Maybe Twitter Actually Is Real Life: An In-Depth Examination of the Connections Between Twitter and Politics

The Harvard community has made this article openly available. Please share how this access benefits you. Your story matters

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Citable link</td>
<td><a href="https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37364696">https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37364696</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at <a href="http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current/terms-of-use#LAA">http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current/terms-of-use#LAA</a></td>
</tr>
</tbody>
</table>
Maybe Twitter Actually Is Real Life: An In-Depth Examination of the Connections Between Twitter and Politics

Andrew W. Shackelford
Department of Computer Science
Harvard University

Abstract

Despite its third-place rank in total users, Twitter represents the most popular social media platform for politicians to communicate with their constituents. In addition, journalists, pundits, and politicians alike have formed a so-called “Political Twitter” bubble that has an outsized influence on policy, media, and politics itself. In this thesis, we examine the relationships between politics and Twitter. We analyzed how partisanship affected politicians’ willingness to tweet about politically toxic events such as impeachment, and found a strong correlation between the partisanship of a given politician and their willingness to tweet about impeachment. We also found that politicians in so-called “crossover” districts, where the representative’s political party does not match the party that their district chose for the presidency, are much less likely to tweet about impeachment than counterparts in non-crossover districts with the same level of partisan lean. Having examined one side of the relation, we then attempted to use Twitter as a predictor for Democratic primary polling. After finding limited results for national primary polling, we focused on the more limited domain of Democratic debate performances. We found strong evidence that the number of times a candidate’s name and username are mentioned during a debate are powerful predictors for polls of that candidate’s debate performance. We conclude by examining the implications of our findings, suggesting that Twitter can in fact be a valid predictor for certain more high-engagement domains, and that despite Twitter’s reputation as hyper-partisan, politicians perform a remarkable amount of self-mediation on the platform.
Acknowledgments

This thesis never would have happened without the support of innumerable people.

Foremost, I would like to thank my advisor Jim Waldo, who introduced me to this topic all the way back in the fall of 2016. His mentorship and guidance throughout my Harvard College career has been invaluable, and he has positively impacted my collegiate experience unlike any other professor. I would also like to thank Professor Mike Smith for agreeing to serve as a reader, and for going above and beyond by offering useful criticism and feedback throughout the writing process.

I would like to thank my family who support and encourage me in everything I do, and whom I can always count on for anything. Mom, Dad, and Matthew, this would never have been possible without you.

Lastly, I would like to thank my friends who have been with me throughout the process of writing this thesis, providing much needed entertainment during study breaks and making my Harvard experience incomparable. Our four years may have been cut short to 3.75, but we made the most of them.
## Contents

1 Background 7

2 Literature Review 10
   2.1 Sentiment Analysis on Twitter 10
   2.2 Classifying Political Text 11
   2.3 Predicting Poll Trends and Election Outcomes 12
   2.4 Takeaways 13

3 Data Gathering 14
   3.1 Twitter’s API 14
   3.2 Random Sample 15
   3.3 Political Sample 20
      3.3.1 Filtering Options 20
      3.3.2 The Downsides of Hashtags 21
      3.3.3 Building a List of Accounts to Follow 26
   3.4 Comparing the Political and Random Samples 29
   3.5 Verifying the Completeness of the Political Sample 30

4 Data Analysis 33
   4.1 The Anatomy of a Single Tweet 33
   4.2 Location Information 35

5 The Effects of Partisanship on Twitter Behavior 37
   5.1 Process 38
      5.1.1 Building a Tweet Classifier 38
      5.1.2 Model Selection 39
      5.1.3 Data Pre-Processing 40
      5.1.4 Feature Selection 42
      5.1.5 Data Filtering 45
      5.1.6 Hyperparameter Tuning 46
   5.2 Results 48
      5.2.1 Engagement 50
      5.2.2 Political Party 51
Buttigieg winning Iowa would definitely be the strongest “politics Twitter is not real life” result
welp. i wrote and published my hottest take. Twitter is real life.
1 Background

Twitter represents one of the most popular social media networks today, ranked third most trafficked overall and with over 81 million users in the United States as of September 2019. [3] Twitter is even more popular in the field of politics than social media as a whole – just look at the President’s Twitter account. Despite the fact that more than twice as many Americans use Facebook compared to Twitter, as of March 3, 2020, President Trump’s Twitter account has 73.3 million followers compared to his Facebook page’s paltry 26 million. Similar ratios are true for former President Barack Obama, House Speaker Nancy Pelosi, and Senator (and presidential candidate) Bernie Sanders, among others. Clearly, Twitter is the preferred method for politicians to interact with the American people. Despite this, pundits often claim that “Twitter is not real life” – that despite the fact that over one in four Americans use Twitter at least once a month, it is not representative of America at large. [4]

To an extent, this is true. Research has shown that Democrats who post often on Twitter are more likely to have a college degree, be white, and have donated to a political organization in the past year than Democrats who don’t. [5] Clearly, there is a disconnect between activity on Twitter and real-world outcomes. However, that disconnect may be smaller than most assume. Some have recently argued that despite Twitter’s demographic differences, its influence on politically active journalists and “elites” affects the framing of news stories and can lead to the real world imitating Twitter. [4]

Whether through this effect or improved demographic analysis, we aim to better examine the connections between politics and Twitter in this thesis. We focus
specifically on United States national politics and how Twitter is both affected by
and can predict political events. By doing so, we aim to better understand the
relevancy of Twitter in politics and how it has the potential to be used in meaningful
ways to aid political campaigns. First we start by examining past literature on
the use of Twitter for political research – through sentiment analysis, political text
classification, and election outcome predictions. Then, we set about determining
the specifics of gathering a dataset for our use – what data should we gather, why
should we gather it, and how can we most effectively capture tweets about politics?
We examine both the soundness of Twitter’s default random sample and the tools
that Twitter provides for situationally pertinent data collection. Finally, we conclude
the first half of our thesis by dissecting the specifics of gathered tweets and what
relevant information they may contain.

With our data gathered, we then set about determining its usefulness to the realm
of politics. We focused on two issues relevant during the data collection period
– the impeachment trial of President Trump, and the 2020 Democratic primary
campaign. We first look at how politicians tweet about the politically toxic subject
of impeachment. To do so, we built a classifier that could determine the subject
of a tweet – whether it was related to impeachment, politics as a whole, or neither
– with over 82 percent accuracy. We then used this classifier to label tweets from
our sample, and examined how partisanship affected the willingness of members
of Congress to tweet about impeachment. We found statistically significant results:
congresspeople are less willing to tweet about impeachment if they are less partisan,
in a less partisan district, or in a district that voted for the opposite party for the
presidency. After examining the effects of partisanship on Twitter behavior, we
then attempted to use Twitter to predict political outcomes. We started by gathering
relevant features for our predictions. Then we attempted to use this dataset to predict
trends in national primary polling averages, with limited results. Following this, we
then applied the dataset to the more limited domain of Democratic debates, whose viewers are usually more politically active and engaged on social media. In our analysis of these debates, we found promising results suggesting that simple two-feature models can be remarkably accurate in predicting polls of debate performance across almost all candidates. While hampered by our limited sample of four debates, we found high correlations with models whose simplicity and intuitiveness suggested that they did not overfit our data.

Lastly, in our conclusion, we review the outcomes of this thesis, suggest future research, and interpret the significance of the results for the ever expanding field of data in U.S. politics.
2 Literature Review

In this thesis, we examine two main uses of Twitter as a dataset – to analyze political activity on Twitter and to use Twitter to predict polling outcomes. Therefore, we reviewed relevant literature on these subjects so that we may take ideas from past work and apply them to our research domain. In order to aid our attempt to classify the subject of tweets, we examined work on political text classification. For the purposes of predicting polling outcomes, we reviewed past work on sentiment analysis, predicting election outcomes, and forecasting primary trends.

2.1 Sentiment Analysis on Twitter

Many researchers have looked at the efficacy of Twitter as a dataset, specifically for the purposes of sentiment analysis. In their paper Sentiment Analysis of Twitter Data, Agarwal et al. aimed to build a model for classifying tweets with positive or negative sentiment. [6] They found that performing certain preprocessing tasks on tweets such as replacing emoticons with a pre-defined sentiment polarity, replacing all mentions with a certain tag, and replacing all negations with a single tag helped improve their sentiment classifier’s accuracy. They also found that replacing sequences of repeated characters in a word (such as “coooooooooooool”) with three of that character (“coool”) helped to lower the feature space and improve the accuracy of their classifier, while still distinguishing between the original word and the emphasized version of the word.

In their paper Twitter Sentiment Analysis: The Good the Bad and the OMG!, Kouloumpis, Wilson, and Moore also examine methods of Twitter sentiment analysis – specifically what kind of features are useful for this task. [7] They found that features based on part-of-speech were not useful, and that using an existing sentiment lexicon dictionary helped but was much less useful than n-grams based on the tweet text itself.
Finally, Pak and Paroubek have examined sentiment analysis on Twitter in *Twitter as a Corpus for Sentiment Analysis and Opinion Mining*. [8] They used similar methods as other researchers, but also introduced two new features – “entropy” and “salience” – to filter n-grams based on their relevance to the classification task at hand. For their model, “entropy” represents the Shannon entropy of the probability distribution of a given n-gram in the different classifications. A high value of entropy in this case would indicate that the n-gram appears with roughly equal probability in any classification, and as such is not a very useful feature. Their second new feature, “salience,” measures the differences in probability of an n-gram appearing in a given classification. This ensures that even if an n-gram appears in one classification once and all other classifications zero times, it won’t be very salient since it still appears rarely even in the classification that it appears in 100% of the time. Pak and Paroubek found that while both salience and entropy were effective features, salience provided a better accuracy and was thus a more useful discriminative feature.

### 2.2 Classifying Political Text

There has also been previous work on classifying political text in particular, both on and off Twitter. In their paper *On Classifying the Political Sentiment of Tweets*, Johnson, P. Shukla, and S. Shukla attempted to classify the sentiment of tweets specifically towards Barack Obama. [9] They, like Agarwal et. al, also removed repeated characters and used certain other feature reduction techniques by removing URLs and usernames. In their results, they determined that Twitter is likely a better source for gauging the sentiment of the public towards current events than it is for determining the general sentiment of the public towards a politician.

Chang and Masterson also looked at classifying political text in *Using Word Order in Political Text Classification with Long Short-term Memory Models*, however they focused on Chinese social media posts and US newspaper articles. [10] They specifically examined the use of a Long Short-term Memory model (LSTM) as
opposed to other machine learning models. LSTM models can take into account the order of words in ways many other models can’t, and can be useful for situations in which words occurring earlier in a text alter the meaning of words occurring later. In their work, they found that LSTM models work best with large training datasets and in situations where the distribution of classification categories is relatively balanced. However, they found that Support Vector Machines outperformed LSTMs significantly on unbalanced datasets where some classifications are more common than others.

2.3 Predicting Poll Trends and Election Outcomes

In their work Leveraging Candidate Popularity On Twitter To Predict Election Outcome, Gaurav et al. looked predicting election outcomes in Latin America using Twitter data. [11] They looked specifically at three presidential elections during a period of three months in 2013, and used the number of times a candidate’s name was mentioned as a predictor. They used not only the candidate’s full name, but also merely their first or last names as keywords. By the use of context clues such as the word “elecciones,” they were able to determine whether common names such as “maduro” referenced a specific candidate or were unrelated to the election.

Mirowski et al. looked specifically at poll trends over time in their paper Predicting Poll Trends Using Twitter and Multivariate Time-Series Classification. [12] They used a Linear Time-series Shapelet (LTS) model to attempt to predict whether polls would increase or decrease in the future for primary candidates in the 2016 elections. They used features such as the average sentiment, average sentiment among unique users, and number of unique users per sentiment as features for their model. They found greater success for their multivariate time-series models compared to other models, but found an accuracy at best of less than 75 percent.
2.4 Takeaways

From past literature, we take away several key insights. First, we note several useful data techniques that aid in text classification in the domain of Twitter: removing sequences of repeated characters in a word, removing URLs and specific usernames, as well as using “entropy” and “salience” metrics that help measure the utility of a term for classification purposes. We also note that LSTM models, while useful for some datasets, do not work well on unbalanced datasets where some classifications are more common than others. All of these findings will be useful for our work on classifying the subject of politicians’ tweets.

For our work on predicting Democratic primary polling, we note a few useful techniques, such as using different versions of a candidate’s name as a predictor, and using context clues to determine whether common names referred to a candidate or were unrelated to the election at hand. We also find that features tied to the number of unique users can be more useful than pure aggregate statistics. Lastly, we note that Johnson, P. Shukla, and S. Shukla determined that Twitter was a better source for gauging sentiment towards specific current events than the overall sentiment towards a given politician. This insight is one of the key factors that led us to focus specifically on Democratic debates after initially starting with national Democratic primary polling.
3 Data Gathering

In this section, we detail the various methods considered to gather data from Twitter. With limited options available to us on the free tier, it was crucial to gather as representative a sample as possible that captured all of “political Twitter.” We start by detailing Twitter’s API functionality, and the available data collection options. We then dive deep into Twitter’s random sample, reviewing studies from other researchers examining its reliability, and determining if our sample matches their results. After studying our random sample, we begin to determine the best possible way to gather a sample of the political tweets we aim to focus on, a “political sample,” as it were. We consider all three of Twitter’s available filtering options: locations, hashtags, and users. We first discuss the drawbacks of locations and hashtags, and then examine the benefits of filtering by users. After settling on user filtering, we compile a list of users to filter by that encapsulates the vast majority of political tweets. Finally, we conduct experiments to verify the accuracy of our political sample, ensuring its relevancy for analysis.

3.1 Twitter’s API

Twitter has had a robust third-party API since soon after its launch in March 2006. Twitter initially started as a service primarily interacted with through SMS – the main reason for its initial 140-character limit. Because of this, Twitter deferred working on its own apps, instead letting third-parties use its API to craft new clients for any platform. Not until Twitter’s acquisition and rebranding of the popular third-party client Tweetie in April 2010 did Twitter launch its own official mobile apps on iOS and Android. [13; 14] In concert with these official client launches, Twitter began limiting its third-party API, starting by limiting the access rates that clients had to data as well as explicitly telling developers how their apps should display data, and encouraging them to simply stop working on creating third-party clients. [15] These
changes started a progressive watering down of Twitter’s API, to the point where it now does not include all of Twitter’s feature set. In fact, despite Twitter introducing polls as a feature in October 2015, its API still allows no way for anyone, regardless of price tier, to access poll results in a tweet, or even detect that a poll is present in a tweet.

In addition, most API features are restricted to accounts with a commercial relationship with Twitter. Twitter separates its API access into “Standard,” “Premium,” and “Enterprise.” Standard access can be obtained within a few minutes of filling out an API access request form, and allows for basic search queries, stream filtering, and posting functionality. Premium access allows better access to searching features, but at a significant price that can escalate quickly. Just 500 queries of tweets posted within the past 30 days costs $149, and the maximum allowed 10,000 queries per month costs $2,499. Searching tweets from all time costs even more per query, with a maximum allowed 2,500 queries per month costing a whopping $1899. For greater access than the Premium API allows, prospective developers can contact Twitter for an Enterprise account, for which pricing is not publicly available. [16]

3.2 Random Sample

For the purposes of this project, two main datasets were collected, the first of which was a random sample of all of Twitter. Ideally, a dataset of every single tweet over a period of time would be gathered for complete analysis. However, Twitter ceased providing access to this so-called “firehose” in 2015, thus making it impossible for anyone, regardless of financial resources, to have access to the entirety of Twitter’s data. [17] This is not to mention that the resources such a dataset would require. As of 2018, there are an estimated 481,000 tweets sent per minute, for approximately 700 million per day. [18] With 10,000 tweets taking up approximately 60 megabytes of space, even just a minute of tweets would require 2.9 GB of storage, and a day would require an astounding 4.2 terabytes. Not only would this quickly saturate
even an extremely high capacity hard drive, but gathering such a dataset would likely
overwhelm Harvard’s proportioned internet connection per user, requiring almost
half a gigabit per second of bandwidth.

Twitter does, however, provide some access through its “decahose” product. Twitter’s
decahose is a 10% random sample of all tweets, available only to its Enterprise
clients. Pricing for the decahose has not been disclosed publicly and likely costs in
the hundreds of thousands or even millions per month.

Lastly, Twitter does provide limited access to a “small random sample” of all
public tweets through its standard “sample stream.” Twitter does not publish any
statistics regarding the current size of the stream. Twitter has previously stated that
the stream constituted a one-percent random sample, and empirical investigation
seems to support that the current stream is still in fact a one percent random sample.
According to research by Kalev Leetaru, the Twitter stream constitutes approximately
a 0.98% sample of all tweets during the period since February 2015. Not only this,
but the daily tweet volume between the stream and Twitter’s overall volume is almost
perfectly correlated, with $r = 0.987$ for all tweets, $r = 0.996$ for non-retweets, and
$r = 0.877$ for (much more rare) verified tweets. [19]

However, the fact that Twitter’s stream is a true one-percent sample does not ensure
accurate analysis. Although the size is accurate and the daily tweet volume is highly
correlated, for Twitter’s one-percent sample to be a useful data analysis tool, it must
also constitute a true random sample of tweets across a variety of topics, regions,
and languages. To verify that the sample is truly random, Leetaru also performed
searches for different keywords to ensure that the sample accurately captured these
trends. Leetaru searched for multiple different topics, each with different levels of
activity on Twitter. These different keywords are reproduced in Figure 1.

Leetaru’s research showed that even on rarely used keywords like “IPCC,” the daily
volume of tweets overall and the daily volume within Twitter’s random sample are
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Number of total tweets</th>
<th>Number of tweets per day</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Climate Change</td>
<td>83.6 million</td>
<td>33,500</td>
<td>0.96</td>
</tr>
<tr>
<td>Global Warming</td>
<td>26.1 million</td>
<td>10,500</td>
<td>0.94</td>
</tr>
<tr>
<td>IPCC</td>
<td>2.1 million</td>
<td>850</td>
<td>0.94</td>
</tr>
<tr>
<td>Carbon Sequestration</td>
<td>60,500</td>
<td>24</td>
<td>0.79</td>
</tr>
</tbody>
</table>

Figure 1: Mentions of Keywords in Random Sample [19]

highly correlated with an $r$-value of 0.94. Even topics as extremely low-volume as “carbon sequestration,” which averaged only 24 tweets a day and less than ten tweets per day for over a third of the sampling period, produce a remarkably high correlation value of 0.79. This research suggests that the Twitter sample is a genuine random sample of all of Twitter, and can be used for valid analysis of trends regardless of size or subject.

To gather tweets from Twitter’s random sample for analysis, we wrote a Python script that would run continuously and download tweets from Twitter’s streaming API. This script used the popular third-party Python library Tweepy – a library that allows developers to interface directly with Twitter’s API without dealing with complex HTTP streaming requests. For our uses, we simply concatenate all tweets as they download into a list for post-processing – no filtering is done in real-time. Our script automatically breaks this list into JSON files of approximately 10,000 tweets as they stream in, resulting in file sizes of approximately 60 MB per 10,000 tweet sample. A copy of the script is available at Appendix A.

This script is hosted within a Google Drive and run nonstop on a server, automatically uploading collected files to Google Drive as they are created. We began running this script on June 10, 2019, and ran it continuously until midnight on January 1, 2020. Looking at the number of tweets per day over time in Figure 2, we see that we average approximately 3.6 million tweets per day, excluding a period from June 20th to June 23rd when the server running our script lost power and didn’t gather data for approximately three days. To remove this gap in data as well as round off our
sample, we will only analyze the sample from July 1, 2019 at midnight to January 1, 2020 at midnight. This gives us a six-month sample consisting of the entire second half of 2019.

We notice that we only have approximately half of the estimated number of tweets per day that should be in our sample. According to one source [18], there are approximately 692 million tweets per day, and therefore should be approximately 6.92 million tweets per day in our sample. Yet another source [20] claims that there are approximately 500 million tweets per day for 5 million tweets per day in our sample. Both of these numbers are larger than our 3.6 million per day number, however they are both extrapolated from a “tweets per minute” figure that may represent a peak that is not representative of the average throughout an entire day. Regardless, our figure is well within an order of magnitude of other reported figures, and represents at least a 0.5% representative sample of all of Twitter. At the very least, all calculations inferred from the random sample can be assumed to be accurate within a factor of 2.
This random sample consists of 661,136,301 tweets during our six-month period of data collection, which we will assume represents a 1% sample of Twitter. With some back-of-the-envelope calculations, this suggests that over this period of time a total of 66.1 billion tweets were posted on Twitter. Our sample averages a whopping 21.42 gigabytes per day for a total of 4.35 terabytes overall.

Using the tweet’s built-in “language” parameter (specified by the user), we see that approximately 30% of the sample is in English, as shown in Figure 3. We also notice periodic fluctuations in the English proportion of tweets, with 26 total periods over the six month-long period. This suggests a period of about 7 days, lining up with the weekday/weekend schedule. We also notice some changes in this periodic schedule coinciding with American holidays, such as July 4th and Thanksgiving.
3.3 Political Sample

3.3.1 Filtering Options

While the random sample has its benefits, it is also limited in that it contains a random sample of all of Twitter, including vast portions that are not related to politics. Only 30% of the sample is in the English language, and an even smaller fraction of that sample pertains to the U.S. political sphere, especially at a national level. In order to better target this specific section of Twitter, we needed a different way to gather our corpus of tweets. In addition to its random sample, Twitter allows users of the API to “filter realtime tweets,” that is, download tweets that mention specific people, keywords, or hashtags in realtime. Twitter does not explicitly say whether this sample is all-inclusive and contains every single tweet that falls under the specified constraints, or if the tweets delivered constitute merely a subsample of the entire set. [21] However, we perform some analysis to determine whether the downloaded tweets constitute a full sample in Section 3.5. In addition to the filter API, Twitter allows for developers to search for specific tweets based on tokens like the user handle, tweet mentions, and tweet text. However, the public version of this API only allows for searches of tweets within the past 7 days, and is not particularly helpful for our analysis.

With access to the standard-level API, Twitter allows users of the real-time filter endpoint to filter by up to 5,000 users, 400 hashtags/keywords, or 25 location areas. [22] However, due to the deprecation of the location geotagging feature as discussed in Section 4.2, filtering by location area is essentially irrelevant. These limits are additive, that is, a developer can gather data from up to 5,000 users as well as tweets containing 400 different hashtags or keywords, and tweets within 25 location areas. With the Enterprise API, these quota limits increase to 250,000 filters, each with up to 2,048 characters defining how the filter should be applied. Crucially, Twitter differentiates its API offering based on the amount of keywords one can filter by, but
not the quality of the results themselves. This suggests that the results themselves are complete, and encompass the entire specified sample.

Now that we had our method to gather a sample of so-called “Political Twitter,” we had to determine what exactly that meant. “Political Twitter” is a sort of catch-all phrase that can have many meanings depending on who one asks. For the purposes of our analysis, we aimed to look at the national United States government in particular. On top of that, we aimed to look mostly at the political interactions themselves, and not the more mundane day-to-day governmental processes. Even with this set of specifications, it was difficult to determine exactly how to gather a representative sample. Should we filter by content, such as through hashtags and keywords, or instead focus on users themselves?

### 3.3.2 The Downsides of Hashtags

This decision did not have to be made without context. In the fall of 2016, I participated in a research group with Professor Jim Waldo [24], also researching the feasibility of using Twitter data to analyze politics. This group, Twitter Predicts 2016 [25], also gathered its data using the “filter realtime tweets” API. We specifically used political hashtags to gather the data, and also used those hashtags to classify the data. However, hashtags present problems both for data labeling and data collection. Consider the tweet in Figure 4, which uses the common “#MAGA” hashtag, short for “Make America Great Again,” President Trump’s 2016 campaign slogan. This hashtag at first glance is a pro-Trump hashtag, and would mark anyone who used it as a Trump supporter. However, context matters, and the hashtag is occasionally used by opponents of Trump in a sarcastic manner to highlight failures of Trump or the Trump Administration, as it is in this case. There are many hashtags that are sometimes used in this manner, and for this reason hashtags are not a great metric to evaluate the true sentiment of a tweet.
There also exist larger problems with using hashtags to collect data. For unknown reasons – perhaps the age skew between Democrats and Republicans – there exist both more, and a much higher use of, right-leaning, pro-Trump, pro-Republican hashtags. [25] This could arise from a variety of factors, including but not limited to better social media organization among Republicans, the presence of bots that use hashtags to spread their message, and a recent trend of declining hashtag use that older, more right-leaning, Twitter users may not have followed. This trend of declining hashtag use also poses problems for collecting data in general. Hashtag use has fluctuated significantly over time. In 2010, the percent of tweets with a hashtag was 14% for tweets written in English, and 11% overall. [26] As shown in Figure 5, the use of hashtags steadily increased to a point until mid-2016, at which point the percent of tweets with a hashtag fell drastically and continued to decline slowly. Even with a peak of approximately 47%, hashtags are not used in the majority of tweets, and were used in less than 40% of tweets in late 2018.

In our random sample for both English-language tweets and all languages, we find that hashtags are used even less than past research suggested. For all tweets as shown in Figure 6, only approximately 17% of tweets contain at least one hashtag. Hashtags are even rarer for English-language tweets, for which, as shown in Figure 7, only 13% of tweets contain at least one hashtag. As such, using hashtags to filter tweets would exclude a large percentage of our potential tweet “population,” as it were. This sharp downturn suggests that hashtag use will continue to decrease in the future, making hashtags even less relevant to data analysis of Twitter.
Figure 5: Variation in Hashtag Frequency Over Time [27]

Figure 6: Hashtag Use in All Languages, Random Sample
In addition, hashtags by design are meant to cater to social events and new trends, and as such, change constantly and are difficult to track. Finding comprehensive lists of politically relevant hashtags is exceedingly difficult, and these hashtags are subject to change all the time. In the aftermath of breaking news events such as the shooting at a Walmart in El Paso, Texas, the hashtags “#ThisIsAmerica” and “#WhiteSupremacistTerrorism” skyrocketed in use for a period of approximately 48 hours, as seen in Figures 8 and 9. The use of these hashtags quickly decayed, however, and by the time they could have been added to our prospective filter list, most of their use had already passed. Despite their short period of use, the two hashtags were used over 250,000 times in a two day period, and as such would represent a large number of tweets to be missed.

For similar reasons, keywords fall prey to these same problems as well, and as such we decided to pick users as our filtering method for our political sample. Keywords are especially problematic due to the international nature of keywords like “climate
Figure 8: Use of #ThisIsAmerica and #WhiteSupremacistTerrorism Per Day, Random Sample

Figure 9: Use of #ThisIsAmerica and #WhiteSupremacistTerrorism Per Hour, Random Sample
change,” “budget,” or “taxes” that are relevant subject matter in any nation around
the world, not just the United States.

Not only is filtering by users not subject to these aforementioned problems, but
this method of sampling also best achieves our desired outcome of capturing the
so-called “Political Twitter” bubble. This method allows us to specify exactly who
falls into our bubble, and allows us to exclude those outside U.S. national politics
that are piggybacking onto specific popular hashtags and keywords.

Most importantly, filtering by users actually helps generate essentially the largest
possible sample of the political sphere, as filtering by user also gathers any tweets that
reply to, retweet, or mention a given user. This helps provide important context about
the relative popularity of both certain users and specific tweets. Additionally, each
of these replies and retweets gives us information about the users who interact with
them, including but not limited to that user’s popularity, their own self-description,
and when their account was created. Filtering by users has given us access to a total
of 13,205,571 different users of Twitter in one way or another. A distribution of the
number of tweets gathered from each user is available in Figure 10. The number of
tweets gathered is on the x-axis, and the number of users who have that number of
tweets gathered is on the y-axis.

3.3.3 Building a List of Accounts to Follow

Now that we’ve established the best method to filter tweets for our political sample,
we must construct the list of users with which to filter. Of course, national politicians
such as members of Congress and the President must be on this list, but compre-
prehensive lists of government officials’ Twitter accounts are surprisingly difficult to
find. Twitter does have a feature called “lists,” which allow users to create lists of
Twitter accounts for other users to follow or look at. Twitter’s official government
and politics account, ‘@TwitterGov,’ maintains lists of members of both the House
and Senate, the Presidential Cabinet, and more. These lists form a comprehensive
set of both official and personal accounts of politicians and official government agencies. In addition, other lists curated by media networks like C-SPAN, Fox, CNN, MSNBC, the New York Times, and others create a thorough set of political journalists, fact-checkers, pundits, TV hosts, and more. With these combined, an extensive list of Twitter accounts both covering and within the U.S. political sphere can be obtained. The full set of Twitter lists is recorded at Figure 11. Additionally, we manually added a few accounts – mainly family members of the President in the political sphere – to round out the sample. Lastly, we added every account that Trump personally follows – thus gaining an idea of what activity he is seeing on his Twitter feed. With these lists specified, we used the Twitter API to create a union of all lists and output the results to a JSON file. The code to perform this is located at Appendix B. A full, alphabetized list of all 2,575 accounts is located at Appendix C. In order to mitigate Twitter’s rate-limiting constraints, we split the list of accounts into three and created a separate subprocess for each list. The parent process restarts each subprocess every hour to prevent HTTP errors that manifest when streams are
kept open for days at a time. The code for the parent process and subprocesses is available at Appendix D.

This stream was completely uninterrupted except for one period of time in August when the server running the script briefly lost power, which knocked out the script over a weekend until we could manually restart it. This interruption began at approximately 9:45 AM on August 23rd, and lasted until 6:42 AM on August 26th, as shown in Figure 12. Quite fortuitously, the interruption occurred over a weekend during a period of no major news events, and did not hamper our analysis in any significant manner. While this interruption is obviously not ideal, it should not have any meaningful impact on the results. All graphs showcasing this data have
smoothed over the period of missing data using a moving average, so that the missing data does not impart inaccurate trends in our results. It is important to note that this disruption only affected our political sample – the random sample was running on a separate server that was not affected.

### 3.4 Comparing the Political and Random Samples

In Figure 12, we observe the difference in tweets collected per day between the political and random samples. In total, the political sample collected 368,435,455 tweets from August 6, 2019 to December 31, 2019, while the random sample collected 522,747,138 tweets over the same period. Assuming that the random sample did in fact gather a 1% sample of all of Twitter, this suggests that the political sample gathered an impressive 0.7% sample of all of Twitter, even more so impressive given its restriction to politically relevant, by-and-large English-language tweets. In fact, as shown in Figure 13, the political sample actually contains over
twice as many English language tweets as the random sample, thus capturing an estimated 2.1% of English-language Twitter.

Smoothing out our graph using a one-week moving average, as in Figure 14, we see a much clearer trend emerge. During politically relevant moments such as House Speaker Nancy Pelosi’s announcement of the impeachment inquiry into Donald Trump and the testimony of relevant officials, the number of tweets per day skyrockets. During U.S.-centric holidays such as Thanksgiving and Christmas, the number of tweets decreases dramatically.

3.5 Verifying the Completeness of the Political Sample

While our political sample clearly gathered an extraordinarily massive number of tweets, it is also important that the sample be comprehensive for what it was intended to capture – namely, every tweet by a certain group of users, as well as every retweet of and reply to those tweets. Manually verifying that every tweet from the specified list was captured would prove difficult (and quickly exhaust Twitter’s API rate limit),
but one way to verify that the sample was collected as intended is to compare the number of retweets gathered with the true numbers according to Twitter. In order to make sure that the sample’s veracity holds even on the most popular of users, we chose to verify the results using ‘@realDonaldTrump,’ by far the most popular user in our sample. In order to do this, we scheduled a script to run at precisely midnight on January 1st, 2020, to gather all tweets (and associated data) from President Trump’s timeline. This data would serve as the “ground truth,” the numbers to which we would compare the results from our sample. The code to gather this data is available at Appendix E. This script was able to gather data for President Trump’s tweets extending back to September 23rd, 2019.

We then cataloged all retweets of Trump tweets from our political sample, and compared the results as shown in Figure 15. This figure shows a cumulative running percentage of the tweets in our sample, sorted by what percentage of the true number of retweets we successfully gathered. As the figure shows, we successfully captured at least 90% of retweets for half of our sample, and at least 85% of retweets for 90%
of our sample. While not perfect, this shows that our sampling method successfully captures the vast majority of retweets for almost all of our sample. These results suggest that our sample is broadly accurate, and trends based on retweets or other similar metrics should be applicable to our analysis. Given the overly massive size of our sample, we feel that capturing at least 85% of retweets for the vast majority of our sample is more than enough to study political trends on Twitter.
4 Data Analysis

With data collected, we must understand what attributes are included in each tweet and how we can use them for analysis. In this section, we begin by dissecting an individual tweet (delivered to us as a JSON object), and examining each of the data fields within it. We reproduce a sample tweet, with irrelevant fields omitted, for the reader’s understanding. We then briefly discuss location tagging on Twitter both of users and individual tweets, and the latter’s recent deprecation in June of 2019.

4.1 The Anatomy of a Single Tweet

A single tweet status from Twitter contains information about the tweet itself, the user who tweeted it, any assets included with the tweet, as well as additional information about the tweet that was retweeted or replied to, if applicable. All of this data is represented in a JSON dictionary object, ranging from the tweet’s text, to the user who tweeted it, to the hashtags used in the tweet, and the language of the user who tweeted it. If the tweet was a retweet, the tweet that was retweeted is included in full as part of the "retweeted_status". If the tweet is instead a reply to a previous tweet, only the previous tweet’s id is included, and if the tweet is a reply to a user as a whole, only the user’s id is included. In addition to these useful data fields, each tweet object also contains less relevant details about the tweet, such as the app used to post, and the user’s profile images and theme colors. In Figure 16, a sample reply to a tweet is reproduced, with irrelevant details omitted.

Of highest importance is the "text", or, if the tweet is so long as to be truncated, the "extended_tweet" ["full_text"] field, which contains the main text of the tweet. Additionally, user ["screen_name"] provides the user who sent the tweet, and "created_at" the time the tweet was posted. The "retweeted_status" field is only present if the tweet is in fact a retweet, and contains an additional tweet object representing the tweet that was retweeted. If "in_reply_to_screen_name"
Lead middle class workers and their families into the future. We want to work and earn and thrive. I donated and support Warren for POTUS.

https://t.co/HdenfWf55h.. https://t.co/QuZXoZ2cIN

@ewarren @OccupyOneLove #ElizabethWarren

is populated, this indicates that the tweet is a reply, whether to a user as a whole or a specific tweet by that user. Particularly, "in_reply_to_status_id" is only populated if a tweet is a reply to a specific tweet. For the purposes of our analysis, we will only consider a tweet to be a reply if it is a reply to a specific tweet, since users often mention political figures in attempts to spread their tweet or with very little context, such as in Figure 17. While Twitter considers this to be a reply to a specific person, we will discard it from our analysis due to the lack of signal it provides.

4.2 Location Information

Until June 18, 2019, Twitter allowed users to report their precise location along with a tweet. [28] This would be reflected in the "geo", "place", and "coordinates" data fields within each tweet. However, they removed this feature in mid-June, and as such it is only available for seven days in the random sample dataset, and is not available at all in the political sample. Additionally, the feature was rarely used by most Twitter users and could even be spoofed by those with the knowledge of how to do so. Despite removing the location feature from individual tweets, Twitter still allows users to tag any photos they post with a precise location, and allows them to report a location (as text) in their bio. However, as exemplified by Figure 18, users can often set a fake location that does not aid in data analysis. Users can also choose to not report a location at all. For these reasons, we decided not to do any analysis of location data on this dataset as it would likely not yield any useful insights.
Tweets, especially those in a political context, often contain links to news articles or press releases or embedded images, memes, or videos. These are embedded in a tweet object under the section of "entities", specifically as "media". This media contains links to the media itself, information about the size and source of the media, as well as details about the user that originally published it. Although there is some potential in analyzing how political media spreads throughout Twitter, the fact that media is often re-posted as a screenshot or with slight edits makes tracking the spread of media quite difficult. That, combined with the computational and space resources required in order to conduct such analysis, made examining such a subject infeasible, and for such reasons we decided to not attempt to do so.
5 The Effects of Partisanship on Twitter Behavior

In this section, we aim to study how politicians use Twitter, and if they change their behavior for partisan or political reasons, such as to avoid tweeting about a politically toxic subject. Specifically, we examine the effects that partisanship has on a politician’s willingness to tweet about the impeachment of Donald Trump. With the inquiry announced on September 24, 2019 and the trial concluded on December 18, 2019, this event falls wholly within our sample period which lends itself to be a useful event to analyze. First, we build a machine learning classifier to determine whether any particular tweet is specifically about impeachment, United States politics as a whole, or on a completely unrelated subject. After building, optimizing, and testing this classifier, we then use it to label all “original tweets” in our political sample – tweets posted by someone from our gathered set of lists.

We first analyze how tweets about impeachment vary from those that are not, specifically by the amount of engagement. We then analyze how partisanship affects a congressmember’s willingness to tweet about impeachment compared to their overall tweet behavior. We study these effects using the congressperson’s partisanship as determined by their voting record as well as the partisanship of their district. Finally, we examine patterns in tweet behavior of House members in so-called “crossover districts,” where a representative’s district split the vote for the House and the Presidency to different parties. With these analyses, we can determine whether real-world political characteristics do in fact affect discourse on Twitter, and ascertain which political characteristics are the most influential. We find that politicians who are less ideological and those who differ from their district’s political lean often moderate how frequently they tweet about impeachment, with the most important characteristic being the difference between a representative’s partisanship and their district’s partisanship.
5.1 Process

5.1.1 Building a Tweet Classifier

In order to analyze how politicians tweet about certain subjects, we must first classify the 1,260,226 original tweets in our sample. The basis of any machine learning classifier starts with training data, so that the classifier can learn the specific details that classify a tweet as pertaining to a certain subject. There is no real way to create this training data except manually by hand, so we took the effort to label a decently sized dataset so that our classifier could be very accurate. To do this, we randomly selected 50 original tweets from each day, and assigned them one of our three labels – {“Impeachment”, “Politics”, “Not Related”} – while recording the results. This took quite a bit of time, but by doing so, we amassed a dataset of 6,881 labeled tweets. We note that this sample is smaller than 50 tweets per each of the 147 days of gathered data. This is due to the fact that sometimes people tweet images or videos with no accompanying text, thus not providing any substantial data for the classifier to train on. As a result, we simply toss out any tweets with no accompanying text, as well as the few for which we could not parse a classification ourselves. In addition, often users will “quote tweet” someone else’s tweet and add a comment. While the comment has semantic meaning with the attached quoted tweet, Twitter does not include the quoted tweet in the API response, and as such our classifier has no access to it. For quote tweets that are semantically ambiguous without the accompanying tweet, we discard the entire tweet as well.

Now that we have a labeled dataset, we must choose the type of classifier we want to use to label the rest of our unlabeled data. There exist many, many types of classifiers, from basic Logistic Regression classifiers to Support Vector Machines, Naive Bayes classifiers, Random Forest classifiers, and more. These classifiers make decisions in different ways: some use a decision tree model, like the Random Forest and XGBoost classifiers, others use Bayesian statistics, like the Multinomial Naive
<table>
<thead>
<tr>
<th>Model Type</th>
<th>Percent Labeled Correctly</th>
<th>Precision*</th>
<th>Recall*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multinomial Naive Bayes</td>
<td>77.0%</td>
<td>0.66</td>
<td>0.79</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>80.7%</td>
<td>0.90</td>
<td>0.62</td>
</tr>
<tr>
<td>Random Forest</td>
<td>79.4%</td>
<td>0.85</td>
<td>0.61</td>
</tr>
<tr>
<td>XGBoost</td>
<td>77.1%</td>
<td>0.89</td>
<td>0.60</td>
</tr>
<tr>
<td>SGD Linear SVM</td>
<td>80.9%</td>
<td>0.85</td>
<td>0.68</td>
</tr>
</tbody>
</table>

*for tweets with “Impeachment” classification

Figure 19: Performance of Different Types of Machine Learning Models on Final Feature Set

Bayes classifier, while others use linear models, like the Stochastic Gradient Descent classifier – which uses SGD to optimize its linear models. While all perform fairly well, some perform better than others, and choosing the right model can be crucial to extracting the best performance from our classifier. While we considered using an LSTM model as discussed in our literature review, the benefits of LSTM models appear mainly with very large training datasets and a relatively even classification distribution, both of which we do not have for our analysis. For these reasons we instead focused on more traditional models.

### 5.1.2 Model Selection

In general, linear models are regarded to do well for text classification problems, due to linear models’ tendency to avoid overfitting and the high number of features inherent to text classification. As we will discuss below, we must break our text into discrete sets of numerical features, which will lead to a high dimensionality that linear models tend to handle better. However, the superiority of linear models should not be assumed, and different models should be tested to ensure that the best model is chosen in the end. Comparing Naive Bayes, Logistic Regression, Random Forest, SGD Linear SVM, and XGBoost models trained on a 75% random subset of our training data with our final set of features and then evaluated on the remaining 25%, we obtain the results in Figure 19. These results confirm that our Linear Support Vector Machine model, trained via Stochastic Gradient Descent, does in fact perform the best as anticipated.
Of particular importance is the precision and recall performance of the model. Giving more detail than just the percent classified correctly, precision and recall help us evaluate how correctly the model labels each different possible classification. Precision states the percent of true positives that are correct, that is, \( \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \). This helps evaluate how often the model incorrectly classifying a tweet as something it is not. Recall, in contrast, states the number of true positives over total positives, that is \( \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \). This helps evaluate how often the model fails to capture a tweet that should be classified. This is especially important for our model given the relatively low number of tweets about impeachment – in our training data, only 11.3 percent of tweets are about impeachment. It could be quite easy for a classifier to only learn the difference between tweets about politics and tweets about other subjects, and not be able to discern which tweets are specifically about impeachment. If such a classifier could perfectly distinguish between politics and non-politics tweets, it could label 88.7 percent of tweets correctly – better than our best model – while not correctly labeling a single tweet about impeachment. It is thus very important that our model have good precision and good recall for impeachment-classified tweets, of which the SGD model performs materially better than the rest.

5.1.3 Data Pre-Processing

After selecting our classifier type, we now need to choose the other main part of our machine learning model – what specific characteristics of each tweet (“features,” in ML parlance) we will use to classify the data. While such a hypothetical would be ideal, we can’t simply feed in the entire tweet object to our model and have it spit out an answer. Not only would such an exercise add a tremendous amount of unnecessary, confounding features (things like the unique ID of the tweet should not have any relevance to its subject), but this is also simply not possible. For most types of classifiers – including the type we’re using, an SGD classifier –
the inputs to the model must be numerical in form. As such, we can’t feed in something like the text string of the tweet to the model and be able to obtain a result. Instead, text classification relies on the creation of many features representing different characteristics of the input text. For example, one set of features could be a representation of how many times each word in the dictionary is present in the input text. This approach, known as the bag-of-words approach, is a common starting point for machine learning classifiers. Looking at the results in Figure 20, we see a promising start. Our model is able to obtain an overall accuracy of 76.9%. The labels {'i’, ‘n’, ‘p’} represent the classifications {“Impeachment”, “Not Related”, “Politics”}, respectively. In the results table, precision and recall are as defined above, “f1-score” is the weighted average of precision and recall, and “support” is the number of tweets in the test data with that classification. Specifically, f1-score is equal to \( \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \).

For our purposes, this 76.9% accuracy statistic specifically refers to the percentage of tweets correctly labeled in the test set. For this feature specification period, we used a random 75-25 train test split, where we train on a random subset of 75% of the tweets and test on the other 25%. We used a specific random seed to ensure that the split was the same each time, so we can perform an apples-to-apples comparison of our different models.

After starting with the basic “bag-of-words” approach, we tried several different data pre-processing techniques to maximize the accuracy of the model. One of the most basic techniques involves removing so-called “stop words,” a name for syntactic and
filler words that don’t carry any real meaning. In English, for example, these words include “a,” “an,” and “the.” The Python machine library that we are using to build this classifier, scikit-learn, allows us to automatically remove a built-in list of English stop words. By doing so, we improved the accuracy of our model to 78.6%.

We then followed some of the recommendations mentioned repeatedly in our literature review – starting by removing lengths of repeated characters. We took any set of repeated characters of length three or more and replaced them with a set of length three. While some methods replace with a length of two, we chose to replace with a length of three to distinguish between repeated characters for emphasis and the normal use of the word. We also removed URLs like many other examples in our literature review. We specifically chose not to remove handle mentions unlike most in our literature review, as we believe these can have a strong predictive power for our model – tweets mentioning Trump are very likely about politics and tweets mentioning Biden have a decent likelihood of being related to the impeachment inquiry. By performing these steps, we improved our accuracy to 79.0%.

5.1.4 Feature Selection

Once these filtering steps are implemented, we can set about adding more features to our model that might allow it to better classify these tweets. The next step from a “bag-of-words” approach is typically to add features for bigrams. Bigrams are unique features for each set of two words appearing one after the other. The hypothesis about bigrams is that the placement of certain words next to each other can drastically change the meaning of the sentence, even when the meaning of the individual words are indeterminate. Consider the example “ukraine scandal” and “ukraine vacation.” Both bigrams have the word “ukraine” – a key word relevant to the impeachment process – and a word largely irrelevant – “vacation” is not relevant to anything and “scandal” could apply to a pop culture event or even the ABC television show. However, when combined as a bigram, it is clear that “ukraine
scandal” represents the scandal for which Donald J. Trump was impeached, while “ukraine vacation” is irrelevant to United States politics as a whole. By incorporating bigrams, more precise and accurate features can be created, at the downside of greatly increasing the feature space. Testing our model with the addition of bigrams, we found that accuracy increased to 79.4%, a modest increase from our model with only single-word features.

Another method of differently incorporating words found in the corpus is to use what is called a tf-idf transformation. Short for “term frequency–inverse document frequency,” tf-idf is a statistic that can be used to weight the word features based on how unique the words are overall within the entire corpus of documents, thus emphasizing terms that are unique to a few documents. To calculate the tf-idf for a document, one simply needs to multiply the $tf$ term, which is simply the number of times that term appears in that document, by the $idf$ term. The $idf$ term is calculated using the formula $idf(t) = \log \frac{n}{df(t)} + 1$, where $n$ is the total number of documents and $df(t)$ is the number of documents in which that term $t$ appears. This results in a factor that increases in inverse proportion to the number of documents the term appears in. Typically, in a final step, the entire tf-idf vector for a document is then normalized.

While tf-idf can be a helpful transformation in many different text classification settings, we found that it was not useful for our model. While certain use cases might greatly benefit from a model that heavily weights terms that are very unique, our model actually suffers. Relatively common words within our dataset such as “impeachment,” “democrat,” “republican,” and “president” are actually all quite predictive for our model and can heavily impact the classification of the tweet. In contrast, relatively unique words are unlikely to have a predictive power for our model, given that there are few words to describe U.S. politics or impeachment that aren’t used commonly in all texts of those kinds. When we tested our model with
the use of a tf-idf transformation, we found that it significantly lowered the accuracy of the model to 63.6%. As a result, we decided not to use the tf-idf transformation in our model.

In addition to examining the text of a tweet when labeling it, we considered using other tweet attributes, such as when it was tweeted and who it was tweeted by. While the date of a tweet might be relevant (there are relatively few tweets about impeachment before Nancy Pelosi’s official announcement of the inquiry on September 24, 2019), we thought it best to omit such a feature since the date of a tweet should not in reality have any real predictive power about the subject of the tweet and using it as a feature might lead to overfitting.

We considered adding the user’s identity, specifically their screen name, as a feature in order to help further improve our model. For example, it is much more likely that The New York Times is tweeting about impeachment than The Verge, a technology news website. Upon testing the addition of this feature, we found that it slightly improved the accuracy of the model to 79.6%. Given the weak improvement adding the screen name provided to our model, we weighed the pros and cons and decided to exclude it as a feature from our model. While it did yield some light predictive power, we again risked overfitting our model, since the publisher of a tweet does not exclude it from tweeting about a certain subject (given the carefully cultivated political sample we collected from). We note looking at the classification reports in Figures 21 and 22 that both precision and recall perform worse in the impeachment category when we add in the screen name feature. Given the importance that we place upon our impeachment classification, as it represents the bulk of our analysis, the slight gain in overall accuracy does not outweigh the loss of performance for our most important classification. As a result, we settled upon excluding any non-text features from our model and instead focused on the text itself.
The final feature we added to our model was one specifically designed to help our model recognize key trends useful for the text classification. We aimed to implement a measure that combined the “salience” and “entropy” metrics as discussed in our literature review – one that gathered keywords specific to one class that are also used at least somewhat often. First, we iterated through every single tweet, and counted which words or bigrams were used at least ten times. Then, we kept only the words and bigrams that are used in tweets of one classification much more than the others – specifically, if the term satisfied the following formula:

$$\max(n_c) \text{ for } c \in C > 0.8 \cdot \sum_{c \in C} n_c$$

where $n_c$ is the number of tweets containing that term with the classification $c$. Using this formula ensured that we created specific features for the presence of phrases that are very unique to a certain classification, and thus are predictive for that classification. This process added a total of 213 new features to our model for the presence of certain words or phrases such as “[impeachment inquiry witness Gordon] sondland,” “whistleblower complaint,” and “jeffrey epstein.” While in theory the model could learn these trends on its own, giving it these specific features improves performance by a non-insignificant amount to an accuracy of 79.8%.

### 5.1.5 Data Filtering

While all of these changes helped improve the model’s performance, we still aimed to improve the model further. One fundamental problem was the fact that some tweets simply don’t have enough data for even an informed human to make a classification.
decision. Many tweets contain no words at all, and only an image or a video that can’t be parsed by our model. Other tweets are extremely short, or provide a short comment on another tweet through Twitter’s quote tweet feature. Despite quote tweets being automatically embedded on Twitter’s website and mobile app, the text and data from the quoted tweet are not included in the data returned by the API, making classification far more difficult. Therefore, for any tweet with less than 10 words, we decided to simply refuse to classify it and instead ignore it in our sample. While we would ideally choose to classify every tweet, there is no feasible method to classify these short tweets, and removing them eliminates less than 15% of our sample. While it is possible that there is some correlation between a tweet’s classification and its length, this is highly unlikely and as such performing this sample pruning should not affect our results. After pruning tweets with less than 10 words, we find that we obtain a significantly improved final accuracy of 80.9%, with results as listed in Figure 23.

5.1.6 Hyperparameter Tuning

Now, after exhausting all possible features and data pruning as part of our training process, we begin the final part of training our SGD model – tuning hyperparameters. Hyperparameters are essentially “settings” of the model that can sometimes affect the success of the model on unseen data – features such as the loss function, regularization penalty, and learning rate in the case of our SGD model. In addition to being set manually, hyperparameters are often set experimentally with a search
\[
\begin{align*}
\text{alpha} &= 0.001 \\
\text{class_weight} &= \text{balanced} \\
\epsilon &= 0.01 \\
\eta\theta &= 0.01 \\
\text{learning_rate} &= \text{constant} \\
\text{loss} &= \text{log} \\
\text{max_iter} &= 6 \\
\text{n_iter_no_change} &= 5 \\
\text{penalty} &= 12
\end{align*}
\]

Figure 24: Optimal Hyperparameters For Final SGD Model

process through the space of possible hyperparameters. Using a technique called grid search, we can exhaust all reasonable settings of the hyperparameters to find those that optimize the performance of our model.

Additionally, we use \(K\)-fold cross validation to ensure that our model generalizes well to new data and that we are not simply training our model to work well on our training data. With \(K\)-fold cross validation, we train our model \(K\) times, each time training the model on \(\frac{K-1}{K}\) of the training data and evaluating on the other \(\frac{1}{K}\) of the training data. We then average the results of the model across all \(K\) iterations, thus ensuring that we evaluate our model based on its generalization across all of the training data. We then finally pick the best model, train it on all of the training data, and evaluate it on the test data to obtain a final score for the model.

Using a typical value of 5 for \(K\), we trained on a set of 177,480 combinations of different hyperparameters for our SGD model. After that initial grid search, we examined that optimal set of parameters, then trained on an additional 32,256 combinations of hyperparameters in order to further optimize the model. After that second round of training, we found that the best hyperparameters were as listed in Figure 24. Training this model on all of the training data and evaluating on the test data, we obtain a final accuracy of 82.15% – over a percent better than our unoptimized model. We also find, as shown in Figure 25, that we have a fairly balanced model in terms of precision and recall across each classification and in
particular have fairly good precision and recall for the all-important impeachment classification. After confirming the accuracy of the model using the train/test split, we then trained the model on all of the labeled data that we have. It is this model that we then used to label all 1,260,226 tweets.

5.2 Results

We begin this subsection by discussing the makeup of our fully labeled sample, and analyzing trends in how the subject of tweets changed over time as the impeachment inquiry progressed. We then follow by analyzing specifically how the behavior of politicians differs based on three key variables – the partisanship of the politician, the partisanship of the politician’s district, and the difference between the two. By doing so, we aim to understand what key factors determine a politician’s willingness to tweet about impeachment. We find that all of these factors have a significant impact on a politician’s behavior, but that the effect is most pronounced in politicians that have a significant difference in political leaning compared to the district that they represent.

We observe a roughly similar makeup in the full corpus as in our manually labeled sample, as shown in Figure 26. We do note a relative increase in “Not Related” tweets and a decrease in “Politics” tweets. We hypothesize that this is due to the makeup of the labeled sample. We labeled 50 tweets from each day, while the total
number of tweets per day increased as time went on. Given the two major American holidays in November and December as well as the holiday shopping season, it is not unreasonable to believe that there were relatively more “Not Related” tweets during the later months that drove up the total number of tweets in the corpus.

Observing the number of tweets with a certain classification over time in Figure 27, we see several spikes in tweets about impeachment in correlation with specific events. Prior to the news reports about the whistleblower complaint on September 8, there are very few tweets about impeachment at all. There is a large spike in tweets in connection with the announcement of the official impeachment inquiry, the high-profile testimony of individuals such as Alexander Vindman and Gordon Sondland, as well as the final impeachment vote on December 18. We also note dips in the amount of tweets during holidays such as Thanksgiving and Christmas.
5.2.1 Engagement

In Figure 28, we plot the average number of retweets and replies per tweet, depending on the tweet’s classification. We find that tweets about politics receive significantly more engagement than tweets about unrelated subjects, and tweets about impeachment receive significantly more engagement than both other categories. Running a t-test against our null hypothesis that the mean engagement of politics tweets and impeachment tweets are equal, we find that we reject the null hypothesis for replies with a $p$-value of $9.15e-49$, and that we reject the null hypothesis for retweets with a $p$-value of $3.72e-155$. Tweets about impeachment clearly receive much more engagement than tweets about politics as a whole.
5.2.2 Political Party

We can also examine the difference in Twitter behavior by political party. In Figure 29, we plot the percent of tweets about impeachment tweeted by each political party. As the figure shows, Democrats tweeted more about impeachment during the run-up and initial days of the inquiry. As the inquiry settled in, Republicans began tweeting more about impeachment, perhaps as an effort to do damage control. Republicans and Democrats briefly swap places when some particularly damning transcripts of closed-door depositions were released on November 4th and 5th, however that trend quickly reversed and Republicans out-tweeted Democrats considerably for the rest of the year, including on the days surrounding the final impeachment vote.

However, the same could not be said for engagement with these impeachment-related tweets as shown in Figure 30. Save for a brief period immediately following the announcement of the impeachment inquiry, Republicans had both more retweets and
replies for the entirety of the impeachment process in the House. This is despite the fact that this analysis is only of members of Congress and does not include the President or Vice-President’s Twitter accounts. It is noteworthy, however, that immediately preceding and following the impeachment process, Democrats have periods of much more relative engagement compared to Republicans. It is possible that this is simply due to increased activity during the impeachment trial by Trump supporters defending the President. However, the precisely timed nature of the upswing and downswing suggest the possibility of a concerted effort of activity supporting Republicans during the impeachment process, whether by bots, paid actors, or both. The perspicuousness of this trend warrants further research, and could yield interesting results.
5.2.3  Ideological Polarity

While the political party of a certain member of Congress can yield interesting insights into how they tweet, we aim for a deeper analysis than just a simple binary. NOMINATE is an scoring process designed by political scientists Keith T. Poole and Howard Rosenthal for analyzing political ideology through legislative behavior. The most commonly used version, DW-NOMINATE [dynamic weighted NOMINATE] is extremely popular among social scientists for all kinds of political analysis. The DW-NOMINATE score ranges from -1.0 to 1.0 on two dimensions: liberal/conservative on economic matters, and liberal/conservative on social issues. However, most analyses focus on the first dimension, as it explains most of the variance in voting patterns. We too will focus on the first dimension for our analysis, examining how a congressperson’s DW-NOMINATE score affects how often they tweet about impeachment.

To match each person’s DW-NOMINATE score with their Twitter handle(s), we first gathered the DW-NOMINATE data from VoteView. [39] We then also gathered a list of all U.S. politicians’ Twitter accounts from ProPublica. [40] With these two datasets, we first gathered only the Twitter accounts from our House and Senate Twitter lists, then matched those accounts with each congressmember’s full name using the ProPublica dataset. Once we had each congressmember’s full name, we used that to match their Twitter handle with their DW-NOMINATE score. It is worth noting that congressmembers often have multiple accounts such as an official account and personal account, and both accounts are assigned the same DW-NOMINATE score, even though official accounts usually are less partisan. With the Twitter accounts now fully associated with each congressmember’s DW-NOMINATE score, we can begin to examine the data.

In Figure 31, we plot how often a congressperson tweets about impeachment given their DW-NOMINATE score. Specifically, we plot what percentage of their tweets
are classified as on the subject of impeachment on the y-axis. On the x-axis, we plot the most liberal congresspeople on the left, and the most conservative on the right, according to their DW-NOMINATE score. We filter the points on the graph to include only those who have tweeted about impeachment at least 10 times over the entire sample period, so that we only include those who have a large enough sample to be representative. Initially, it is hard to discern many trends from the graph. There are clear clusters on the left and right sides, representing the intense partisanship of modern-day America, but most congresspeople do not seem to tweet about impeachment extremely often. However, many of these accounts are official accounts which are typically more apolitical than the personal accounts that most congresspeople tweet from when they wish to express an opinion. In addition, some congresspeople may simply prefer not to express their opinion about impeachment on Twitter, instead preferring to use television or their own voting record to express their views.

To combat the effects of these accounts, we took a different tact of looking at the data. Instead of specifically looking at how one’s DW-NOMINATE score affects how much they tweet about impeachment, we looked at the effect one’s DW-NOMINATE score
has on the maximum amount they would be willing to tweet about impeachment. Some people may have the political capital to tweet about impeachment but choose not to, but anyone without the political capital to tweet about impeachment without negative ramifications must choose to refrain from tweeting about the subject. To examine this effect, we take a running maximum of data points from the center outwards, looking at the theoretical maximum that any candidate at or exceeding that polarity would be willing to tweet. Specifically, we only include datapoints that are greater than the previous maximum datapoint encountered, starting from the center (0.0) and running an outward pass in each direction. The results of this filtering of the datapoints is plotted in Figure 32.

Here, a much clearer trend appears. There are clear lines extending up and outward for each partisan direction. It is worth noting that by the simple nature of taking the running maximum, any trend would have to be positive as we move away from the center. However it is the number of datapoints, the clarity of the trend, and the slope of the line that lend evidence to support our hypothesis. Running a linear regression on DW-NOMINATE and $\text{abs}(\text{DW-NOMINATE})$, we find clear evidence that supports the existence of this trend, as shown by the coefficients in Figure 33.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Estimate</th>
<th>Std. Error</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-28.561</td>
<td>4.509</td>
<td>0.000135</td>
</tr>
<tr>
<td>DW-NOMINATE</td>
<td>-22.201</td>
<td>6.406</td>
<td>0.007099</td>
</tr>
<tr>
<td>(\text{abs}(\text{DW-NOMINATE}))</td>
<td>215.680</td>
<td>15.398</td>
<td>2.04e-7</td>
</tr>
</tbody>
</table>

**Figure 33: Linear Regression Results, By DW-NOMINATE Score**

For the DW-NOMINATE variable, we find a statistically significant coefficient at \(p < 0.01\), and our \(\text{abs}(\text{DW-NOMINATE})\) coefficient is significant at the \(p < 0.000001\) level. Not only that, but our coefficients are quite large. For every 0.1 increase in the absolute value of one’s DW-NOMINATE score, we expect an increase in their willingness to tweet about impeachment of 21.6%. Additionally, we expect that for any two given congresspeople with the same absolute DW-NOMINATE score, the liberal congressperson would be more willing to tweet about impeachment than the conservative one, the degree of which would correspond to their polarity from center. This regression has an adjusted \(R^2\) of 0.932, showing that it explains a high amount of the variance. As shown in Figure 34, this regression does run into issues as the DW-NOMINATE score approaches zero. This is due to the fact that there are quite simply no congresspeople with an absolute DW-NOMINATE score < 0.15, and no conservatives with a DW-NOMINATE score < 0.215. If these hypothetical congresspeople existed, a \((\text{DW-NOMINATE})^2\) or \(e^{-\text{abs}(\text{DW-NOMINATE})}\) term might be useful, however our model cannot fit these terms if there are no congresspeople to fit them on. As a result, we claim that the fact that our model returns negative percentages for certain scores does not disqualify the model but instead is simply a result of the current polarization of Congress. If necessary, we could clamp the percentages to the range of \([0, 100]\) to ensure that they stay within bounds.

The coefficients obtained from this linear model highly suggest that the amount that a given congressperson is willing to tweet about impeachment depends on their level of ideological partisanship. Not only are the results statistically significant, but their magnitude is sufficiently large so as to represent a large difference in the willingness of a congressperson to tweet about impeachment based on their DW-NOMINATE
score. A congressperson with a DW-NOMINATE score of 0.2 is only willing to tweet about impeachment at most 11% of the time, while a congressperson with a DW-NOMINATE score of 0.5 is willing to tweet about impeachment more than 68% of the time. This suggests that less partisan congresspeople are less willing to tweet about impeachment, whether due to personal politics or the politics of their constituents. In the next section we will examine this further, studying how the politics of a representative’s constituents leads to differences in behavior.

### 5.2.4 District Partisanship

While the partisan lean of a congressperson’s legislative votes can often be correlated with the politics of their constituents, there is still some room for variation between the two, and as such the legislative votes of a congressperson should not be used as a proxy for political pressure from constituents. Instead, we will use the Cook Partisan Voting Index, often abbreviated as PVI. [38] The Cook PVI measures how each House district performs at the presidential level compared to the nation as a whole. For example, in 2016, Hillary Clinton had 2.1% more votes than Donald Trump nationwide. If a certain House district voted for Clinton by only 1 percent,
that district would have a PVI of Trump +1.1%. If the district had instead voted for Clinton by 3 percent, it would have a PVI of Clinton +0.9%. Cook also has metrics from the 2012 election of Obama vs. Romney that they incorporate into their PVI calculations, taking the average of the 2016 and 2012 results. Finally, they round results to the nearest whole percentage point. Since the PVI dataset is limited to House districts, we will look specifically at members of the House of Representatives in this analysis.

In Figure 35, we plot how often a representative tweets about impeachment given their district’s Cook PVI score. As before, we plot the most Democratic districts on the left and the most Republican on the right. We again filter the points on the graph to include only those who have tweeted about impeachment at least 10 times over the entire sample period, so that we only include those who have a large enough sample to be representative. Unlike the DW-NOMINATE graph, this graph is much less clustered. While there are still clear Democratic and Republican clusters, there are at least a few swing districts with a PVI at or near 0, whereas before there were no congresspeople with a DW-NOMINATE score of absolute value less than 0.15. As before, we will take a running maximum of data points from the center outwards,
Figure 36: Running Maximum of Tweets About Impeachment, By Cook PVI

looking at the theoretical maximum that any candidate at or exceeding that PVI would be willing to tweet. We again take this running maximum starting from the center of 0 and running an outward pass in each direction. The results of this filtering are plotted in Figure 36.

A much clearer trend appears in this graph, albeit different from that in the DW-NOMINATE graph. There are still clear lines extending up and outward for each partisan direction, however the nature and slope of the lines vary greatly by party. On the Republican side, the trend quickly grows upward before leveling off at a fairly low PVI of approximately +15. In contrast, on the Democratic side, the running maximum increases slowly until about -5, at which point it begins to rise like the Republican side, yet still at a slower pace. This is despite the fact that a “zero” PVI represents a district that voted for the Democratic candidate +3% on average over the past two elections. It is quite astounding that a district with a PVI of R+1 (and thus voted for the Democratic candidate by 2% on average) has a maximum tweet rate about impeachment of over 27%, yet no district under a PVI of D+3 (and presidential vote of D+6) will tweet about impeachment more than 8% of the time. To confirm our visual observations statistically, we ran two different linear regressions. First, we ran
the same linear regression as we did for the DW-NOMINATE score, using the Cook PVI as a variable and $\text{abs}(\text{Cook PVI})$ as our second variable. Then, we ran a second linear regression replacing the $\text{abs}(\text{Cook PVI})$ variable with $\sqrt{\text{abs}(\text{Cook PVI})}$, in order to try to capture the diminishing returns as district partisanship increases. The results of the first regression, with an adjusted $R^2$ of 0.817, are listed in Figure 37, while the results of the second regression, with an adjusted $R^2$ of 0.870, are listed in Figure 38.

Given the significantly better adjusted $R^2$ value we obtained from the second regression, as well as the more significant p-value for $\sqrt{\text{abs}(\text{Cook PVI})}$ compared to $\text{abs}(\text{Cook PVI})$, we chose to base our analysis on the second regression that utilized the square root. For the Cook PVI variable, we find a statistically significant coefficient with $p < 0.01$, and our $\sqrt{\text{abs}(\text{Cook PVI})}$ coefficient is significant at the $p < 0.000001$ level. The magnitude of our coefficients are again quite significant – for every increase in the square root of the absolute value of a district’s PVI by one, we expect an increase in their representative’s willingness to tweet about impeachment of 17.2%. Additionally, we expect that for any two given representatives with the same absolute PVI score, the Republican representative would be more willing to tweet about impeachment than the Democratic one, an interesting reversal of the trend in the DW-NOMINATE regression. As shown in Figure 39, this regression performs fairly well, albeit not as well as the DW-NOMINATE regression. It fails to capture the relatively low willingness to tweet about impeachment in the
slightly Democratic PVI districts, and the corresponding high willingness to tweet about impeachment in the slightly Republican PVI districts. This difference is quite interesting, and lends credence to the commonly-held belief among pundits that moderate Democrats are afraid of looking partisan, while moderate Republicans have no qualms about tweeting in regards to partisan subjects. These results suggest an asymmetric polarization among political parties, where Republicans are more willing to tweet about politically toxic subjects than Democrats at the same partisan lean. Such a topic deserves further research in the future to see if it manifests itself in other areas on Twitter. We will continue to look at partisan divides in the next section, focusing specifically on representatives whose party affiliation does not match the partisan lean of their district – so-called “crossover districts.”

5.2.5 Crossover Districts

Perhaps the most obvious effect that partisanship has on Twitter behavior appears in so-called “crossover districts.” These are districts in which the district’s presidential party preference does not match the party of the representative that the district elected to the House. We aim to examine the difference in how candidates from
districts with similar PVI tweet depending on whether they are in a crossover district or match the party lean of the district to which they were elected. Since the United States population has voted for the Democratic candidate by 3 percent, on average, over the past two elections, this will be our boundary line for determining whether a representative is in a crossover district. Any Republican representative in a district with a PVI less than 3 or any Democratic representative in a district with a PVI greater than 3 will be defined as a crossover district. Any district with a PVI of exactly 3 is by definition not a crossover district due to its even partisan lean.

In Figure 40, we plot the percent of tweets about impeachment for representatives from all districts with a PVI between -3 and 13, inclusive. These represent the most partisan crossover districts – John Katko (R) in NY-24, and Ben McAdams (D) in UT-4. Crossover districts are plotted in orange, and non-crossover districts in blue. The vertical red dashed line represents the partisan boundary – any crossover districts to the left of the line are represented by Republicans, and any crossover districts to the right are represented by Democrats. Crucially, we adjusted our filtering step for this process – now we only filter out any representatives who have never tweeted about impeachment. Any representative who has tweeted about impeachment at
least once and has tweeted at least a total of ten times is included on the graph. We lowered the threshold for this filter because representatives in crossover districts tweet about impeachment so little that keeping the impeachment tweet threshold of 10 cut the number of crossover districts represented from 24 to one. Without relaxing our filter, essentially all of our data would have been removed because of the very trend we are trying to identify.

It is very clear from the graph that representatives in crossover districts tweet about impeachment much less often than their non-crossover brethren. On average, representatives in crossover districts tweet about impeachment less than three percent of the time, while those in non-crossover districts tweet about impeachment more than ten percent of the time. This trend becomes even clearer in Figure 41, where we plot the average for each PVI, broken into crossover and non-crossover districts. In this graph, the mean for every PVI is lower for crossover districts than non-crossover districts, and is significantly lower except for one outlier at the PVI of R+8. The differences become more extreme at the outer PVI edges, where the average representative in a regular district with a PVI of R+13 tweets about impeachment
more than 25 percent of the time yet a representative in a crossover district with that PVI tweets about impeachment less than 1.3 percent of the time.

Running a t-test between the two groups of representatives, with a null hypothesis that the mean of crossover districts is equal to the mean of non-crossover districts, we obtain a p-value of 0.001456, significant at the $\alpha = 0.01$ significance level. This strongly suggests that representatives in crossover districts choose to tweet about impeachment less than those in non-crossover districts. We also ran t-tests between groups of representatives in specific PVI ranges, in order to determine if there is a statistically significant difference between representatives with similar PVIs. While we wished to run these t-tests at each individual PVI, there are so few crossover districts that our sample size was so small as to be infeasible. We instead binned representatives at similar PVIs together according to the following bins: $\{[-3, 1]; (1, 4]; (4, 6]; (7, 13]\}$. Our results are listed in Figure 42. $n_c$ represents the number of crossover districts in the sample, and $n_d$ represents the number of regular districts in the sample.

As the results show, there is some evidence for changes in tweeting behavior based on crossover districts, especially in the $(4, 6]$ bin. Unfortunately, the number of crossover districts is simply too small to give statistically significant results at the PVI level, and the $(4, 6]$ bin is the only statistically significant result at the $\alpha = 0.05$ significance level. The other bins do have fairly negative test statistics, but the p-values are not strong enough to be statistically significant. Regardless, this is still strong evidence that supports our hypothesis that representatives in crossover districts limit their tweeting about impeachment compared to their non-crossover

<table>
<thead>
<tr>
<th>Bin</th>
<th>$n_c$</th>
<th>$n_d$</th>
<th>Statistic</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[-3, 1]$</td>
<td>3</td>
<td>38</td>
<td>-1.35</td>
<td>0.177</td>
</tr>
<tr>
<td>$(1, 4]$</td>
<td>8</td>
<td>37</td>
<td>-1.49</td>
<td>0.138</td>
</tr>
<tr>
<td>$(4, 6]$</td>
<td>8</td>
<td>19</td>
<td>-2.11</td>
<td>0.036</td>
</tr>
<tr>
<td>$(7, 13]$</td>
<td>5</td>
<td>91</td>
<td>-1.48</td>
<td>0.140</td>
</tr>
</tbody>
</table>

Figure 42: Binned T-test of Crossover vs. Non-Crossover Districts, By Cook PVI
counterparts, and is more of an indictment of the small number of crossover districts than anything else.

Overall, these results show strong evidence to support the hypothesis that members of Congress change their tweeting behavior both based on their personal ideology and the ideology of their district. We find statistically significant evidence that suggests that members of Congress with more partisan DW-NOMINATE scores are more willing to tweet about impeachment, and that members of Congress in more partisan districts are also more willing to tweet about impeachment. We find the clearest evidence of this trend when comparing using crossover districts, finding that representatives in crossover districts are much less willing to tweet about impeachment than those in non-crossover districts. Additionally we find that, even with a low sample size, there is significant evidence to support this trend even in districts with the exact same partisan lean. Overall, these results yield conclusive proof that political leaning, and the lack thereof, greatly affects the behavior of politicians on Twitter when tweeting about partisan issues such as impeachment.
6 Predicting Results in the Democratic Primary

In this section, we attempt to flip the relation and examine how Twitter can be used to predict political outcomes instead of how politics affects Twitter. First, we examine the utility of replies on Twitter, and whether they provide any meaningful predictive power. Then, we methodically gather features to use as inputs for our models that attempt to predict outcomes in the Democratic primary using Twitter.

With our features gathered, we first briefly examine changes in popularity among candidates. We then attempt to build a model to use Twitter data to predict trends in Democratic primary polling. We follow by focusing on the more narrow domain of Democratic debates, specifically predicting polls of debate performance using Twitter data from during the debates. We find limited results when attempting to predict Democratic primary polling, but meaningful success when focusing on the narrower, more politically active domain of debate watchers, suggesting that Twitter is a much more effective predictor when the target population’s demographics more closely match the demographics of Twitter’s user base.

6.1 Process

6.1.1 The Utility of Replies

In addition to tweets from our list of politically relevant accounts, our political sample collects retweets of and replies to those accounts as well. Both the numbers of retweets and replies, as well as the content of the replies themselves, can serve as valuable data sources for estimating the popularity of both specific tweets and the candidates themselves. For our research into the Democratic primary, we considered the large corpus of replies as a promising avenue to examine when determining the popularity of certain candidates.
However, upon examining the set of replies, we determined that while the number of replies as a total might be insightful, the replies themselves were often impossible to parse, even by a human. Even when the replies could be classified with a positive or negative sentiment, we found that replies were overwhelmingly negative, and did not seem to focus much on the content of the tweet or the candidate themselves – not varying much from tweet to tweet, or candidate to candidate. In Figure 43, an example of a typical reply is reproduced. This reply is vague, insults the candidate in a bizarre way, has poor capitalization and grammar, and is all-around just weird. While this tweet is certainly negative, it is bizarrely so and does not seem to be related to any real political event or intent to vote.

Even charitably-deemed “nice” replies are few and far between. In Figure 44, a tweet that could be construed as nice (and was classified as such for the purposes of our analysis) is reproduced. While the tweet is at least not blatantly aggressive or insulting, this tweet provides no indication of support and instead simply asks a question, which, depending on the answer, might significantly affect the voter’s propensity to vote for the candidate. As a result, while classifying the tweet as
positive is valid (at least in the context of Twitter), it can’t be sincerely interpreted as a measure of support.

Lastly, there are tweets that are simply too difficult to parse even by a human, much less a machine learning classifier. Consider the tweet in Figure 45, a reply to Andrew Yang’s support of a petition to have popular podcast host Joe Rogan moderate a Democratic debate. This tweet has no discernible opinion towards Yang, and only a slight hint of sarcasm or mockery towards Rogan. This tweet lends no insight on this person’s support for Yang, or any other presidential candidate for that matter. As such, the sentiment is unknown and can’t really be quantified at all. In addition, there are some tweets that have a clear sentiment, but are unrelated to the candidate, such as tweets expressing support for protesters in Hong Kong and the like.

After manually classifying a small sample of 140 randomly selected replies to presidential candidates, we found the following breakdown of reply sentiment, as shown in Figure 46. As the graph shows, 57% of replies are classified as negative. In contrast, only 16% of replies are classified as positive – and this includes replies like those in Figure 44 that are more neutral than positive, to be precise. In fact,
more replies are classified as incomprehensible – unable to be parsed to the best of our ability – than positive. Within our small test dataset, negative replies outnumber positive replies by more than 3 to 1.

Given the results from our test sample, we found it to be a fruitless exercise to attempt to build a reliable classifier for the sentiment of these replies. Not only are the replies hard to parse even for a human, but the replies are such that they provide very little information about actual support for the candidate. Other metrics such as retweets and follower numbers pose a more credible signal for analyzing engagement with candidates, since the relatively large number of these interactions cannot be easily swayed by bots or highly engaged users, unlike replies.
We also note that users reply much less than they retweet. In our entire political sample, the median number of retweets per user is one, but the median number of replies per user is zero. This means that over 50% of users – who already showed engagement by retweeting someone on our list – never replied to any of the other 2,575 accounts for any time in a six month period. The mean number of retweets per user is a quite large 18.97, yet the mean number of replies is only 8.82. Lastly, among users who have both retweeted and replied at least once, the average number of retweets divided by replies is 10.97, that is, for every one time a user replies, they also retweet someone 11 times, on average. This suggests that even among engaged Twitter users, replies are still relatively rare.

For all of these reasons, we instead simply view replies as a corollary for negative sentiment, since the relatively high opportunity cost of engagement lends itself to upset or otherwise concerned users. More importantly, by choosing to use only the number of replies as a feature, instead of some arbitrary sentiment classification, we allow our model to correctly correlate replies with any possible influence in the polls.

### 6.1.2 Gathering Features

After examining the feasibility of classifying replies, we set about gathering other features to test the predictive power of our Twitter dataset, specifically looking at the 2020 Democratic primary. Using the over one billion tweets gathered from both our political and random samples, we aimed to generate useful features for analysis and model training. We demarcate these features by each hour, thus giving us the ability to analyze trends on an hour-by-hour basis.

We started with basic features such as retweets, replies, and follower numbers. Using our political sample, we gathered the number of retweets and replies of each candidate by hour, as well as how many followers they had at that time. We then moved onto more advanced features, starting with what we call “new engagement.”
While retweets and replies are useful metrics, often these retweets and replies can be filled with highly engaged users and bots that reply to or retweet almost every single tweet from a candidate. To counteract this, we created a new metric that measures how many times a user retweets or replies to a candidate for the first time in history. We do this by iterating through our political sample in chronological order, and only counting a retweet if they have never retweeted that candidate before, following the same method for replies. Since we gather a complete sample of all interactions with these candidates dating back to August 6th, we can rest assured that our calculated figures are quite close to the actual numbers once our date of interest is sufficiently past the date when we commenced data gathering. Our justification for this is that any user who is retweeting or replying with such frequency as to skew the data will almost certainly have retweeted or replied to our candidate within the 18 day window between the start of data gathering (August 6) and the start of our gap in data (August 23rd). Then, from the end of our data gap (August 27) onwards, we can assume that our collected figures are close to accurate. These are the main features that we gathered from our political sample.

In addition to these main features, we gathered some additional features to see if they yielded any predictive value, although we did not suspect that they would. Despite our investigation and determination that replies yield very little sentiment or useful animation, we also gathered the sentiment of all replies by hour using the TextBlob sentiment classifier. [34] TextBlob is a Python text processing library, that, using a built-in analyzer based on the “pattern” library [35], can report a text’s “polarity” and “subjectivity” on a scale from -1.0 to 1.0. For our purposes, we simply took the average of the polarity of every reply to a candidate and cataloged it by the hour the reply was posted. We also used this same process to record the sentiment of every tweet posted by the candidate themselves, in addition to tracking each candidate’s number of tweets per hour. We lastly gathered the words used in replies to each candidate, again demarcated by hours. We stored this in a generic “bag-of-words”
format, keeping track of the number of times each word was used in a reply to each candidate.

We also gathered many features from our random sample. While the political sample yields great insight on the political sphere itself, the random sample contains valuable data on how popular a candidate is in the world at large. Filtering for tweets that are English-language only, we gathered the number of tweets that mentioned a candidate, again by hour. We gathered this metric in two ways: first we gathered the number of tweets that mentioned the candidate’s specific Twitter handle, such as ‘@ewarren’ in the case of Elizabeth Warren. We then also gathered the number of tweets that mentioned the candidate by any of their names or nicknames that we could be reasonably confident were a mention of the candidate themselves. In the case of Pete Buttigieg, we counted any tweet that had either “Buttigieg” or both “Mayor” and “Pete.” However, for a candidate like Joe Biden, we only counted tweets that contained “Biden,” since “Joe” is a common American first name. A full list of how we identified each candidate in the tweets is available at Appendix F. For each method of counting mentions, we separately counted both the number of tweets that mentioned each candidate, and the average TextBlob sentiment of those mentions, again separating by hour. This concludes the features that we gathered for the Democratic primary model.

6.2 Results

In this subsection, we analyze the effectiveness of Twitter as a predictor for the Democratic primary. We focus on two areas – national primary polling, and surveys of candidates’ debate performance as rated by those who watched the debate. We focus on these two areas to compare and contrast the effectiveness of Twitter as a predictor for an entire national population versus a subsection of the population that is more politically active and therefore more likely to post their opinions on
Twitter. We find much greater success when focusing on the narrower domain of debate polling as opposed to national primary polls.

6.2.1 Primary Polling – Temporal Split

Throughout our analysis, we limit the candidates we track to the top five in the primary process that had declared their campaigns prior to the start of data collection: Amy Klobuchar, Bernie Sanders, Elizabeth Warren, Joe Biden, and Pete Buttigieg. As our target dataset, we picked what is the de facto gold standard for primary polling – FiveThirtyEight’s presidential primary polling average. [36] This average takes into account every respected national poll, weighting each poll for house effects and other factors associated with the polling firm and sampling method. It is widely considered to be a better method than simply taking an average of all recent polls, and certainly better than looking at each poll as it is released. However, even the FiveThirtyEight average will often jump over half a percentage point as new polls are incorporated into the model. Under our hypothesis that Twitter data could be used to predict primary polling, the Twitter data representing candidate support is continuous – it should not spike on certain days as a new poll is released. Instead, it should ebb and flow gradually as the amount of support for different candidates changes. For this reason, we instead used a moving average of the FiveThirtyEight data as our target variable to predict. Specifically, we took a moving average of the past three days, the current day, and the future three days (totaling a week). We did this since polls are typically conducted over at least a three day span and then released and incorporated into FiveThirtyEight’s model. For this reason, incorporating future data into the moving average should not affect the temporal accuracy of the average, since that “future” data actually comes from polls conducted in the past.

Applying this moving average smoothed out the FiveThirtyEight data considerably, and with that done we began work on attempting to evaluate Twitter’s effectiveness as a predictor. We attempted to do this with two different models: a time-series
model that had access to the previous day’s polling data, and a generalized model that only had access to the Twitter data specifically. A time series model is a model that uses a previous known “ground truth” result from the current day, in addition to other input data, to attempt to predict the next day’s result. In mathematical terms, it uses the Twitter data from Day $n$, $x_n$, and the polling result from Day $n-1$, $y_{n-1}$, in an attempt to predict the next day’s polling result, $\hat{y}_n$. Time series models have the benefit that they can correct from past mistakes. Even if our prediction $\hat{y}_n$ is incorrect, we can use the true value $y_n$ along with $x_{n+1}$ to predict $\hat{y}_{n+1}$. We can also, if we are confident enough in our model, use the $\hat{y}$ predictions as future input and predict the model far into the future without having access to the previous day’s ground truth data, subject to the obvious caveat that the model will be less accurate as $n$ increases. The alternate kind of model is a more generalized model that solely uses the Twitter data to make a projection of what the polling average would be. We trained these models using data gathered during the months of September, October, and November, and then used these models to attempt to predict the results during December, thus splitting our data temporally.

We tested both kinds of models in our attempts to use Twitter as a predictor, training specific models for each candidate since the signals from the Twitter data will likely affect different candidates in different ways. Despite training both sets of models on the same set of data, we must evaluate them very differently. While the generalized model has some value if it is close to accurate, the time series model has an extremely high bar to meet to prove its worth. This is due to the fact that the time series model has access to the previous day’s polling information. A rudimentary model can simply predict the previous day’s polling information as today’s result and be remarkably close, since the polling average does not change much from day to day. As a result, any time series model we construct must be more accurate than simply predicting the previous day’s data.
In our attempt to predict primary polling, we not only used the features we gathered as described in Section 6.1.2, but also calculated additional features based on them. Mainly, we tested taking the difference of that day’s feature and the previous day, thus calculating the day-to-day change in followers, retweets, or whatever other variable we were testing. In addition, we also tried taking the 7-day moving average of variables, in an attempt to both neutralize ebbs and flows in user behavior due to the weekday/weekend cycle and smooth out any anomalies from particularly popular or unpopular tweets.

Despite testing all possible types of models and various subsets of features on each of the top 5 candidates, we failed to find any parameter set that worked well enough to not be considered noise. Our time series predictor was especially unavailing, failing to match the baseline of predicting the previous day’s data, and often faring far worse. A particularly interesting note is that in general adding more features to our time series model caused the model to perform worse. We hypothesize that the simpler models, such as those with only the previous day’s data and one or two other features, allowed the machine learning algorithm to pick up on the significance of the feature representing the previous day’s data. In contrast, the more complex feature sets likely confused the algorithm, caused it to overfit, and fail to accurately predict future data.

Our generalized model fared slightly better, at least within the context of what we deemed an acceptable accuracy given its lack of access to previous poll data. While we still fared relatively poorly, we found modest success in predicting Pete Buttigieg’s future polling based on his Twitter data using a Random Forest Regressor. As shown in Figure 47, our model, although noisy, not only follows but actually predicts Buttigieg’s fall in the polls over the month of December. Unfortunately, Buttigieg was the only candidate for which the model even meaningfully matched the results, so it is quite possible that this result is simply noise or just pure luck.
6.2.2 Primary Polling – Random Split

However, the previous model was trained with a specific set of data. We gave the model access to polling data from September, October, and November, and then used that model to attempt to predict the data in December. We chose this method of training the model in order to be able to accurately make claims on our model’s ability to predict the future, since it had no access to any of the future data. We believe choosing this strategy may have contributed to our model’s struggles to accurately predict polling averages based on Twitter data. Machine learning models work best when they are given data similar to the data they have been trained on – while they do excel (and thus are popular) at analyzing never-before-seen data, they do best when that data is still within the bounds of the data that they have been trained on. We believe that part of the model’s inability to predict the future trained solely on the past is that the input data from the future set goes out of bounds of the model’s training data. As the election gets closer and all candidates gain followers,
• Sentiment of tweets with handle mentions [random sample]
• Number of tweets with handle mentions [random sample]
• Sentiment of tweets with candidate references [random sample]
• Number of tweets with candidate references [random sample]
• Number of first-time retweets of a candidate [political sample]
• Number of first-time replies to a candidate [political sample]
• Difference in number of followers of a candidate compared to the previous day

Figure 48: Features Used in Primary Poll Model

the number of interactions for all candidates increase and thus may be unlike any data the model has ever seen before.

We can mitigate this problem using a so-called “train-test random split.” Instead of using two distinct sets of data from different time periods to train and test our model, we can randomly separate the data into train and test sets. This has the benefit of removing any temporal biases and ensuring the model sees the entire range of data that it might have missed in the temporal split. When using this random split, we found significantly improved results for the generalized model. However, these results have an obvious caveat that we are now technically using the future to predict the past. There is a possibility that our model might detect hidden trends within the data and simply average from the points it knows about, thus making a prediction not based on the test data itself, but the test data’s relationship to the training data that the model already knows about.

For this reason, when testing this method, we used only variables that we believed could be reasonably assumed to not be strongly correlated from day to day. These variables, listed in Figure 48, include the number of mentions of each candidate in the random sample, the difference in followers compared to the previous day, and other features that are not believed to strongly correlate. These features are unlike more traditional metrics for a candidate’s popularity such as total followers, which roughly follows a increasing, approximately linear trend. With this group of variables, we found surprisingly impressive results at predicting a poll rating given
past data. Using a Random Forest Regressor trained on the training subset of our data, we are able to obtain an $R^2$ of 0.694 when predicting on our test subset of data. Impressively, our mean squared error is approximately 0.87 points, meaning that on average we can predict any polling average within 0.94 points just based on Twitter data. In Figure 49, we show the predicted polling average values compared to the actual polling average values, again for Pete Buttigieg. It is important to note that this model actually uses a subset of features from the model shown in Figure 47 with the only difference being what subset of data we train the model on. In Figure 47, the model is trained on a continuous portion of data from September through November and then tested on December, while Figure 49 shows a model trained on a random 75% subset of data from the September-December period and tested on the 25% of data not included in that previous subset.

With these findings, we’ve shown that using Twitter data to predict polling results, at least in the crowded field of the Democratic primary, could at best be considered a mixed bag. Given the relatively small changes in polling data from day-to-day,
we’ve seen that a time series model is not an effective tool, and cannot approach the accuracy of simply predicting the previous day’s results. We found greater success with a generalized model that ignored previous polling and instead attempted to predict results based on Twitter data alone. With this model, we still ran into issues when predicting future data that went outside the bounds of the training data, and at best found one moderately successful result amongst a collection of failed results. We finally encountered success when switching to a randomized train-test split, with the caveat that using such a split raises questions about the efficacy of a model that predicts the past with access to future data. Despite using variables that can reasonably be considered to not strongly correlate from day-to-day, concerns exist that such a correlation is hidden in the data that our machine learning model used to successfully make accurate predictions. Our results raise questions about the feasibility of using Twitter data to predict society at large, while leaving open the possibility for future research to determine if the random split model can be modified to work without access to future data.

6.2.3 Debate Polling

After finding limited results when attempting to predict primary polling, we set about focusing on a narrower subject – debate polling. We chose to focus on this narrower field in part due to Johnson, P. Shukla, and S. Shukla’s findings that Twitter is a better source for gauging sentiment towards specific events than overall public sentiment towards a given politician. [9] In partnership with FiveThirtyEight, polling firm Ipsos has conducted both pre- and post-debate polls for each Democratic primary debate. [37] The polls do not simply measure the amount of support for a candidate before and after the debate, but ask specific questions about which candidate had the best performance, which candidate was the most likely to defeat President Trump in the 2020 election, and more. Most importantly, Ipsos polls the same group of people before and after the debate, and only tabulates results for those who answer
both surveys and indicate that they watched the debate. This ensures that the results represent the opinions of those who actually watched the debate, and eliminate some of the sampling error by polling the same group of people both times.

In addition, by focusing on the debate instead of the primary at large, we hope to capture a set of Democratic primary voters that are more active on Twitter. Those who watch debates even before the beginning of the election year are much more likely to be on Twitter and thus be in our dataset. According to the New York Times, 27% of Democrats who post political content on social media say they don’t follow the news much, as opposed to 59% of Democrats who don’t post political content on social media. [5] This suggests that those who follow the news and watch the debates are more likely to post on social media and thus be included in our Twitter sample. By focusing on debates, we aim to narrow down the set of voters we are trying to capture to those that are much more likely to be in our dataset.

For our analysis, we aim to predict a certain candidate’s debate performance in terms of the percentage of respondents who rate the performance as “good” minus the percentage who rate the performance as “poor.” There were a total of four Democratic debates during the period in which both our political and random samples were gathering data. While our random sample was gathering data for the first two Democratic debates prior to August 6th, these debates were held over a two-night period thus greatly changing the dynamics of the debate, and the polls were conducted using a different polling firm, so for this reason we chose to exclude those debates for our analysis. We chose to attempt to estimate scores for the top five primary candidates who participated in all four debates: Amy Klobuchar, Bernie Sanders, Elizabeth Warren, Joe Biden, and Pete Buttigieg. Looking at the debate scores listed in Figure 50, we note the high variance in scores from debate to debate among most candidates. We also note that, save for Amy Klobuchar, the debate
1. Sentiment of tweets with handle mentions [random sample]
2. Number of tweets with handle mentions [random sample]
3. Sentiment of tweets with candidate references [random sample]
4. Number of tweets with candidate references [random sample]
5. Number of first-time retweets of a candidate [political sample]
6. Number of first-time replies to a candidate [political sample]
7. Sentiment of replies to a candidate [political sample]
8. Number of replies to a candidate [political sample]
9. Number of retweets of a candidate [political sample]

scores are neither linear nor monotonic, that is, they both increase and decrease in different proportions from each debate to the next.

In building a predictive tool for these results, we had to be especially careful given the extremely small sample size of 4 debates. Unfortunately, as is often the case in social science (especially for an event that only occurs every 4 years, at minimum), there were no more debates available for us to test our predictive powers with. For that reason, we chose to use a very basic linear regression as our model for these predictions – otherwise, we could easily overfit the data and find a model that is not effective on new data. With only four datapoints, we are limited to a maximum of two features, lest we completely overfit our model. In order to reduce the possibility of correctly fitting our model due to random chance, we will limit ourselves to using the same two features per candidate. Otherwise, for each candidate we could pick one of the 36 different combinations of features and likely find one model that works simply due to luck. We are also benefited by the lack of monotonicity or linearity in the debate results, thus decreasing the likelihood that linear temporal trends like the number of followers, retweets, or likes are correlated with the debate results simply due to the date that the debate occurred on.
In Figure 51, we list the features used in testing our model. When working on this model, we tested all different 2-feature combinations of this set in order to determine which set of 2 features worked best across the different candidates. For each debate, we gathered the feature data from all tweets sent while the debate was live. Since most debates were three hours, but one was two, we divided each statistic by the number of hours in the debate in order to obtain a per-hour metric. It is this metric we used as an input to our linear regression. We tested all 36 (9 choose 2) combinations, and then evaluated how well they performed using $R^2$ as a metric. The results from the top five performing sets are shown in Figure 52.

The best performing feature set, by far, is (2, 4) which is the number of times a candidate’s Twitter handle is mentioned and the number of times a candidate’s name is mentioned. However, the results become even clearer once we switch to adjusted $R^2$ instead of $R^2$ as our evaluation metric of choice. While both statistics measure the descriptive power of models, the adjusted r-squared value compensates for the number of features and number of data points in the model. This metric is extremely useful for our model given its limited number of data points, and will help determine if our model is actually fitting the data well due to its predictive power or due to chance.

Looking at the results in Figure 53, we see that while the order of feature sets remains the same, the adjusted $R^2$ obviously drops. However, now the difference between the first feature set and the rest has widened dramatically. Of all 36 combinations, only 2 have an adjusted $R^2$ greater than 0. Although the mean adjusted $R^2$ of our best feature set does drop by approximately 0.15 compared to its regular $R^2$ value,
this is mainly due to its performance with Elizabeth Warren. Its value with Joe Biden drops slightly, and for the rest of the candidates the drop is negligible. This suggests that this model is actually quite predictive with these candidates and this feature set.

Examining the coefficients for each candidate in Figure 54, we note that, except for Warren, all candidates have positive coefficients for Handle Mentions (feature 2) and negative coefficients for Name Mentions (feature 4). It is likely no coincidence that it is also Warren for whom this model performs far worse than any other candidate.

Interpreting the coefficients, it seems that for four out of the five candidates, an increase in handle mentions suggests an increase in debate performance, while an increase in name mentions suggests a decrease in debate performance. This could be in part due to differences in Twitter behavior depending on one’s support of a candidate. If one supports a candidate, he or she might be more likely to know and/or use that candidate’s official Twitter handle in their tweets. In contrast, someone who is not a supporter may simply reference the candidate by name instead of mentioning their handle. If this assumption is correct, it would then suggest a correlation between the number of handle mentions and the amount of positive support for a candidate, as well as the number of name mentions and the amount of negative sentiment towards a candidate. Warren, for whatever reason, exhibits a different trend than the other four major candidates. It is possible that this is the case.
since her Twitter handle, ‘@ewarren,’ is not simply her first and last name, unlike the other four major candidates, and is thus used proportionally less often by her supporters. Further research into why Warren does not exhibit the trend shown by the other major candidates could prove quite informative.

In Figure 55, we display the results of our regression in a graphical format. The number of handle mentions per hour lies on the x-axis, and the number of name mentions per hour on the y-axis. The predicted polling result by our regression is displayed by color, clamped to the range of debate performances exhibited by all candidates. The four debate performances are marked by circles with a red outline, and the interior fill color of the circle represents the true debate performance. In some graphs this may not be clear; that is simply because the regression predicted the debate performance with a negligible amount of error. The difference between Warren and the rest of the candidates is abundantly clear in these graphs as the gradient switches direction in Warren’s graph. Also clear is the roughly linear correlation between number of handle mentions and number of name mentions in the dataset. At the same time, it is clear that both features are important, for example, consider the two data points in Sanders’ graph with the same number of name mentions but different numbers of handle mentions that clearly affect his performance. The similarity in graph appearance across four out of the five major candidates suggest the generality of the model and support the claim that this model would be useful in predicting future debate performances.

While the claims we can make from this analysis are limited due to the extremely small size of our dataset, the results are quite encouraging in suggesting that a random sample of Twitter can be a very useful tool in predicting debate performance for candidates. The model’s similar features for four out of five major candidates as well as its remarkable success compared to every other combination of features suggest that it is truly a predictive model and not simply luck. The existence of such
Figure 55: Linear Regression Graphical Results
a model poses to be a large boon for campaign teams, allowing them to adjust their strategy and message in real time as data comes in, instead of waiting several days for a poll to be conducted. It also allows smaller campaigns to better adjust their campaign message without paying for expensive and time-consuming polls. Using this data, campaigns could quickly decide whether to highlight or move on from a specific debate performance or talking point, and better fine tune their messages for voters they wish to target. The ability to use this kind of data in real time has great potential in the ever more data-driven field of campaign politics, and deserves more research.
7 Future Research & Conclusion

7.1 Results Summary

In this thesis, we have examined Twitter in-depth to study its relationship with contemporary politics. As a part of this examination, we’ve gathered over 6.5 terabytes of data representing 1,086,104,118 tweets. With this corpus of over one billion tweets, we’ve found trends in Twitter behavior as well as useful predictors for political outcomes.

We started by examining the feasibility of classifying the subject matter of tweets, given their limited length. We found success in building a classifier with over 82% accuracy despite the relatively extreme class distribution. We then used this classifier to label all of the political tweets we gathered. We first found that tweets about impeachment are significantly more popular than those about other subjects. Then we analyzed tweets sent by members of Congress, and analyzed how congresspeople use Twitter depending on their partisan lean. We found statistically significant evidence that more partisan politicians are more willing to tweet about impeachment, and that representatives from more partisan districts are similarly willing to tweet more about impeachment. We also found that, regardless of partisanship, representatives in districts that split the vote between the presidency and House are much less willing to tweet about impeachment than those from districts who did not.

After studying how politics affected Twitter behavior, we looked at how Twitter behavior can be used to predict political outcomes. First, we attempted to predict trends in Democratic primary polling using Twitter data. We found little success in attempting to predict these outcomes, although we found greater success when using a random train-test split, thus suggesting that part of our model’s shortcomings came from Twitter’s tendency to surpass the bounds of previously seen data. We found much more success when attempting to predict debate performance, a metric that
surveys a demographic much more likely to be active on Twitter. We found that with just two features we could make remarkably accurate predictions about a candidate’s debate performance, and that these same two features were high quality predictors across almost all candidates. While it is hard to generalize these findings given the limited sample size of four debates, they provide positive evidence that predicting such outcomes using Twitter is possible and could revolutionize data-driven politics in the future.

7.2 Future Research

Building upon the research conducted in this paper, future work could examine other various ways in which politicians mediate their Twitter behavior for partisan reasons. While impeachment was chosen as our metric for partisan behavior due to its relevance during our data collection period, other partisan issues such as universal healthcare, abortion, or gun control could be examined in future research as well. Further examination into the differences in behavior between political parties, both by their politicians and supporters, could yield interesting results, especially in the differences we’ve found between moderates of both parties. In addition, research could examine the behavior of presidential candidates, how behavior changes between the primary season and the general election, and how “moderate” versus “extreme” lanes are formed.

Even more useful future research could be performed to further our preliminary findings in attempting to predict Democratic primary outcomes. In particular, those with access to data from previous primary cycles could examine the effectiveness of our model on previous debates, and study the differences between primary debates in different political parties and primary years. Crucially, future research could verify that our findings hold among debates other than the four that we studied. Future research could also examine the differences we found in Senator Warren’s results and determine if there is any reasoning behind the differences in her model’s behavior.
There is much more to study in this area to determine whether Twitter is effective in predicting political outcomes, from a state House race to the Presidential general election. Our findings in this thesis lay a solid groundwork for future research to expand upon, and it is evident that Twitter lends at least some predictive power, especially in domains that match its demographics.

7.3 Conclusion

In conclusion, Twitter is clearly an effective tool for analyzing the field of politics. Twitter dominates the political social media landscape, and its discourse can have real effects on events outside its domain. Twitter behavior clearly varies based on the politics of each individual using it, despite the presumption that Twitter is by-and-large a place for the hyper-partisan. Not only does Twitter reflect real-life politics, but it can also serve as a predictive tool for public opinion, and provide useful feedback for politicians aiming to sharpen their message. In short, maybe Twitter actually is real life.
References

   https://twitter.com/ezraklein/status/1224529115107082240.

   https://twitter.com/cwarzel/status/1230222511624744960.

   https://www.statista.com/statistics/248074/
   most-popular-us-social-networking-apps-ranked-by-audience/.


   democratic-electorate-twitter-real-life.html.


https://twitter.com/chrislhayes/status/117150620906205186


https://twitter.com/KarenLTaylor2/status/1178318017282879493

https://twitter.com/eclecteelectric

[31] Name Unknown. Twitter post. August 14, 2019, 8:39 a.m.  
https://twitter.com/cantwealljustg5/status/1161618289342201857

[32] Name Unknown. Twitter post. August 14, 2019, 10:16 a.m.  
https://twitter.com/papajhorn/status/116164280909237625

[33] Name Unknown. Twitter post. August 14, 2019, 2:51 p.m.  
https://twitter.com/esraisttoll/status/1161712080489197568


Appendices

A random_sample.py

---

random_sample.py

Andrew Shackelford
ashackelford@college.harvard.edu
B.A. Candidate in Computer Science, Secondary in Government
Senior Thesis

random_sample.py

Use Tweepy to gather a random sample of all tweets using
Twitter’s sample stream API.

---

```python
import tweepy
import json
import time
import os
import threading
import traceback

tweets = []
total_tweets = 0
start_time = ""
done_sampling = False
write_lock = threading.Lock()

def stream_writer():
    global tweets
    global done_sampling
    global total_tweets
    global start_time
    global write_lock

    while True:
        if len(tweets) >= 10000 or done_sampling:
            # make output file string
            datestr = time.strftime("%Y_%m_%d")
            if not os.path.isdir(datestr):
                os.mkdir(datestr)
            timestr = time.strftime("%Y_%m_%d-%H_%M_%S")
            outstr = datestr + "/" + timestr + ".json"

            with open(outstr, 'w') as f:
                # write tweets to file
                write_lock.acquire()
                total_tweets += len(tweets)
                json.dump(tweets, f)
                print("wrote " + timestr + " to file with " + str(len(tweets)) + "
tweets."")
                print(str(total_tweets) + " total tweets downloaded since " +
                      start_time + ".")
                tweets = []
                write_lock.release()
```

---
if done_sampling:
    return
else:
    time.sleep(1)

#override tweepy.StreamListener to add logic to on_status
class StreamListener(tweepy.StreamListener):
    def on_status(self, status):
        # handle incoming tweet
        global tweets
        global write_lock
        write_lock.acquire()
        tweets.append(status._json)
        write_lock.release()

def load_credentials():
    with open(os.path.join(os.getcwd(), '..', 'credentials.json'), 'rb') as f:
        credentials = json.load(f)
    return credentials

def initialize_api(creds):
    auth = tweepy.OAuthHandler(creds['consumer_key'], creds['consumer_secret'])
    auth.set_access_token(creds['access_token'], creds['access_token_secret'])
    api = tweepy.API(auth)
    return api

def main():
    global done_sampling
    global start_time

    # load credentials and api
    start_time = time.strftime('%Y_%m_%d-%H_%M_%S')
    credentials = load_credentials()
    api = initialize_api(credentials)

    # create listener and writer
    streamListener = StreamListener()
    stream = tweepy.Stream(auth=api.auth, listener=streamListener)
    writer_thread = threading.Thread(target=stream_writer)
    writer_thread.start()

    while True:
        try:
            # sample tweets (blocking)
            stream.sample()
        except KeyboardInterrupt:
            print("Quitting due to keyboard interrupt")
            # cause writer to write out to file, and wait for it to finish
            done_sampling = True
            writer_thread.join()
            exit()
        except Exception as e:
            print("Error occurred")
            print(e)
            traceback.print_exc()

if __name__ == "__main__":
    main()
gather_political_twitter_user_ids.py

Use Tweepy to gather all accounts from the specified set of Twitter lists, and manually add a few accounts plus those that Trump personally follows.

import tweepy
import json
import os

def initialize_api(creds):
    auth = tweepy.OAuthHandler(creds['consumer_key'], creds['consumer_secret'])
    auth.set_access_token(creds['access_token'], creds['access_token_secret'])
    api = tweepy.API(auth)
    return api

def load_credentials():
    with open(os.path.join(os.getcwd(), '..', 'credentials.json'), 'r') as f:
        credentials = json.load(f)
    return credentials

def main():
    credentials = load_credentials()
    api = initialize_api(credentials)

    with open('political_twitter_lists.json', 'r') as f:
        political_twitter_lists = json.load(f)

    political_user_ids = set()
    for political_twitter_list in political_twitter_lists:
        account_lst =
            api.list_members(owner_screen_name=political_twitter_list['screen_name'],
                             slug=political_twitter_list['slug'],
                             count=5000)
        for account in account_lst:
            political_user_ids.add(account.id)

    # Manually add a few accounts that aren't covered by these lists:
    manual_add = ['POTUS', 'mike_pence', 'IvankaTrump', 'EricTrump', 'PressSec']
    for screen_name in manual_add:
        political_user_ids.add(api.get_user(screen_name=screen_name).id)

    # Manually add a list of people who Trump follows
    trump_friends_ids = api.friends_ids(screen_name="realDonaldTrump")
    political_user_ids.update(trump_friends_ids)

    with open('political_user_ids.json', 'w') as f:
        json.dump(list(political_user_ids), f)

if __name__ == "__main__":
    main()
<table>
<thead>
<tr>
<th>Twitter Accounts In Political Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>@11thHour</td>
</tr>
<tr>
<td>@1Bobcohn</td>
</tr>
<tr>
<td>@1PatriciaMurphy</td>
</tr>
<tr>
<td>@60Minutes</td>
</tr>
<tr>
<td>@aacuca1</td>
</tr>
<tr>
<td>@AaronBlake</td>
</tr>
<tr>
<td>@AaronKatersky</td>
</tr>
<tr>
<td>@aaronzinter</td>
</tr>
<tr>
<td>@AbbieBoudreau</td>
</tr>
<tr>
<td>@Abby4Iowa</td>
</tr>
<tr>
<td>@abjyphilip</td>
</tr>
<tr>
<td>@AB11_WTVD</td>
</tr>
<tr>
<td>@ABC30</td>
</tr>
<tr>
<td>@ABC7NY</td>
</tr>
<tr>
<td>@ABCNewsAlsion</td>
</tr>
<tr>
<td>@ABCPolitics</td>
</tr>
<tr>
<td>@abettel</td>
</tr>
<tr>
<td>@ABonTV</td>
</tr>
<tr>
<td>@ABRIndusNY</td>
</tr>
<tr>
<td>@AC60</td>
</tr>
<tr>
<td>@Acosta</td>
</tr>
<tr>
<td>@adambeam</td>
</tr>
<tr>
<td>@AdamKushner</td>
</tr>
<tr>
<td>@adamoldmanNYS</td>
</tr>
<tr>
<td>@adamhousley</td>
</tr>
<tr>
<td>@adamjohnsonNYC</td>
</tr>
<tr>
<td>@AdamKinzinger</td>
</tr>
<tr>
<td>@adamnagourney</td>
</tr>
<tr>
<td>@adamplye</td>
</tr>
<tr>
<td>@AdamSchiff</td>
</tr>
<tr>
<td>@AdamSerwer</td>
</tr>
<tr>
<td>@adamsmithtimes</td>
</tr>
<tr>
<td>@AdamWolnner</td>
</tr>
<tr>
<td>@adrianaisadaz</td>
</tr>
<tr>
<td>@AFiancoTX</td>
</tr>
<tr>
<td>@AFTERhebell</td>
</tr>
<tr>
<td>@agegan</td>
</tr>
<tr>
<td>@agulaparte</td>
</tr>
<tr>
<td>@AHMcIcom</td>
</tr>
<tr>
<td>@AHF regelm</td>
</tr>
<tr>
<td>@ajaffe</td>
</tr>
<tr>
<td>@alaceostate</td>
</tr>
<tr>
<td>@albamicosa</td>
</tr>
<tr>
<td>@AlexMacGillis</td>
</tr>
<tr>
<td>@alex_mallin</td>
</tr>
<tr>
<td>@AlexRoarty</td>
</tr>
<tr>
<td>@alexanderbolton</td>
</tr>
<tr>
<td>@alexburnesNYY</td>
</tr>
<tr>
<td>@alexcast</td>
</tr>
<tr>
<td>@alexlevenstein</td>
</tr>
<tr>
<td>@AlexTVNews</td>
</tr>
<tr>
<td>@AlexNBCNews</td>
</tr>
<tr>
<td>@AlexPadillaCA</td>
</tr>
<tr>
<td>@AlexPapag</td>
</tr>
<tr>
<td>@AlexParkerDC</td>
</tr>
<tr>
<td>@alexromano</td>
</tr>
</tbody>
</table>
D Political Sample Streaming Code

D.1 pt_parent_process.py

---

Andrew Shackelford
ashackelford@college.harvard.edu
B.A. Candidate in Computer Science, Secondary in Government
Senior Thesis

pt_parent_process.py
A parent process to independently call three subprocesses, one for each stream of data from politically relevant accounts.

---

import subprocess
import threading

def subprocess_one():
    subprocess.call(["python", "political_twitter_sample_1.py"])

def subprocess_two():
    subprocess.call(["python", "political_twitter_sample_2.py"])

def subprocess_three():
    subprocess.call(["python", "political_twitter_sample_3.py"])

def main():
    while True:
        print("restarting and calling subprocesses")
        thread_one = threading.Thread(target=subprocess_one)
        thread_two = threading.Thread(target=subprocess_two)
        thread_three = threading.Thread(target=subprocess_three)
        thread_one.start()
        thread_two.start()
        thread_three.start()
        thread_one.join()
        thread_two.join()
        thread_three.join()

if __name__ == "__main__":
    main()
import tweepy
import json
import time
import os
import threading
import traceback

tweets = []
total_tweets = 0
start_time = ""
done_sampling = False
write_lock = threading.Lock()

def stream_writer():
    global tweets
    global done_sampling
    global total_tweets
    global start_time
    global write_lock

    while True:
        if len(tweets) >= 10000 or done_sampling:
            # make output file string
datestr = time.strftime("%Y_%m_%d_1")
if not os.path.isdir(datestr):
o.mkdir(datestr)
timestr = time.strftime("%Y_%m_%d-%H_%M_%S")
outstr = datestr + "/political_1_" + timestr + ".json"

with open(outstr, 'w') as f:
    # write tweets to file
write_lock.acquire()
total_tweets += len(tweets)
json.dump(tweets, f)
print("wrote " + timestr + " to file with " + str(len(tweets)) + " tweets.")
print(str(total_tweets) + " total tweets downloaded since " + start_time + ".")
tweets = []
write_lock.release()

if done_sampling:
    return
else:
time.sleep(1)

# override tweepy.StreamListener to add logic to on_status
class StreamListener(tweepy.StreamListener):
def on_status(self, status):
    # handle incoming tweet
write_lock.acquire()
tweets.append(status._json)
write_lock.release()
def load_credentials():
    # load credentials
    with open(os.path.join(os.getcwd(), '..', 'credentials1.json'), 'rb') as f:
        credentials = json.load(f)
    return credentials

def initialize_api(creds):
    # initialize the api
    auth = tweepy.OAuthHandler(creds['consumer_key'], creds['consumer_secret'])
    auth.set_access_token(creds['access_token'], creds['access_token_secret'])
    api = tweepy.API(auth)
    return api

def get_political_user_ids():
    with open('political_user_ids_1.json', 'r') as f:
        political_user_ids = json.load(f)
    return map(str, political_user_ids)

def main():
    global done_sampling
    global start_time

    # load credentials and api
    start_time = time.strftime("%Y_%m_%d-%H_%M_%S")
    credentials = load_credentials()
    political_user_ids = get_political_user_ids()

    try:
        # create writer
        done_sampling = False
        writer_thread = threading.Thread(target=stream_writer)
        writer_thread.start()

        # create listener
        api = initialize_api(credentials)
        streamListener = StreamListener()
        stream = tweepy.Stream(auth = api.auth, listener=streamListener)

        # get filtered tweets by user id (NOT blocking)
        stream.filter(follow=political_user_ids, async=True, stall_warnings=True)
        time.sleep(60 * 60)

        # restart stream after an hour
        stream.disconnect()
        done_sampling = True
        writer_thread.join()
    except KeyboardInterrupt:
        print("Quitting due to keyboard interrupt")
        # disconnect stream, cause writer to write out to file, and wait for it to
        # finish
        stream.disconnect()
        done_sampling = True
        writer_thread.join()
        exit()
    except Exception as e:
        print("Error occurred")
        print(e)
        traceback.print_exc()
        stream.disconnect()
if __name__ == "__main__":
    main()
gather_trump_ground_truth.py

---
Andrew Shackelford
ashackelford@college.harvard.edu
B.A. Candidate in Computer Science, Secondary in Government
Senior Thesis

gather_trump_ground_truth.py
Use Tweepy to gather all of Trump's available tweets and
---

```python
import os
import json
import tweepy
import time
from datetime import datetime

def load_credentials():
    # load credentials
    with open(os.path.join(os.getcwd(), '..', 'credentials.json'), 'rb') as f:
        credentials = json.load(f)
    return credentials

def initialize_api(creds):
    # initialize the api
    auth = tweepy.OAuthHandler(creds['consumer_key'], creds['consumer_secret'])
    auth.set_access_token(creds['access_token'], creds['access_token_secret'])
    api = tweepy.API(auth)
    return api

def gather_trump_tweets(api):
    trump_tweets = []
    i = 1
    while True:
        # gather all pages of data
        res = api.user_timeline('@realDonaldTrump', page=i, count=200)
        if len(res) == 0:
            break
        for status in res:
            trump_tweets.append(status._json)
        i += 1
    return trump_tweets

def main():
    api = initialize_api(load_credentials())
    while True:
        if datetime.now().year == 2020:
            results = gather_trump_tweets(api)
            with open('trump_tweets_ground_truth.json', 'w') as f:
                json.dump(results, f)
            break
        time.sleep(1)

if __name__ == '__main__':
    main()
```
F Candidate Identifications

- Andrew Yang
  - “andrew” AND “yang”
- Elizabeth Warren
  - “warren”
- Beto O’Rourke
  - “beto”
  - “rourke”
- Pete Buttigieg
  - “buttigieg”
  - “mayor” AND “pete”
- Bernie Sanders
  - “bernie”
  - “sanders”
- Amy Klobuchar
  - “klobuchar”
- Kamala Harris
  - “kamala”
  - “harris”
- Julian Castro
  - “castro”
- Cory Booker
  - “booker”
- Joe Biden
  - “biden”
- Tom Steyer
  - “steyer”
- Tulsi Gabbard
  - “tulsi”
  - “gabbard”