic regressor did (565 vs. 564). However, our logistic regressor did produce noticeable increases in correct predictions for Democrats. Our logistic regressor produced roughly forty more correct results for Democrats than Republicans, leading to a somewhat large increase in correct democratic predictions (~76% accuracy vs. ~73% accuracy). Further research into the various features analyzed is needed to determine why this is the case, but this nevertheless gives some insight into why our logistic regressor is proving to be more accurate. Furthermore, at this time, tuning on both of these classifiers has not yet been done.

Future results will include more fine tuning to these classifiers, including adjustments to max depth, the learning rate, random states, and the number of features and estimators.

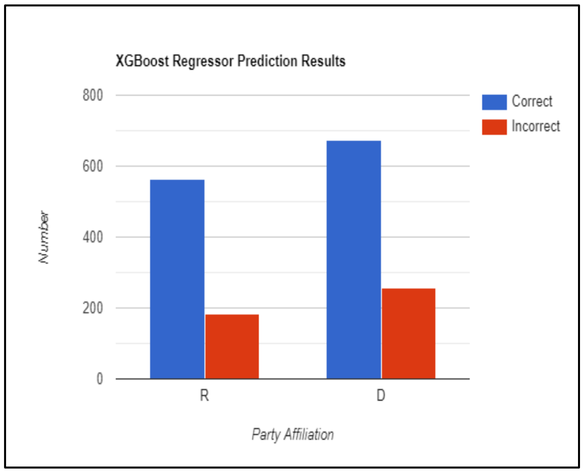

Figure 4: XGBoost Classifier results broken down by party.

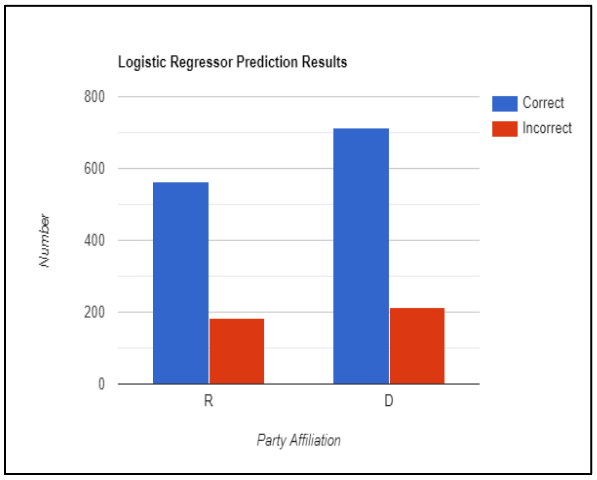

Figure 5: Logistic Regressor results broken down by party.

6.3 – Duplicating Results and Personalized Testing

In an effort to improve the validity and replicability of the results, an interface has been provided in the code to allow all users the opportunity to interact with the data. The interface is accessible in the *src* folder of the project and includes step by step instructions through a terminal window on how to interact with the data. To start the interface, simply download the projects files, install the required libraries and resources by running *pip install* in the main directory, and then running *python src/interface.py*. Users can start their own pipeline to download and analyze fresh data, or they can interact with saved models based on the aforementioned results and training data to test single tweets or pull information from a user's account.

```
Starting Program…
Would you like to 1) Start the pipeline, or 2) run tweet analysis on training
data (1/2)? 2
Would you like to 1) run analysis on a single tweet/text, or 2) pull from a live
feed (1/2)? 1
Please enter the text of the tweet. President Trump must be removed from office
as a result of today’s attacks on the capital, and this is an example of his bad
leadership.
Some weights of the model checkpoint at absa/classifier-rest-0.1 were not used
when initializing BertABSClassifier: ['dropout_4711', 'extractor']
- This IS expected if you are initializing BertABSClassifier from the checkpoint
of a model trained on another task or with another architecture (e.g.
initializing a BertForSequenceClassification model from a BertForPretraining
model).
- This IS NOT expected if you are initializing BertABSClassifier from the
checkpoint of a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).
Some weights of BertABSClassifier were not initialized from the model checkpoint
at absa/classifier-rest-0.1 and are newly initialized: ['dropout_37']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
['The XGBoost model predicts that the tweet is D.']
['The Logistic Regression model predicts that the tweet is D.']
```

Figure 6: An example of using the interface to interact with the project's data. In this example, we use of training data to run the (paraphrased) version of a Tweet taken from Speaker Nancy Pelosi after the 1-7-21 protests in the U.S. capital. Bold represents user input, whereas italics represents output from the interface.

6.4 – General Notes

It is important to note that these predictions are all on Congressmen and other political figures included in our search database. Therefore, it is possible, and indeed likely, that these predictive capabilities may diminish upon the introduction of non-political content. However, while the algorithm may have difficulty classifying single tweets as Republican or Democrat, its overall predictive capabilities may remain somewhat high in terms of account classification. As long as some of the user’s tweets contain even remote political or social beliefs, it becomes possible to break down these tweets and run them through the classification system. Furthermore, it is possible that with increased training on more “common” Twitter users, or various non-political Twitter users, that my bag-of-words and vectorizing models may be able to more accurately predict non-political users and their various speech and sentiment patterns. This, alongside other issues, is discussed more in-depth in the following section.

7 - Limitations

Ultimately, this project was limited in study due to both my own personal capabilities, and time constraints as well. As a result, it is important to stress the need for future work on this topic and the field in general, as well as to address some limitations. The most important limitation to address within the project is the use of "political ideology." Political ideology is a broad idea, and something that can encompass a wide variety of subtopics, as was evident by the previous discussion on aspects and categories (see Section 4.2 – Hybrid Approaches). As such, it is important to note a few issues with my approach.

First of all, a key limitation is the lack of a clear definition for not only political ideology, but what encompasses it. In terms of political ideology itself, some people isolate one's "ideology" to their clear political ideals, for instance their position on something like abortion. However, to others, ideology may also consist of other beliefs, such as their position on human behavior and its ability to change (often referred to as "liquid") or be static ("plastic") when surrounded by different environments. This debate, and others like it, can provide information on one's ideology even though it is not explicitly political, but the matter of its inclusion in the definition of ideology is what raises one issue.

On a similar vein is the issue of what beliefs coincide with what party. In American politics, there are dozens of topics and ideas that are "associated" with each party. For instance, "socialism" and "capitalism" are often related with Democrats and Republicans, respectively. However, this sole belief, while important in helping to classify someone's location on the political spectrum, is not the only attribute that comes into play. Thus, although I did my best to incorporate a wide variety of aspects into my analysis, it is undeniable that there are aspects of each "ideal" or party that may have not been included.

In addition to this issue, one must also recognize that many ideals are not exclusive to one side – there are many conservatives in favor of gun laws and restrictions, a belief typically associated with a liberal way of thinking, just as there are many religious liberals who are against abortion, a belief typically associated with conservatives. In a similar vein, there are many Republicans who dislike the Republican president Donald Trump and many Democrats who dislike the current President-Elect Joe Biden. As is evident by these examples, it may be difficult to actually "classify" someone's political ideology, because while a party's beliefs may be associated with a certain set of beliefs, the average American may have a set of beliefs with ideas from both sides. This may even be further hampered by the radicalization that's going on in the U.S.'s political climate. With politicians becoming more and more radical, both major political parties in the United States have been becoming more and more marginalized from their supporters, with two clear "groups" emerging from both – a more moderate side, comprised of those with beliefs that are either not extremely strong/radical or beliefs that may overlap with the opposing party, and a second, more radicalized group that is similar to many of the outspoken politicians. As previously mentioned, our training and testing data was based on the tweets of current U.S. politicians. So, while there are some moderate and reasonable politicians in Congress, as well as some self-proclaimed "independents," the tweets of the more radical politicians may only be useful/accurate when identifying supporters who share these stronger beliefs.

With all of that said, the key limitation I am attempting to point out is that it may be difficult to simply take a person's Twitter account and predict their "political ideology," or whatever that may be. Although we have already mentioned a number of issues, there exist many more, like the potential for someone to alter/hide their feelings on social media to appeal to a certain audience base, or a general lack of data for someone who may not constantly be talking about politics or a wide berth of ideas. With this in mind, the results of this project, while promising, are limited in the scope of their possibilities. The section that follows looks at some ideas for future work to improve the predictive capabilities and reliability of the pipeline.

8 – Future Work

In terms of future work, the first area of research that could impact this study is the use of "hashtags." Hashtags are (typically) short words preceded by a # symbol that work to give a conversation some sort of "theme." They work to "define" the conversation, and also link a single tweet to any number of other tweets with the same hashtag (Fishman). This connectivity provides clear potential for those interested in sentiment analysis. Because hashtags allow us to find other tweets with a similar theme, it becomes possible to analyze tweets that may connect to our original point of study. In doing so, we can collect a plethora of additional datapoints which may be able to provide more conclusive evidence as to a person and/or groups' thoughts on the given hashtag and "topic." It is important to stress that a given analysis could potentially skew the results, as all Twitter uses have the potential to use a given hashtag, meaning any "connected" tweets could be positive, negative, or neutral. However, a more refined search (i.e., an aspect-based analysis) should provide an apt analysis and give solid results.

Another area of interest would be looking into the impact of language on political party. Although not mentioned in the results sections, one of the limitations of the code and the required libraries was an inability to preform ABSA on languages other than English. Ultimately, this means that the data, for purposes of analysis, had to be removed from the training and testing sets. However, I hypothesize that the use of foreign languages could greatly impact political polarity classification. Ultimately, we can assume that most politicians using a foreign language in a public post do so because of a large abundance of non-English speaking citizens within their jurisdictions. Similarly, studies have also shown that a majority of Hispanic citizen, regardless of their level of English comprehension and speaking, tend to be Democratic in nature (Lopez et. al. 2016). Combined together, we can make the assumption that most Americans speaking only Spanish on Twitter or living in areas with politicians who routinely use Spanish on their social media accounts, are more likely to be Democratic than other counterparts. Using this assumption, we can hypothesize that this language could be used to predict polarity to some degree and may be useful in such a classification algorithm. Further research into both the connections between language and polarity, as well as language and its impact on sentiment analysis, are needed to confirm these suspicions.

Finally, another area of further research is the increased predictive capabilities of other algorithms. Although this study did its best to incorporate a variety of machine learning methods and models, there are numerous other algorithms that were not studied in this research project. For instance, Liu proposes the idea of using Hidden Markov Model (HMM) chains or Conditional Random Fields (CRFs) to increase accuracy in real-world problems with a wide range of possible states and values (Lui 2020). This is in addition to the numerous other techniques employed in other studies, such as Support Vector Machines (SVM), which all may lead to additional accuracies.

9 - Implications & Applications

While there is much work to be done on the classification systems that have been created, it is evident from both this classification system and the numerous ones mentioned within the literature review is that there is a great possibility that things like tweets and Facebook posts can be used to determine personal aspects of someone's life, such as political polarity. This introduces a number of concerns, and this section is dedicated to discussing these concerns as they relate to my own project and similar studies, as well as possible benefits that may arise from the technology.

However, before a discussion on these benefits and negatives may be discussed, it is important to note the difficulty in providing an in-depth analysis of this topic and why a simple pro-con analysis style is used. While many ethicists and scientists alike have begun to delve deeper into "cyber-ethics" in recent years, there still remains not only a lack of literature on the topic as a whole, but also a lack of a set of methodologies one can follow while undergoing such an analysis. Hence, this section simply seeks to discuss the pros and cons of this technology and discuss what should be done with this in mind.

9.1 – The Bad

This paper was in part inspired by a 2017 report from Stanford University on the use of facial recognition for the prediction of a person's sexuality and sexual preferences. Nicknamed the "Gaydar" study, the controversial study exposed several issues within the computer science field. Although the results were claimed as fraudulent by many social justice groups, the algorithm was able to correctly identify someone's sexuality with between 75% and 91% accuracy, dependent on factors including gender, number of scans given, and more (Murphy 2017, Wang & Kosinski 2020). As mentioned, the study brought to light a number of issues, including the ease at which the data (in this instance, facial scans and photos) was able to be pulled from websites, and the ease at which the machine was able to classify extremely personal information, leading to concerns at how this algorithm, and the others out in the world like it, can intrude on personal privacy and data privacy.

Thereupon, the first issue that I believe must be addressed in terms of “the bad” is discriminatory practices. Unfortunately, discrimination is nothing new in the field of technology. Discrimination is a common occurrence among algorithms, both intentionally and on accident. For instance, a 2017 ProPublica report detailed how insurance algorithms were unintentionally charging higher prices for African Americans, and Ruha Benjamin has an entire book dedicated to discussing other instances of race and its overlap with technology (Kirchner & Angwin 2017). Coeckelbergh also mentions a number of racial issues within his book, noting how a Florida algorithm used to determine likelihood of criminals recidivating was biased against African Americans (Coeckelbergh 2020). Thus, it is evident that race and other factors can play a key role in how these programs and many others make decisions.

With this in mind, while race and political party are not definitive predictors of each other, as was previously mentioned, more minorities tend to be members of the Democratic party. As a result, algorithms that specifically target a certain political party, regardless of their reasons, may be biased in favor of or against certain races and ethnicities. This discrimination is important, because, as Coeckelbergh mentions, it could influence almost every factor of someone’s life, including “whether or not individuals get a job get credit, end up in jail, or even experience violence against them” (Coeckelbergh 2020). While it is true someone could easily lose a job over their political beliefs, the latter of the list is especially important in today’s political climate, as the United States is currently facing more racial tension and political tension than it ever has, with both factors unfortunately being involved in the motivation of a number of attacks within the country in recent months and years. Thus, from these examples, it should be clear that the discrimination that can occur from being able to find one’s political party is extremely possible and unfortunately likely, leading to a number of social, economic, and political consequences.

Finally, a third potential negative consequence of a technology like mine is its potential to radicalize. In the following section, I discuss potential targeting methods and how they could be used to save election money and use it for more economically efficent purposes. However, the same targeting could also be used by politicians for radicalization purposes. At the time of writing this paper, the nation is in the aftermath of the 2020 presidential election and the turmoil that unfolded alongside it. This turmoil, alongside the events of the summer of 2020, mainly the Black Lives Matter (BLM) protests in cities like Seattle, caused great damage to the nation, and some of the most destructive of the events were led and organized by radicals, often with the support of their sides’ leading politicians. Should this sort of technology be public, it could allow politicians to more easily locate and target the radicalized support bases and engage with them in what could be destructive behavior. So, while there may be benefits of this targeting, there are downsides as well.

9.2 – The Good

While there are fewer “benefits” that could arise from this type of technology, I do believe there are some possible positives, the first surprisingly dealing with a form of discrimination. Ultimately, we know that in America, elections are costly. Indeed, many estimates on the cost of the 2020 presidential election are upwards of 14 billion dollars (Cillizza 2020). While we all know that election spending is always high, one of the reasons election spending is so high is because politicians are trying to advertise to a broad audience and secure as many independent or swing voters as possible, since a) they likely have the support of members of their own party, and b) it will likely be difficult to gain the support of the opposing party.

Unfortunately, studies have shown that independent voters may be the most difficult and costly to identify and target. For instance, one report shows that independents may cost upwards of $250 per person to identify/locate, more than $75 dollars the cost of targeting a Democrat and $200 the cost of targeting a Republican (Mandese 2020). Although the study in hand specifically focused on the use of television to target potential voters, similar results may be prevalent in other forms of media and targeting methods. While this form of advertising, or “microtargeting,” has its own benefits and flaws/threats, the use of our algorithm to target solely independent, or extremely moderate members of the opposing party, could work to reduce costs associated with American political elections by reducing the costs associated with identifying and successfully targeting a specific group (Borgesius et. al. 2018).

Another possible “benefit” of the software could be in its potential utilization in determining fake social media accounts, or “bots,” and misleading information. The idea of Twitter bots influencing U.S. politics has been around for a while, with accusations and beliefs spiking throughout and in the aftermath of the 2016 presidential election. The so-called “Russian twitter bots” were accused of manipulating the election via the spreading of misinformation, or “fake news,” and far-right propaganda (“Russian interference in the 2016 US presidential election,” 2017, O'Connor & Schneider, 2017). Although the methods here would need more tuning, these methods could be applied by companies like Twitter to help identify far-right, or even far-left Antifa accounts of both real people and bots and remove them from the platform before damage could be done.

Similarly, it could be used to determine what terms, phrases, and sentiments are typically evident in the propaganda and fake news and use this data to implement more careful monitoring systems in the future. While the system would almost undeniably have some false positives, the benefits of removing fake news and radical speech from platforms may outweigh the costs of removing a few members who aren't engaged in these behaviors. And on the other side of the spectrum, these programs could allow us to test concerns over whether or not Twitter and other social media platforms are *actually* discriminating against certain political groups. By analyzing the accounts of the users being banned and "censored," we could determine if this was based on the actual language and facts involved, or if the members are being targeted solely because of their political party (although it is important to note there could be a connection between party and the dissemination of "fake news").

9.3 – What Should Happen?

So, with this discussion in mind, it is evident that the potentials of such technology are both good and bad. If this is true, then this poses an important question – how do we protect the people while also ensuring that we do not stifle innovation? My proposal builds on some of the ideas that Coeckelbergh discusses, mainly the ideas of increased data privacy and protection. While this project was only made possible by having access to the Twitter API, the ability to download all the data I did for this project is concerning. I will preface this argument by stating that this data would be impossible, or extremely difficult, to download in bulk and operate on without access to the Twitter API. While some may argue web scraping (an automated process to manually download data from a URL) may work, the process would be extremely slow and would most likely be recognized by Twitter's site, leaving the API as the only reasonable option. That being said, I do believe there are numerous issues with the API.

For one, the interactions with the API are not really limited. In my interactions with the API, I had no problems downloading any data from a user, including their location, their tweets, and other information included in the API. In fact, the only issue I did have while working with the API was simple rate-limiting. While this in and of itself is not concerning, as the Twitter API is private (meaning users must apply to receive access to it), what is concerning is that Twitter is doing little to monitor how those with access use the data. While I was required to state my need/purpose for using the API when I applied, Twitter has not asked me for updated information or proof of usage since I got developer access almost a year ago now. In reality, I could be doing something entirely different then the academic project I told Twitter I was working on. This does present some issues, as those with access to the Twitter API may be using it for nefarious or at least gray-area activities. So, with these in mind, what follows are my proposed solutions.

In an effort to create greater privacy on Twitter and social media networks in general, I propose setting all new accounts to private by default. Private accounts are not accessible by APIs and cannot be viewed by those engaging in web scraping or similar methods of data retrieval. Because the information on a private account is not viewable to the public (i.e., people with normal accounts) or developers looking to gain information, it makes the information much more secure, while not harming the ability of the user to engage in the platform and everything that it has to offer. Additionally, content creators, notable users like politicians or celebrities, and the average user in general would be able to turn their profile back to public, removing the restrictions if they want. Thus, this opt-out policy will help protect more data than the current opt-out policies do.

While the previous solution is one that can fix a lot of issues, especially those targeting a majority of the general public, another solution for preventing unnecessary developer access would be to simply create more restrictions on API access. As previously mentioned, my API access was fairly easy to receive, and I've had no restrictions on it to-date. To strengthen security, Twitter, and others offering similar APIs, could require developers to provide proof of usage throughout their development process. For example, I may have been required to show Twitter my project's GitHub repository so Twitter can ensure I'm using the API in the way I said I was, and not in any potentially bad actions. While this would undeniably put more burden on Twitter, it could lead to safer applications being developed in the future and could also lead to a reduction in the number of applications overall as API keys are determined to be being used for other purposes.

The good thing about both solutions is that they can easily mitigate some of the previous concerns about negative consequences, while preserving the ability to reap the benefits of this technology. Hypothetically, if Twitter were to develop a similar technology for in-house use, while adhering to the additional privacy and API changes I have mentioned, they would be in the position to benefit. By having the technology closed source, Twitter can not only utilize the algorithm to reduce search costs associated with promoting politicized advertised tweets to its members, but also to determine potential radical and dangerous accounts without the need for reporting. While safeguards would need to be put in place to ensure these accounts are not censored or banned without human review, this closed source idea would allow Twitter to take advantage of these benefits, all the while not allowing third-party actors to exploit the API and develop a program for malicious reasons, be it recruitment, discrimination, or something entirely different.

Therefore, in the end, it is evident that there are a number of ethical and legal concerns surrounding these technologies. As we continue to innovate, these issues will only continue to grow in prevalence until they come to the forefront of society. It is important that we begin to discuss these issues now, so that we can get ahead of these problems before they grow out of hand. In doing so, we can learn to maximize our benefits and minimize our costs from new technologies as they inevitably do emerge. While I have offered up some solutions for increased data security and privacy measures on Twitter and similar platforms, there exist many other solutions as well. Regardless, actions must be made to ensure we can benefit from technologies like Twitter, while minimizing the potential misuse of it as well.

10 – Conclusion

By designing my own pipeline composed on various machine learning algorithms and sentiment analysis tools, I’ve been able to demonstrate how a rudimentary pipeline is capable of interacting with data accessible through the Twitter API to determine political polarity in addition other personal information. With a pipeline capable of predicting a select group of users’ political polarity at an above 70% success rate, I’ve been able to show how sentiment and textual analysis are good indicators of political polarity and methods that, with future research, have the potential to only grow in their measures of predictive capabilities. However, as is also evident from this paper, technology is a rapidly growing field, and one that promises to continue to disrupt society in both good and bad ways in the centuries to come. While the benefits of this technology are far-reaching and can be strengthened through future research in the field, it is nevertheless evident that there are many downsides and potentially harmful consequences that can result from usage of this type of technology. By analyzing both these positive and negative impacts, I’ve determined some possible solutions focused on increased privacy settings and restrictions that may allow us to experience the benefits of this technology while also mitigating some, if not many, of the downsides. So, while it is clear that my algorithm is still in the testing phase, and much more work is needed to be done to improve the capabilities and accuracy of the algorithm, its potential and possible uses are undeniable. Hopefully, this paper will provide a jumping off point for future research and demonstrates the need to continuously address concerns of data privacy and similar areas as future research brings up more unprecedented discoveries and situations.

References

Bakliwal, A., Foster, J., Van der Puil, J., O'Brien, R., Tounsi, L., & Hughes, M. (2013, July 13). Sentiment analysis of political tweets: Towards an accurate classifier. Retrieved October 28, 2020, from http://doras.dcu.ie/19962/1/foster2013.pdf

Benjamin, R. (2019). *Race after technology: Abolitionist tools for the New Jim Code*. Cambridge: UK.

Brownlee, J. (2020, August 14). Logistic Regression for Machine Learning. Retrieved November 16, 2020, from https://machinelearningmastery.com/logistic-regression-for-machine-learning/

Cillizza, C. (2020, October 29). Analysis: The absolutely stunning price tag of the 2020 election. Retrieved November 16, 2020, from https://www.cnn.com/2020/10/29/politics/2020-election-cost-money-trump-biden/index.html

Coeckelbergh, M. (2020). *AI ethics*. Cambridge, MA: The MIT Press.

Dandrea, Alessia, et al. “Approaches, Tools and Applications for Sentiment Analysis Implementation.” International Journal of Computer Applications, vol. 125, no. 3, 2015, pp. 26–33., doi:10.5120/ijca2015905866.

Ding, T., Deng, J., Li, J., & Lin, Y. (2017, July). *Sentiment Analysis and Political Party Classification in 2016 U.S. President Debates in Twitter*. Retrieved October 28, 2020, from http://d-scholarship.pitt.edu/34716/1/SentimentAnalysis.pdf

Dwivedi, R. (2020, June 3). Introduction to XGBoost Algorithm for Classification and Regression. Retrieved November 16, 2020, from https://www.analyticssteps.com/blogs/introduction-xgboost-algorithm-classification-and-regression

Fishman, E. (n.d.). How to create and use hashtags. Retrieved October 19, 2020, from https://business.twitter.com/en/blog/how-to-create-and-use-hashtags.html

Frederik, J. Z. B., Judith Möller, Sanne, K., Ronan Ó Fathaigh, Kristina, I., Tom, D., et. al. (2018). Online political microtargeting: promises and threats for democracy. *Utrecht Law Review*, *14*(1), 82–96. https://doi.org/10.18352/ulr.420.

Gandhi, R. (2018, May 17). Naive Bayes Classifier. Retrieved November 16, 2020, from https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

Hagendorff, T. (2020). The Ethics of AI Ethics -- An Evaluation of Guidelines. *Minds & Machines*. doi:10.1007/s11023-020-09517-8

Hasan, Ali, et al. "Machine Learning-Based Sentiment Analysis for Twitter Accounts." Mathematical and Computational Applications, vol. 23, no. 1, 2018, p. 11., doi:10.3390/mca23010011.

Introduction to Boosted Trees. (n.d.). Retrieved November 16, 2020, from https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Kirchner, L., & Angwin, J. (2017, April 05). Minority Neighborhoods Pay Higher Car Insurance Premiums Than White Areas With the Same Risk. Retrieved October 30, 2020, from https://www.propublica.org/article/minority-neighborhoods-higher-car-insurance-premiums-white-areas-same-risk

Liu, B. (2020). *Sentiment analysis mining opinions, sentiments, and emotions*. Cambridge: Cambridge University Press.

Logistic Regression For Machine Learning and Classification. (2019, August 27). Retrieved November 16, 2020, from https://kambria.io/blog/logistic-regression-for-machine-learning/

Loper, E. (2019). Source code for nltk.classify.naivebayes. Retrieved November 16, 2020, from https://textblob.readthedocs.io/en/dev/_modules/nltk/classify/naivebayes.html

Lopez, M., Gonzalez-Barrera, A., Krogstad, J., & López, G. (2016, October 11). Latinos and the American political parties. Retrieved October 19, 2020, from https://www.pewresearch.org/hispanic/2016/10/11/latinos-and-the-political-parties/

Mandese, J. (2020, September 29). Report Reveals Independent Voters Most Expensive To Target Via TV, Republicans The Cheapest. Retrieved November 16, 2020, from https://www.mediapost.com/publications/article/356326/report-reveals-independent-voters-most-expensive-t.html

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, et al. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. *10th International Workshop on Semantic Evaluation (SemEval 2016)*, Jun 2016, San Diego, United States. ⟨hal-02407165⟩

Morde, V. (2019, April 08). XGBoost Algorithm: Long May She Reign! Retrieved November 16, 2020, from https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Murphy, H. (2017, October 09). Why Stanford Researchers Tried to Create a 'Gaydar' Machine. Retrieved October 28, 2020, from https://www.nytimes.com/2017/10/09/science/stanford-sexual-orientation-study.html

O'Connor, G., & Schneider, A. (2017, April 03). How Russian Twitter Bots Pumped Out Fake News During The 2016 Election. Retrieved November 16, 2020, from https://www.npr.org/sections/alltechconsidered/2017/04/03/522503844/how-russian-twitter-bots-pumped-out-fake-news-during-the-2016-election

Pedregosa et. al. (2011). Scikit-learn: Machine Learning in Python: Logistic Regression. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Pedregosa et. al. (2011). Scikit-learn: Machine Learning in Python: Feature Extraction. Retrieved from https://scikit-learn.org/stable/modules/feature_extraction.html

Pranckevičius, T., & Marcinkevičius, V. (2017). Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. *Baltic Journal of Modern Computing*, *5*(2). doi:10.22364/bjmc.2017.5.2.05

Seif, G. (2019, September 14). A Beginner's guide to XGBoost. Retrieved November 16, 2020, from https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7

Sloan, L., & Quan-Haase, A. (2017). *The SAGE handbook of social media research methods*. London: Sage Publications.

Swaminathan, S. (2019, January 18). Logistic Regression - Detailed Overview. Retrieved November 16, 2020, from https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

Update: Russian interference in the 2016 US presidential election. (2017, September 28). Retrieved November 16, 2020, from https://blog.twitter.com/en_us/topics/company/2017/Update-Russian-Interference-in-2016--Election-Bots-and-Misinformation.html

Wang, Y., & Kosinski, M. (2020, May 25). *Deep neural networks are more accurate than humans at detecting sexual orientation from facial images*. Retrieved October 28, 2020, from https://osf.io/zn79k/. doi:10.17605

Zainuddin, N., Selamat, A. & Ibrahim, R. Hybrid sentiment classification on twitter aspect-based sentiment analysis. *Appl Intell* 48, 1218–1232 (2018). https://doi.org/10.1007/s10489-017-1098-6.

Appendix

Appendix A

```
topics = {
    "economics": [
        "consumption",
        "commerce",
        "economics",
        "economic",
        "trade",
        "gdp",
        "China",
        "investment",
        "stock market",
        "stocks",
        "stock",
        "goods",
        "financial",
        "fiscal",
        "economical",
        "profitable",
        "economy",
        "efficent",
        "finance",
        "monetary",
        "management",
        "economist",
        "macroeconomics",
        "microeconomics",
        "protectionism",
        "resources",
        "real value",
        "nominal value",
        "capital",
        "markets",
    ],
    "police": [
        "blm",
        "defund",
        "police",
        "militiarization",
        "police officer",
        "sheriff",
        "crime",
        "fatal",
        "shooting",
        "abuse of power",
        "line of duty",
        "protect",
        "protecting",
        "patrol",
        "patrolling",
        "law enforcement",
        "riot",
        "looting",
        "arrest",
        "racism",
        "law",
        "black lives matter",
    ],
    "foreign_policy": [
        "imperialism",
        "occupation",
        "un",
        "united nations",
        "united",
        "hrc",
        "who",
        "free trade",
        "anarchy",
        "nationalism",
        "foreign",
        "china",
        "russia",
        "cuba",
        "multinational",
        "regional",
        "trade",
        "international",
        "commerce",
        "alien",
        "refugee",
        "border",
        "ambassador",
        "israel",
        "pakistan",
        "terrorism",
    ],
    "immigration": [
        "alien",
        "wall",
        "emigration",
        "immigration",
        "migration",
        "illegal alien",
        "naturalization",
        "visa",
        "citizenship",
        "refugee",
        "welfare",
        "family reunification",
        "border",
        "immigrants",
        "enforcement",
        "seperation",
        "asylum",
        "sanctuary city",
        "sanctuary cities",
    ],
    "president": [
        "trump",
        "president",
        "genius",
        "smart",
        "incompatent",
        "idiot",
        "leadership",
        "cabinet",
        "election",
        "biden",
        "elect",
        "harris",
        "joe biden",
        "donald trump",
        "government",
        "corrupt",
        "russia",
        "head of state",
        "presidency",
    ],
    "military": [
        "military",
        "armed forces",
        "air force",
        "coast guard",
        "national guard",
        "army",
        "navy",
        "marines",
        "combat",
        "forces",
        "invasion",
        "occupation",
        "overseas",
        "over seas",
        "foreign policy",
        "defense",
        "intelligence",
        "military intelligence",
        "militaristic",
        "militia",
        "peacekeeping",
        "occupy",
        "regiment",
        "noncombatant",
        "naval",
    ],
    "abortion": [
        "abortion",
        "birth control",
        "contraceptives",
        "condoms",
        "abortion laws",
        "feticide",
        "abortion clinic",
        "pro-choice",
        "pro choice",
        "prochoice",
        "abortion pill",
        "trimester",
        "first trimester",
        "planned parenthood",
    ],
    "health": [
        "healthcare",
        "ppe",
        "personal protective equipment",
        "covid",
        "health care",
        "health",
        "coronavirus",
        "covid-19",
    ],
}
```

Appendix A: All of the aspects that our ABSA algorithm is searching for in Tweet. See Appendix B for details on how searches are operated.

Appendix B

```
if detect(row['tweet']) == 'en':
    results = nlp(row['tweet'], aspects=needToRun);
    for term in needToRun:
        if term in row['tweet'].lower():
            if results[term].sentiment == absa.Sentiment.negative:
                data.at[index, term] = 1
            elif results[term].sentiment == absa.Sentiment.positive:
                data.at[index, term] = 2
        else:
            data.at[index, term] = 0
```

Appendix B: Code snippet from sentimentCacher.py detailing how the ABSA works. As long as the tweet is in English, we search through the ABSA results. If the word was in the tweet, we get the associated sentiment, and if not, it defaults to 0.

Appendix C

```
def limit_handled(cursor, finished, total, diction):
   while True:
      try:
         yield cursor.next();
      except tweepy.RateLimitError:
         print("Rate limit reached. Preparing new process...");
         toBeCompleted = diff(total, finished);
         with open("finishText.txt", "w") as txt_file:
            for line in toBeCompleted:
               txt_file.write(line + "\n");
         tweets = pd.DataFrame(diction, columns=['user', 'tweetText', 'simplePola
rity', 'nouns']);
         print("Exiting file...");
         subprocess.Popen(["py", "twitterAPI.py", "finishText.txt"], shell=True);
         sys.exit();
      except tweepy.TweepError:
         print("Error processing the user... Skipping now");
         break;
      except StopIteration:
         break;

def createFrame(api, tweetsToPull, users):
   diction = [];
   finishedUsers = [];

   for user in users:
      params = user.split(", ");
      if len(params) < 2:
         continue;
      userName = params[0];
      party = params[1];
      state = params[2];
      if state not in stateBreakdown.keys():
         stateBreakdown[state] = {party: 1};
      cursor = tweepy.Cursor(api.user_timeline, screen_name=userName, include_rts
            =True, tweet_mode='extended').items(tweetsToPull);
      for post in limit_handled(cursor, finishedUsers, users, diction):
         cleanedTweet = clean_tweet(post.full_text);
         if cleanedTweet[0:2] == 'RT':
            cleanedTweet = cleanedTweet[2:];
         blob = TextBlob(cleanedTweet);
         polarity = blob.sentiment.polarity;
         nouns = blob.noun_phrases;
         diction.append([userName, cleanedTweet, polarity, nouns, party, state]);
      finishedUsers.append(user);
```

Functions from the twitterAPI.py file, demonstrating how tweets are pulled from users and how rate-limiting in the Tweepy library is handled. Since we have to create a new cursor once rate-limits have been reached, we simply restart the process again to handle new users.

Appendix D

```
def createXGBClassifier(bag, dataset, y_train):
    '''
    Parameters:
        dataset:
            Type: np.Dense
            TFIDF/BoW Data
        sentiments:
            Type: np.Matrix
            All ABSA analysis for texts.
        y_train:
            Type: np.Array
            Array of all training data.
    Returns trained XGB Classifier.
    '''
    dataset = dataset.drop(['tweet', 'party'], axis=1);
    sentiments = dataset.to_numpy();
    x_train = np.append(bag, sentiments, axis=1);

    xgb_model = XGBClassifier(random_state=9,learning_rate=0.3)
    xgb_model.fit(x_train, y_train)

    filename = '../data/user_models/xgboost_model'
    pickle.dump(xgb_model, open(filename, 'wb'))
    return xgb_model;

def vectorize(dataframe):
    '''
    Parameters:
        dataframe
            Type = pd.DataFrame
            Contains all training data.
    Converts all text into a bag of words.
    '''
    bow_vectorizer = CountVectorizer(stop_words='english')
    if 'parsed_text' in dataframe.columns:
        bagOfWords = bow_vectorizer.fit_transform(dataframe['parsed_text'])
    else:
        bagOfWords = bow_vectorizer.fit_transform(dataframe['tweet'])

    x_train = bagOfWords.todense();
    y_train = dataframe['party'];

    filename = '../data/user_models/vectorizer_model'
    pickle.dump(bow_vectorizer, open(filename, 'wb'))

    return x_train, y_train, bow_vectorizer;

def main():
      bagOfWords, y_train, vectorizer = vectorize(data);
      xgb_classifier = createXGBClassifier(bagOfWords, data, y_train);
```

Code snippet of how the XGB Classifier model is created for training purposes. Uses the imported CSV data, as well as bag-of-words data object, to create the classier. These models are also exported through the Pickle library for use in the interface. The exact same system is used to create the Logistic Regressor, just with a different model.

Appendix E

```
def test_and_predictionsV2(classifier, topics, vectorizer, testingData):
    '''
    Parameters:
        classifier:
            Type: XGBClassifer or LogisticRegressor
            Classification system.
        topics:
            Type = Dict
            Contains all topics and keywords.
        vectorizer:
            Type = TFIDF or Count Vectorizer (Bag of Wordss)
            Vectorization system for word frequency.
    Get multiple rows at once, parse them all, use that for prediction.
    '''
    data = pd.read_csv(filename)
    data = data.dropna();
    correct = 0;
    incorrect = 0;

    data = data.head(data.shape[0]);
    bagOfWords = vectorizer.transform(data['tweet']);
    bagOfWords = bagOfWords.todense();
    sentiments = data.drop(['tweet', 'party'], axis=1).to_numpy();
    totalInformation = np.append(bagOfWords, sentiments, axis=1);
    results = classifier.predict(totalInformation);
    counter = 0;
    corrects = {};
    incorrects = {};
    for index, row in data.iterrows():
        if results[counter] == row['party']:
            correct = correct + 1;
            if row['party'] in corrects.keys():
                corrects[row['party']] = corrects[row['party']] + 1;
            else:
                corrects[row['party']] = 1;
        else:
            incorrect = incorrect + 1;
            if row['party'] in incorrects.keys():
                incorrects[row['party']] = incorrects[row['party']] + 1;
            else:
                incorrects[row['party']] = 1;
        counter = counter + 1;
    return correct, incorrect, corrects, incorrects;

def main():
    correct, incorrect, corrects, incorrects = test_and_predictionsV2
      (xgb_classifier, topics, vectorizer, testingFilename);

print("Out XGBoost got " + str(correct) + " out of " + str((correct + incorrect))
      + " right. This equates to a " + str(correct/(correct+incorrect) * 100) +
      " accuracy level.")

    print("The regressor got a " + str(corrects['R']/(corrects['R']+
      incorrects['R']) * 100) + " percent accuracy level for Republicans,
      and a " + str(corrects['D']/(corrects['D']+incorrects['D']+
      incorrects[' D']) * 100) + " percent accuracy level for Democrats.")

    correct, incorrect, corrects, incorrects = test_and_predictionsV2
      (logistic_regressor, topics, vectorizer, testingFilename);

    print("Our LogisticRegressor got " + str(correct) + " out of " +
      str((correct + incorrect)) + " right. This equates to a " +
      str(correct/(correct+incorrect) * 100) + " accuracy level.")

    print("The regressor got a " + str(corrects['R']/(corrects['R']+
      incorrects['R']) * 100) + " percent accuracy level for Republicans, and
      a " + str(corrects['D']/(corrects['D']+incorrects['D']+
      incorrects[' D']) * 100) + " percent accuracy level for Democrats.")
```

A code snippet detailing how the testing data is parsed and tested on the training models. We use the previously saved bag-of-words model (see Appendix D) to create a new bag for the testing data. After preforming some processing, we test the data and print out the results to determine how each model did for both parties.