BOTNET CAMPAIGN DETECTION ON TWITTER

A Master’s Project
Presented to
Department of Computer and Information Sciences
SUNY Polytechnic Institute
Utica, New York

In Partial Fulfillment
Of the Requirements for the
Master of Science Degree

by
Jeremy D. Fields

August 2016

© Jeremy Fields 2016
Botnet Campaign Detection on Twitter

Except where reference is made to the work of others, the work described here is my own or was done in collaboration with my advisor and/or the members of the advisory committee. Further, the content of this work is truthful in regards to my own work and the portrayal of other's work. This work, I claim, does not include proprietary or classified information.

Jeremy Fields
Botnet Campaign Detection on Twitter

Master of Science Thesis in Computer and Information Sciences

Department of Computer Sciences

SUNY Polytechnic Institute

Approved and recommended for acceptance as a thesis in partial fulfilment of the requirements for the degree of Master of Science in Computer and Information Sciences

Sept. 30, 2016

DATE

Saumendra Sengupta, Ph.D.

Joshua White, Ph.D.

Scott Spetka, Ph.D.
ABSTRACT

The goal of this thesis is to investigate and analyze botnet activity on social media networks. We first start by creating an algorithm and scoring method for “likely bots,” and analyze them in conjunction with their neighboring messages to determine whether there is a likely group of bots, or botnet. Chapters 1 & 2 cover the overview of the work, and the previous research done by others.

Multiple datasets were collected from Twitter, over different time frames, including random samples, and targeted topics. Chapters 3 & 4 cover the methodology and results of the approach using these datasets. The method is shown to have high accuracy.
ACKNOWLEDGMENTS

Foremost, I would like to thank my parents Karen & Glenn. This would not have been possible without their continued love and support. They have, and continue to aid and encourage me through any endeavor.

I would also like to thank my adviser, Dr. Saumendra Sengupta, and also Dr. Joshua White for their support and guidance with this project.
# TABLE OF CONTENTS

ABSTRACT .................................................................................................................... iv  
ACKNOWLEDGMENTS ................................................................................................. 1  
TABLE OF CONTENTS ............................................................................................... 2  
LIST OF TABLES ........................................................................................................... 5  
LIST OF FIGURES ......................................................................................................... 7  
ABBREVIATIONS .......................................................................................................... 9  
NOMENCLATURE ........................................................................................................ 10  
Chapter 1 ..................................................................................................................... 11  
1.1 Overview ............................................................................................................... 11  
1.2 About ..................................................................................................................... 12  
1.3 Why & Significance ............................................................................................... 13  
1.4 Previous work ......................................................................................................... 15  
1.5 Importance and Relevance of Social Networks ..................................................... 16  
1.6 Problem .................................................................................................................. 17  
1.7 New Approach, or Extension of Existing? ............................................................. 19  
1.8 Does it Solve, or Alleviate? ................................................................................... 19  
1.9 Scalability and Extensibility .................................................................................. 20  
1.10 Sources of Data ..................................................................................................... 21  
1.11 Limitations ............................................................................................................ 21  
1.12 Assumptions ......................................................................................................... 22  
Chapter 2 ..................................................................................................................... 23  
Literature Review ......................................................................................................... 23  
Chapter 3 ..................................................................................................................... 41  
3.1 Equipment ............................................................................................................... 41  
3.2 Datasets Used ......................................................................................................... 42  
3.3 Data Treatment & Arrangement ............................................................................ 49  
3.4 Tools & Libraries ................................................................................................... 50  
3.5 Method .................................................................................................................... 52
Appendix A: ................................................................................................................................. 122
Appendix B: ................................................................................................................................. 125
Appendix C: .................................................................................................................................... 129
LIST OF TABLES

Table 1: Specifications of desktop ................................................................. 41
Table 2: Specifications of laptop ................................................................. 42
Table 3: "20k" dataset statistics ................................................................. 43
Table 4: "1k" dataset statistics ................................................................. 43
Table 5: "jadine" dataset statistics ............................................................. 44
Table 6: "isis" dataset statistics ................................................................. 44
Table 7: "politics_250k" dataset statistics .................................................. 44
Table 8: "red4tum" dataset statistics .......................................................... 45
Table 9: "sample1" dataset statistics .......................................................... 45
Table 10: "pure_spam" dataset statistics ..................................................... 45
Table 11: "politics_10k" dataset statistics ................................................... 46
Table 12: "superbowll1" dataset statistics .................................................... 46
Table 13: "superbowllive" dataset statistics ................................................ 46
Table 14: "2_clean_25k" dataset statistics .................................................. 46
Table 15: "ny_primary_250k_1" dataset statistics .......................................... 47
Table 16: "ny_primary_250k_2" dataset statistics .......................................... 47
Table 17: "snl" dataset statistics ................................................................. 47
Table 18: "politics" dataset statistics .......................................................... 48
Table 19: Example of a bot's score ............................................................... 59
Table 20: Example thresholds ...................................................................... 61
Table 21: Number of neighbors compared for "1k" ...................................... 62
Table 22: Number of neighbors compared for "red4tum" ............................. 63
Table 23: Number of neighbors compared for "politics10k" .......................... 64
Table 24: Text similarity example ............................................................... 67
Table 25: Example of thresholds used ........................................................ 80
Table 26: Bot statistics on "1k" ..................................................................... 82
Table 27: Bot statistics on "2_clean_25k" ..................................................... 85
Table 28: Example of bot account names in "2_clean_25k" ........................... 88
Table 29: Bot statistics on "20k" ................................................................. 89
Table 30: Bot statistics on "isis" ................................................................... 90
Table 31: Bot statistics on "jadine" .............................................................. 93
Table 32: Bot statistics on "ny_primary_250k_1" .......................................... 96
Table 33: Bot statistics on "ny_primary_250k_2" .......................................... 97
Table 34: Bot statistics on "politics_10k" ..................................................... 99
Table 35: Example of detected bot accounts names .................................... 101
Table 36: Bot statistics on "politics_250k" ................................................... 102
Table 37: Bot statistics on "red4tum" .......................................................... 103
Table 38: Bot statistics on "sample1" .......................................................... 104
Table 39: Bot statistics on "snl" ........................................................................................................... 106
Table 40: Example of detected bot names in "snl" .............................................................................. 107
Table 41: Bot statistics on "superbowl1" ............................................................................................ 108
Table 42: Bot statistics on "superbowllive" ......................................................................................... 109
Table 43: Final statistics 1 .................................................................................................................. 114
Table 44: Final statistics 2 .................................................................................................................. 114
LIST OF FIGURES

Figure 1: Left - sentiment of arbitrary data. Right - sentiment of spam tweets ............29
Figure 2: Sentiment score of users vs. bots [9] ...............................................................29
Figure 3: Tweet timeline example ..............................................................................55
Figure 4: Neighbors compared vs Tweets per second ....................................................66
Figure 5: Neighbors compared vs. Time taken .............................................................66
Figure 6: Entropy of the "isis" dataset ........................................................................69
Figure 7: K-Means clustering on "politics" entropy .......................................................70
Figure 8: Sentiment of normal users vs. spammers [9] ................................................71
Figure 9: Sentiment analysis of random tweets ..........................................................72
Figure 10: Sentiment analysis of spam .......................................................................73
Figure 11: High score percentage of "1k" ..................................................................74
Figure 12: High score percentage of "sample1" ...........................................................74
Figure 13: High score percentage of "superbowl1" .....................................................75
Figure 14: K-Means clustering of high score percentage on "politics_250k" ..............76
Figure 15: K-Means cluster of high score percentage on "superbowl1" ......................77
Figure 16: K-Means cluster of high score percentage on "sample1" .........................78
Figure 17: K-Means cluster of high score percentage on another random sample ....78
Figure 18: Example of an organic network graph .......................................................81
Figure 19: Subset of bot network graph of "1k" ............................................................82
Figure 20: Full bot network graph of "1k" ..................................................................83
Figure 21: Example of detected spam in "1k" ..............................................................84
Figure 22: Full bot network graph of "2_clean_25k" ..................................................85
Figure 23: Example of detected spam in "2_clean_25k" .............................................86
Figure 24: Example of detected spam in "2_clean_25k" .............................................87
Figure 25: Full bot network graph of "20k" ..................................................................89
Figure 26: Example of detected spam in "20k" ............................................................90
Figure 27: Full bot network graph of "isis" .................................................................91
Figure 28: Example of detected spam in "isis" .............................................................92
Figure 29: Full bot network graph of "jadine" ...............................................................93
Figure 30: Subset of bot network graph "jadine" ..........................................................94
Figure 31: Example of detected spam in "jadine" ........................................................95
Figure 32: Full bot network graph of "ny_primary_250k_1" .......................................96
Figure 33: Full bot network graph of "ny_primary_250k_2" .......................................98
Figure 34: Full bot network graph of "politics_10k" ...................................................99
Figure 35: Example of detected spam in "politics_10k" .............................................100
Figure 36: Full bot network graph of "politics_250k" ................................................102
Figure 37: Full bot network graph of "red4tum" ........................................................103
Figure 38: Full bot network graph of "sample1" ........................................................104
Figure 39: Example of detected spam in "sample1" .................................................. 105
Figure 40: Full bot network graph of "snl" ................................................................. 106
Figure 41: Full bot network graph of "superbowl1" ................................................... 108
Figure 42: Example of detected spam in "superbowl1" ............................................ 109
Figure 43: Full bot network graph in "superbowllive" ............................................ 110
Figure 44: Example of detected spam in "superbowlive" ........................................ 111
Figure 45: Bot creation times .................................................................................. 112
ABBREVIATIONS

OSN – Online Social Network
SVM – Support Vector Machine
KNN – K Nearest Neighbors
SMO – Social Media Optimization
URL – Uniform Resource Locator
NOMENCLATURE

Bots – Software application that runs automated tasks over the internet

Botnet – A group of bots

Friends – On Twitter, when two people follow each other (bi-directional)

Followers – On Twitter, when one person follows another (uni-directional)

Tweet – A message posted to Twitter, limited to 140 characters
Chapter 1
Introduction

1.1 Overview

Twitter is a ‘microblogging’ social media website. It stands out from its competitors, such as Facebook and LinkedIn by the fact that it limits posts or tweets (text-based message) to only 140 characters. Twitter is also unique in that relationships can be directed, whereas on Facebook, for example, they are not. This is made possible due to the way that Twitter allows relationships between users to be created. There are two main relationships: friend and follower. A follower is defined as a directed edge from user A to user B, such that user A will see what user B posts, but not the other way around. If user A is a friend of user B, then this would be defined as: user A follows user B AND user B follows user A. Simply put, both users will see what each other posts if they are friends. In most social networks, there is only the notion of being “friends,” where the relationship is undirected and reciprocal.

Twitter is also to blame for the popularization of the octothorpe (#) sign, or as it’s more popularly known on Twitter, a hashtag. Unexpectedly from Twitter, hashtags grew organically from the user base as a means to denote a tweet pertaining to a particular topic [25].

Twitter is one of the largest websites in the world, and as of the time of this writing, it is ranked the 10th most popular site globally, as reported by Alexa [42]. Similarly, it is
ranked as the 8th most popular site in the United States. Twitter boasts having 320 million monthly active users, and over 1 billion unique monthly visits. Furthermore, they boasted that an astounding 500 million tweets are sent every day [26]. Yet another astounding statistic, although outdated due to Twitter selectively revealing official statistics at their will, is that they boasted on average that 460,000 new accounts were created daily in 2011 [28].

Twitter also boasts a generous public API, giving 1% of all tweets via a stream for free to anyone who has a set of developer API keys, which can be obtained by anyone for free. Along with this, they also offer several other API functions, such as the ability to stream tweets based on keywords or geolocation, among other filters. This has led to extensive research in many areas, due to the wealth of information publicly available.

From everything stated above, it is clear that due to the popularity and number of users, that spammers and bots have often chosen Twitter as their platform of choice. Of course, this is not unique to Twitter, as any popular social media site will see its fair share of bots and spammers. While statistics vary between research papers, such as 16% [3], 7% [19], and 10.5% [1] to name a few, Twitter has, on at least one occasion, officially stated that approximately 8.5% of all active users are likely bots, as stated in their 2013 SEC filing [28].

1.2 About

This thesis is an approach to detecting a subset of bots on Twitter, that at best is under-researched. This approach will be generic enough to be adaptable to most, if not all social networks. Performance may vary, depending on the amount of information available, but overall it should be adequate. Twitter was chosen only due to its free public API which provides a wealth of data to work with.
The subset of bots I chose to focus on are those that can evade most, if not all current detection methods. This is simply because they have little to no information associated with them that can be analyzed to make a determination. Although any account on Twitter inherently has information associated with it, it is very easy to blend in with the majority of users who are simply “lurkers” - those who only consume content, but do not contribute. How can you determine if an account is a bot if they don’t do anything? By the time they act, it will be too late to detect. The only solution would be a real time, or near real-time, detection algorithm.

The kinds of bots that would have these properties would be those who don’t have immediate monetary goals, as most do. The majority of bots are driven ultimately by money, either by spreading monetized URLs, or URLs that will infect a user to enlist their computer into a botnet, of which access can be sold to, or used for other means. The bots in this subset primarily would be those that have goals of shifting public opinion. This could be celebrities, politicians, or even nation state actors. It is certainly not an unreasonable assumption that an actor could be building up a “sleeper” botnet over time for later use. In the meantime, these bots would be inactive, and to all others, simply appear as “lurkers” or inactive accounts. Current bots are relatively easily detectable, as they rely on constantly sending malicious URLs, sending out an above average amount of tweets, or having suspicious friend networks - to name a few attributes.

1.3 Why & Significance

I have actively been researching and analyzing Twitter data since January 2015, for various reasons, personal and professional. In my efforts, when gathering data for my work on [35,36,37], I happened by chance to see something unusual. At the time, I was building a dataset of approximately 10,000,000 tweets, using the Twitter API to obtain tweets containing specific keywords. The research was based on politics, so the dataset was filtered by the most popular hashtags, in regards to politics, at the time of collection. Some
of these keywords were words like: vote, politics, democrat, republican, etc. In this collection, I noticed a large number of tweets, all sent within milliseconds of each other, containing text such as:

"Vote for JaDine at http:\/\t.co\Snw9dZjc1Z \ forty-four #PushAwardsJaDines"

Upon further inspection, I noticed that each tweet was from a separate account. Furthermore, each tweet was slightly different. Note in the example above, the text “forty-four” - this changed every ten tweets or so, to increment to the spelled out version of the next number. This was caught in my dataset, simply because of the word “vote.” Due to the number of similar tweets, it was self-evident that this was a malicious campaign to try and gain votes for some nature of an award. It turns out “JaDine” is a Filipino TV show, of which the main actors are James, and Nadine [46].

As stated in others’ research, bots can influence public opinion [11,33,34,44,45], especially the reporting done in [45] where the Syrian intelligence agency is alleged to have used Twitter and Twitter bots to attempt to shift public opinion. This is certainly an extremely powerful tool, and as with most powerful tools, there is the inevitability that it will be used for malicious or evil purposes at some point. I do not intend to address any philosophical aspects of shifting public opinion, but some may inherently see any program capable of this as evil.

Regardless of how you see the topic of botnets shifting public opinion, it is undeniably a powerful tool possibly capable of societal change. For this reason, this is project certainly has significance.
1.4 Previous work

In reference to 1.3, this is not easily detectable. These bots would evade most methods. Most approaches looking at usernames would fail unless they were looking for common substrings. Those looking for sequences of numbers appended on common usernames would not detect this. Similarly, some may be detected because they all have common URLs, but those methods created before Twitter masked URLs in the text of a tweet by proxying all URLs through Twitter’s own URL shortener would require updating. Also, as the text of each tweet is different, those methods that detect the simpler method employed by bots (e.g. 1-2 character differences, sequence of numbers in text) would fail. In addition, the majority of methods primarily look at network analysis. This would likely fail, as a personal inspection of a small sample of these accounts shows a wide variance in regards to friends, followers, and friend/follower ratio, with no discernable characteristics or red flags. Admittedly, some of the existing approaches would likely detect this.

However, all investigated research could easily be beaten. What if URL wasn’t used? What if usernames are sufficiently and randomly generated to look organic? What if they friend and follow fellow bots to make their networks look organic? What if they, up until this point, have no history of messages?

If a malicious actor wanted to beat these detection methods, it could certainly be done. With [28] reporting that there are possibly currently 8.5% of all Twitter accounts which have never performed any action, there is immediately a wealth of bots that have little information of value to analyze. Furthermore, it would certainly be possible to friend and follow fellow bot accounts to create organic friend/follower ratios for each bot. It also would certainly be possible for these accounts to have sufficiently different usernames. Now, what if these bots tweeted messages with a safe, or no URL, in coordination to shift public opinion? These current methods would not detect this. Even if one could, how can you detect this before it’s too late?
Even in the bots mentioned, they have clearly evaded bot detection algorithms. From my manual inspection of a small sample of accounts, all were still active, and not yet suspended. It is worth noting that this data was collected in October 2015, and they are still active as of April 2016. What they’re doing works, and while it would certainly be difficult to measure, they may have possibly shifted public opinion.

What if this was a nation state?

1.5 Importance and Relevance of Social Networks

In this information age, social media has become ever prevalent. As reported by the website Statista, approximately 2.04 billion people used at least one form of social network in 2015. This number is predicted to grow to 2.55 billion by 2018. In the United States, 78% of citizens have a social network profile. Globally, 31% of all humans use at least one social network platform [43].

Clearly this is a powerful tool, which can have far-reaching effects, either for good or bad. Beyond social interaction between individuals, social media networks have become an important vehicle for organizations to advertise, promote, and also to interact and communicate with their current and potential customers.

Social networks have the potential to be used for malicious purposes as well, enough so that DARPA has funded at least one project on the detection of actors utilizing social networks to shift public opinion [33]. It has also been shown that politicians [44] and nation states [45] have used social media in attempts to shift public opinion. This can pose a very serious threat, and thusly social networks and their workings are a research-worthy topic. As this number of people using social media has consistently grown year after year since at least 2010 as reported by [43], and it is predicted to continue this growth,
to encompass over one-third of all humans by 2019, it is certainly relevant and important to research topics in regards to social media.

1.6 Problem

In this digital age, bots, spammers, cyborgs, and otherwise malicious actors have always been in an arms race with the platforms that they act on. These actors’ goals are typically, and most often, ultimately driven by monetary goals. Whether this is through the spread of malicious URLs which contain links to sites which host drive-by downloads of malicious software, or simply through the spread of URLs to monetized sites, it is ultimately for the goal of monetary gain. In the case of malicious software, rarely is it the case that an actor simply wants to harm others; rather, it is often the case that this malicious software can do something that can benefit the malicious actor. This may be through software such as CryptoLocker[40], which encrypts data unless a ransom is paid to decrypt your data. It may be through more subtle means such as silently installing software which will enlist a user's computer to a botnet, the usage of which can later be sold. There are many types of spam, but they typically ultimately lead to some monetary goal. Rarely is it the case that the goal is limited to harm only, although this is occasionally the case. These situations, as stated, are rare but are often the works of state actors, or other organizations with goals much larger than monetary gain. This was the case with StuxNet, whose goal was to harm an Iranian nuclear enrichment facility to delay the progress towards creating a nuclear weapon [41]. No matter the bot, or method that bots use, it is always an arms race between bot evasion and bot detection methods. While bot detection methods, specifically on Twitter, are a highly researched area, I posit that there is a subset of bots that rarely act, but can possibly have a much larger impact than encrypting a user's files, or infecting a user's computer. While this is not a new notion, it certainly is an under-researched area. The problem I pose is that how can you detect bots, which otherwise have no data
associated with them that can be used for detection until it’s too late? This has a loaded answer, as there are many aspects to this.

Consider that Twitter has reported in 2014 that approximately 8.5% of all Twitter accounts were ones that never performed any action [28]. Twitter has since ceased to report this statistic, and has never reported the total number of registered accounts. However, others have stated that the total registered accounts is approximately 1.3 billion. This would mean that there are approximately 110,500,000 inactive accounts. Depending on your perspective, this could mean that there are 110,500,000 sleeper bot accounts. I understand that this is highly unlikely, but it would be unwise to assume that there is a negligible number of bots in that set of accounts.

Depending on the motives of these bots, the problem could result at best as an annoyance, and at worst, societal changes caused by actors with malicious goals.

The majority of current and past research is largely overlapping in regards to what pieces of information are considered when determining whether an account is a bot or legitimate. The most often used attributes are: friends, followers, friend/follower ratio, URLs, and network analysis. This is fine, and undeniably serves its purpose, and I don’t doubt the validity of their research.

However, there are bots who will not be detected by any of these methods. Consider an actor whose motivation is not the goal of spreading malicious, or monetized URLs. Consider a botnet that rarely, if ever, tweets, and has an organic-like friend count, follower count, and friend/follower ratio by mostly friending other bots. As these bots rarely tweet, any analysis on how often they tweet, and time between tweets (avg., median, etc.) would not reveal anything. Also, consider that these bots, when they do tweet, all have non predictable, randomly generated names, and the contents of their tweets are randomized to some degree so that some, or all are different. Also, as these bots have no malicious or monetary goals, their tweets contain no URLs that could be analyzed. How can you detect a bot if there is no information to analyze?
Now, consider that these bots are operated by a state actor, and their goal is to shift public opinion. This would be the worst case scenario, but these tactics have been mentioned in others’ papers as ones used by politicians [33], celebrities, and I’ve found similar tactics in my own research, which will be detailed further on. It is certainly not an unheard of subject. This is noted in a paper by DARPA [33], and [34,44,45], and an experimentation from Zhang et al. in [11].

1.7 New Approach, or Extension of Existing?

It would be hard for me to conclusively say that my approach is completely new, or extensions of existing work. The design and idea are completely my own. However, there are others who have the same high-level design as my own. Specifically, in the sense that I am doing simple comparisons on attributes, as opposed to the very common machine learning approach. Similarly, with the attributes used, the combination used is my own, and is not an extension of others. Of course, others do look at some of the attributes I use. The attributes chosen were influenced by the work of others, most notably the usage and thresholds for using entropy as in the works of White and Matthews [2], and the use and threshold for sentiment analysis by Hu et al. in [9].

1.8 Does it Solve, or Alleviate?

It would be naive to assume that this is an end-all solution. More importantly, I do not intend for this to be so, only merely to detect bots that other methods will likely miss. Regarding catching all coordinated, moderate-to-high volume spam campaigns, then I would hope that this will solve it. Again, it would be naive to assume that will be the case. Even if it were, bots will always try to evade detection methods, as it’s a constant arms race between bot detection and evasion methods. In the general sense of detecting bots, then this will only alleviate the issue by detecting a subset of all bots.
1.9 Scalability and Extensibility

This design and method, due to its simplicity, will be both scalable and extensible. While it will be beyond the scope of my work to implement the method in this fashion, the design will support this. For example, since a sliding window approach is used when analyzing tweets, the only memory required is: N * 5075. Where N is the number of tweets selected for the sliding window, and 5,075 is the average number of bytes in a single tweet. A sliding window of 1,000 tweets would only require approximately 5MB of memory.

There would be some challenges in making this scalable, but addressing them is certainly within the realm of feasibility. Approximately, on average, 5,700 tweets are sent every second on Twitter [47]. On my own hardware, which will be detailed further on, I can process only 23 tweets per second, when each tweet is compared against 40 neighboring tweets, and 47 tweets per second when each tweet is compared against 20 neighboring tweets. In either scenario, this obviously isn’t fast enough.

However, this program is written in Python which is many orders of magnitude slower than a lower level language, such as C [48]. It would be a safe assumption that if this code was ported to C, heavily optimized, and running on a high-end commercial grade server, there would be a massive increase in tweets processed per second. While it’s beyond the scope of this work to do so, I would take an educated guess and say that processing several thousand tweets per second would certainly be feasible. This is not even considering any distributed processing approaches. If this program could be parallelized, then it of course would be possible to process the full Twitter stream in real time. In a simpler approach, if say speeds of processing 2,000 tweets per second could be achieved on a single machine, then a trade-off could be made by analyzing only one third of the stream by three separate machines. This would reduce accuracy, but as my results sections will show, spam is easily detectable with using only 1% of Twitter’s tweets stream. There are several methods that could be used to analyze 100% of tweets in real time, and I’m sure there are others that have not been discussed. Regardless, analyzing 100% of all tweets is certainly feasible, and at worst, would only slightly reduced accuracy. In the best scenario,
a single high-end machine using an optimized, lower level program would be able to do this.

In regards to extensibility, this program and approach are already written in a way that can be extended upon, as I have had to do this myself when adding or removing metrics from the detection algorithm. This is further supported by the general simplicity of this approach. In others’ approaches, this would require retraining SVMs to support new attributes and metrics. In mine, it could be done in a 30 seconds by writing only a few lines of additional code.

1.10 Sources of Data

My source of data is from Twitter. As mentioned above, Twitter provides a free public API for receiving a stream of tweets. There are various streams that can be consumed, such as: unfiltered 1% of all tweets, filtered by keywords, filtered by geolocation, and others. I have obtained many datasets through work I have done on my own, and in research papers that I have co-authored [35,36,37].

1.11 Limitations

The primary limitations are only that I do not have the funding to subscribe to the full Twitter “firehose” stream which provides 100% of all tweets as provided by GNIP [38]. I am also limited by that even if I could afford this stream, estimated at $30,000 for 50% of all tweets in 2010 [39], I would not have the computational power to process this volume of tweets. Regardless, 1% of all tweets is certainly sufficient data to work with, if needed.
1.12 Assumptions

It is assumed that Twitter will provide statistically sound results, without bias in its ‘Sample Stream,’” or at least a level of soundness and unbiasedness. This is also assumed for queries on search terms.
Chapter 2

Literature Review

“Detecting Automation of Twitter Accounts: Are You A Human, Bot, or Cyborg?” by Chu et al. [1] is very closely related to my topic. Its primary goal is the detection and classification of three groups: humans, bots, and cyborgs. Their approach consists of four components: entropy component for measuring temporal and behavioral complexity, spam detection for determining if content is spam, account properties which gathers useful information such as URLs, devices, and API, and finally a decision maker which uses the previous three components to make decisions. It covers several key points of interest, such as common API usage between each group, follower-to-friend ratio, and usage patterns. These formulas and statistics will certainly help guide my approach. Overall, this was an excellent paper that achieved what they set out to do. If I were to add to this work, I would simply redo what they’ve done on Twitter as it is in its current state. This paper was written in 2011, and much has changed in Twitter since then. While this paper is closely related to my work, they take a far more sophisticated approach by utilizing a classification system, and decision engine. I believe this is unnecessary for bot detection, but certainly has its purpose in a research setting. I believe that bot detection can be simpler, and still attain a high accuracy of correct classification. Although it wasn’t their intention, their approach is far from agile and also more computationally expensive.

“It’s You on Photo?: Automatic Detection of Twitter Accounts Infected with the Blackhole Exploit Kit” by White and Matthews [2] is likely the closest topic to mine in regards to design and approach. White et al. set to characterize and classify tweets related to the Blackhole Exploit Kit by utilizing several trivial methods that weren’t reliant on any
other external services, as several related papers’ work were. The usage of the MapReduce framework to make large scale analysis is also something I intend to explore, and possibly develop as well, if feasible. Overall, this was a great paper, which accomplished its goal by means of a combination of intuitive and simple classification methods. This is something that is becoming increasingly rare, as often approaches are made more complicated than necessary. As stated, this approach led to conclusive results and statistics. This paper also reinforces my design and approach of combining simple metrics to create conclusive results. It also reinforces the notion of calculating entropy of a tweet, which along with Chu et al. seems to provide a strong indicator of whether a tweet is from a human, cyborg, or bot. Also in common is the statistics concluding that bots tend to have strong correlations with specific APIs. Overall, this was a good paper. If I could repeat the work, I would add other metrics to their characterization of Twitter accounts, and I would also make it suit a more general purpose. While the author achieved their goals, my approach will contain further metrics beyond what they used, and while their approach is specific to one type of bot, mine will aim to be general purpose.

“Detecting and Analyzing Automated Activity on Twitter” by Zhang and Paxon’s [3] approach was to detect automated tweets based primarily on patterns in the times these messages were posted. This work was on 106,573 accounts collected over a three-week period in 2010. This paper is highly relevant, but unfortunately outdated. While published in 2011, the analysis was on data collected in April of 2010. Since this time (over 6 years), Twitter has grown astronomically, and it would be unwise to assume that bot evasion and bot detection algorithms have not evolved since. This paper, as well as several others, have also reinforced the notion that bots tend to use specific APIs. While the Twitter landscape was different in 2010, personal research has shown that there is certainly still some truth to this. While this paper is highly relevant, unlike the other papers reviewed, it is too broad in scope by attempting to detect automated accounts in general. As stated in Chu et al., automation on Twitter is a double-edged sword in the sense that automation can certainly be good from a news organization, for example, but spammers of course are not desirable. This approach certainly has use, but it only covers a subset of the problem today; however,
I plan to investigate the usage of some of their approach to implement in my own. If I were to repeat or add to their work, I would first redo the research as is, on the current Twitter landscape and compare results. Secondly, I would add in further metrics to limit the set of automated tweets to those who are sending or posting spam.

“Detecting Spammers on Social Networks” by Stringhini et al. [4] is about an approach for detecting spam on Myspace, Facebook, and Twitter. Their approach consisted of creating 300 accounts for each network and using each account passively. Specifically, they would not seek out to “friend” or “follow” any other account, but would accept any incoming request. They performed periodic checks on accounts over the course of one year, between June 2009 to June 2010. They then logged each interaction, which is used in a final analysis. Again, this paper is outdated, given the rapid pace of change in social networks; however, their approach is very general, and would likely hold up quite well today. They utilize novel metrics such as message similarity and URL ratio to compute a final score for each bot. In my opinion, while these approaches are appropriate, I believe the addition of machine learning adds unnecessary complexity. Rather than computing common metrics in detected spammers and using machine learning, I imagine a simple algorithm to check for these generic markers would suffice. The authors did indeed achieve their goal. However, I think the “honey pot” approach is less than ideal in this scenario. It certainly makes for easier detection, but has little use in detecting anything remotely close to real-time. I would not repeat or add to their work, but that is only because I prefer an “active” approach rather than a passive one in this scenario. As stated, my work will differ in that it will be an active approach to bot detection. However, this paper is still relevant, and the metrics used are worth investigating.

“The Rise of Social Bots” by Ferrara et al. [5] is primarily talking about the issues and possible solutions, rather than a specific implementation of a new solution. This, however, makes it an excellent paper to reference while working. It primarily focuses on
“human-like bots,” which isn’t particularly related to my work. Regardless, it talks about some important issues and approaches, and further validates this work.

“These bots mislead, exploit, and manipulate social media discourse with rumors, spam, malware, misinformation, slander, or even just noise. This may result in several levels of damage to society. For example, bots may artificially inflate support for a political candidate [29]; such activity could endanger democracy by influencing the outcome of elections. In fact, these kinds of abuse have already been observed: during the 2010 U.S. midterm elections, social bots were employed to support some candidates and smear their opponents, injecting thousands of tweets pointing to websites with fake news [29]. A similar case was reported around the Massachusetts special election of 2010 [30]. Campaigns of this type are sometimes referred to as astroturf or Twitter bombs.”

This paper reinforces the validity and importance of bot detection by discussing the possible social and political problems that can arise from bots. Clearly, this is a relevant topic, worthy of researching. With the upcoming election, and bots ever evolving, the problem will only get worse unless the state of bot detection keeps up. Overall, this is a great paper for reference.

The authors of “Detecting Clusters of Fake Accounts in Online Social Networks” Xiao et al. [6] are employees of LinkedIn, who have access to the full data of each account. They use this data in a three step process: first building a cluster of accounts, then a ‘profile featurizer,’ and finally a decision-making step. They employ machine learning techniques as many other researchers have, but these authors have access to data that most do not, such as IP addresses. A worthy quote addressing the impact of bots:

“For example, they can weaken the credibility of the network if users start to doubt the authenticity of profile information. They can also have negative impact on the networks’ ad revenue, since advertisers might question the rates they pay to reach a certain number of users if many of them are not real people.”

Again, this paper reinforces the validity of this topic, although from a different angle than Emilio et al. by talking essentially about brand degradation, which is caused by mistrust in
an OSN. The author accomplishes their goal, which results in a high accuracy cluster detection, claimed at 95% accurate matches. Overall this paper was good, and gives insight to other metrics that may be added to my toolbox, such as pattern detection in naming schemes. If I were to do this research again, I would do it as is. Either I’m skeptical that such a high accuracy rate can be detected for entire clusters, or the author is only trying to catch large bot networks which are unsophisticated such that they are easily detected due to: using the same IP, similar names, registered very closely in time, etc. and not trying to detect any other kinds of bots. My work intends to achieve a very similar goal, but I believe it can be done in a much simpler way.

“Tweetnet: Finding the Bots in the Flock” by Blum et al. [7] is only loosely relevant, but discusses issues of detecting bots in a botnet. Specifically, the authors set up a mock Twitter environment, and made a game of it. This consisted of two teams, each of which was in control of a bot master. The goal was to determine, in a stream of tweets, which account was the bot master. While this paper is on the informal side, and only loosely relevant, it still sheds further light on methods used for detecting bots. Some techniques coincide with other papers, such as analyzing the frequency and time of tweets sent. Overall, it was an interesting read. It was more of a project and learning exercise than anything, but as stated, still sheds light on some of the issues to consider when both making a bot, and detecting one. If I were to repeat this work, I would want to do a similar game, but using Twitter rather than a mock version. This would certainly have its hurdles, but would be a fun learning experience regardless.

“Uncovering Social Spammers: Social Honeypots + Machine Learning” by Lee et al. [8] is very similar to Stringhini et al., and were published around the same time frame, presumably unbeknownst to each other. They both had the approach of creating “honey pot” accounts, and both did so on both MySpace and Twitter. They both employed a passive approach of only investigating users who performed an action on them, such as: friend requesting, direct messaging, and public messaging. They then use a support vector
machine learning approach by manually training it on manually classified information. This paper was good, but the data used is now outdated. They collected Myspace data from Oct 2007 to Jan 2008. Myspace, for all intents and purposes, has been abandoned since then, in favor of Facebook. Twitter data was scraped over a two-month period beginning in August 2009. Again, Twitter has changed drastically since this time, and so have the bots. Overall, this paper did not provide any new insight over Stringhini et al., and this is merely due to the fact that I read this paper second. If I were to repeat their work, I would repeat the experiment as is in the current landscape in both OSNs. I suspect that their methods would not stand the test of time, although expectedly, due to bots getting more sophisticated as to overcome these kinds of detection methods.

“Social Spammer Detection with Sentiment Information” by Hu et al. [9] had an approach that included sentiment information, along with other common pieces of information used in bot detection to evaluate its effectiveness. They used two data sets: one from Dec 2009 to Aug 2010, and a second from Aug 2013 to Oct 2013. This was an ‘outside of the box’ approach in my opinion, by including sentiment analysis as an attribute for determining if a user is a spammer or not. They conclude in part that, there’s a direct correlation between sentiment score and spammers. I actually didn’t believe this, so I recreated this experiment with my own tools and my own datasets, and my results ended up being almost exactly in line with theirs.
The important metric in my output is the “Avg. polarity w/out neutrals,” which while a subset of a sentiment score, is the most telling, and most often used. This is compared to the efforts reported by Hu et al. as shown below.

This paper is excellent due to the thorough analysis of sentiment scores and how they relate to bots. The other papers thus far have certainly shed light on intuitive metrics; however, this is something I would have never considered. This will give me another telling, useful tool to use in my analysis. If I had a chance to redo this research, I likely wouldn’t change...
anything. I think they achieved their goal, and have taught me something I wouldn’t have considered. Their work certainly stands up, as my own tests resulted in the same conclusion.

The author Murmann [10] of “Enhancing Spammer Detection in Online Social Networks with Trust-based Metrics” doesn’t give great detail on the data set, only stating they start with a ‘number of known spammers and legitimate users,’ of which he then collected 200 friends and followers for each. The process then only looks at user information, rather than “posts” or “messages.” Specifically, he looks at numbers such as: friend-follower ratio, friends added per day, followers added per day, updates per day, etc. From these numbers, he creates an algorithm to compute a ‘trust’ value from these statistics, in part by using a modified PageRank algorithm. This thesis is similar to several other papers reviewed; however, there is an emphasis on determining ‘trust,’ which is a unique approach thus far. I think this approach has its uses, but can easily be fooled. I think a network of spam bots, each of which friends each other, and is only activated during certain times (i.e., they’ve been hired to do a job) could pass as legitimate, because individual bots don’t tweet often and have a ‘legitimate’ friend-follower ratio, which appears to be the primary metrics utilized. The author also takes into account ‘friends/followers added per day,’ but this is easily fooled by building a botnet over time. In the end, the author claims a high accuracy. While this may be true, I think that at least in the current state of Twitter, it would be ignoring many bots, and likely entire botnets. The author claims that he would like to explore community belongingness as an indicator; however, I think this would be the wrong approach. Consider a bot net, where each bot friends/follows each other. An unsuspecting legitimate user who friends one of these bots would be seen as an outlier who doesn’t belong, and thus would be classified as a bot. Overall, this was an interesting paper, where the notion of a network and trustedness is taken into account. However, I do not intend to look at these metrics for various reasons, including those stated above. Lastly, this is certainly not a real-time, or near real-time system, which is an aspect I hope to address. If I were to do this research again, I would
want to do it on a different dataset, one that contains sporadically active spamming campaigns to see if it would detect these kinds of bots.

“On the Impact of Social Botnets for Spam Distribution and Digital-influence Manipulation” by Zhang et al. [11] is mostly about an experiment in running a botnet on Twitter. The author starts by purchasing 1,000 Twitter accounts for a mere $57. Then, by abusing Twitter’s system by only retweeting spam from a single bot master, they successfully propagate spam, only at the cost of a single bot master. This was possible due to the fact that Twitter only punishes the originator of spam, not those who retweet it. The author details the methods and evaluation of controlling a botnet on Twitter. This provides very useful insight into the kinds of bots I am trying to detect. This botnet, as with many others, are control by a single master. The bots monitor the master for a signal, and when received, they perform some action. To have a noticeable impact, these actions must be performed within a short period of each other if the goal is to sway public opinion, for example. There would be of little use if there were only a small number of tweets per unit of time. The goal is to be noticed, and therefore the bots must act in unison. My work will be to detect these kinds of spamming campaigns. Overall, this was a good paper. This was more on the side of developing a botnet, rather than detecting one. However, this provides useful insight into how they work, which can aid in detection. If I were to redo this research, I would simply redo it as is. While this paper isn’t too old (2013), I would still want to see how well this style of botnet would hold up in Twitter in its current state, as it is probably a safe assumption that Twitter has ever evolving bot detection algorithms.

“Tweets as Impact Indicators: Examining the Implications of Automated “bot” Accounts on Twitter” by Haustein et al. [12] is on topic in the sense that it is about bots, but is not particularly relevant otherwise. However, it does contain some useful statistics. The authors of this paper aim to discuss the impact of automated Twitter accounts that spread research information, specifically from arXiv[31]. They start by analyzing the state of Twitter at the time of this writing (2016), and cite references stating that other
researchers have found: 16% “exhibit a high degree of automation,” 10.5% are bots, and an additional 36.2% are cyborgs (human assisted bots). They finish the article by showing the results of almost 45,000 tracked articles, which come from a total of 51 accounts. While these are interesting statistics, since it isn’t fully on topic, this review will be kept short.

“Die Free or Live Hard? Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers” by Yang et al. [13] is about the analysis of current bot detection methods at the time of writing (2011), and proposes a new method using neighbor-based detection features, among other methods. They worked with a dataset consisting of almost 500,000 user accounts, and 14 million tweets. They also analyze bot evasion tactics. They break down the evasion tactics into two categories: profile-based feature evasion, and content-based feature evasion. As with several other papers, their proposed solutions, in part, rely on graph based methods. While this has its strong points, it will still leave out botnets. Of course, Twitter was different when this was written five years ago. The authors certainly achieve their goal, and provide excellent analysis on detection methods, and evasion methods used. However, even with their success, they claim to achieve an 83% detection. While certainly admirable, it still leaves out a non-negligible number of bots. My work differs in that it will aim to catch some of the harder to detect bots left in the 17%. Overall, this is a great paper that contains a lot of information that will be referenced in the future.

“Spam Filtering in Twitter Using Sender-Receiver Relationship” by Song et al. [14] as many others, uses a graph-based approach for determining bot accounts. The authors used a dataset consisting of almost 150,000 accounts and 270,000 tweets. This data was collected presumably in 2011, which dates this. Their primary method consisted of calculating the sender-receiver ratio of accounts as a primary indicator to determine if accounts were bots. They also considered the relationships between users, among other things. While this approach, as many others, certainly has its place, I don’t believe it would stand the test of time. In their conclusion, they state that they use distance and connectivity
features as they are hard to manipulate. I would argue that this is not the case, at least in the current state of Twitter. As mentioned in other reviews, bots can certainly friend and follow each other to create a realistic looking, organic network. Overall, this paper was good. It covers many things that have been covered in other papers, which only reinforces their validity. If I could do this research again, I would simply redo it as is in the current state of Twitter, as I suspect it would not perform as well as it did in 2011. This work differs from mine in several ways, primarily the usage of network based features used as indicators for determining if an account is a bot or not.

Yardi et al.’s. [15] work in “Detecting Spam in a Twitter Network” was on how bots can abuse trending topics, hashtags, and memes to further their goals. Specifically, Sarita et al. follow the life cycle of a specific endogenous meme, #robotpickuplines. While not entirely on topic, it still shows some further useful insight into yet another method which bots can use. Then Yardi et al. started with the collection of the first degree network, which contained 8,616 users, and collected statistics to show the number of directed edges, unique followers, and unique friends. The authors achieved their goal of simply showing some of the properties of bots within the lifecycle of an endogenous meme. I would love to do this research again to see how it holds up today. I imagine it would be far more difficult to detect bots this way in the current state of Twitter, as I doubt there would ever be any lag time between a meme becoming a trending topic, and bots joining in on it. I think it would be a very fuzzy line in regards to trying to detect who is legitimate and who is a bot. I’m sure it would have interesting results, regardless. Overall this was a good paper, but not entirely on topic. It still provides useful insight into yet another method of evasion that a bot can utilize.

Wang’s work on “Don’t Follow Me: Spam Detection in Twitter” is on yet another machine learning approach for spam detection. The authors worked with a dataset of ‘around 25k users, 500k tweets, and 49m follower/friend relationships.’ They utilized and analyzed the performance of a Bayesian classification system. Interestingly enough, they
claim this to be the first effort to automatically detect spam on Twitter. Naturally, this will date their work, but not necessarily invalidate it. Since this is dated, and possibly one of the first bot analysis performed on Twitter, it has a simple approach. However, some aspects of this are similar to my own design. They look at simple attributes such as: replies, mentions, URLs, and repeat tweets. They show that a naive Bayesian classifier was the superior method tested, which is promising as I intend to use a Bayesian classifier, although for a different metric in my own work. Overall, this was an interesting paper, primarily due to the fact that they claim to be the first to work on bot detection on Twitter. Naturally, this work is dated because of this. I would not repeat this research, simply because I would rather repeat the similar works of others, with more advanced methods. This work differs from mine, as I do not intend to use any machine learning techniques, and I intend to analyze many more features than their work.

“Detecting Spammers on Twitter” by Benevenuto et al. [17] work with a dataset consisting of over 54 million users, and almost 1.8 billion tweets. This data was collected starting in August 2009. The authors follow the often used approach of training a machine learning framework via supervised training. They used the also often used attributes consisting of, but not limited to: friend-follower ratio, tweet ratio, time between tweet, tweets per day, etc. At this point, this is very similar to work done by several other previous papers reviewed. As with the others, this differs from my work in that I will not be using any machine learning frameworks, and I will be looking at a smaller subset of attributes than the authors [23]. I would not repeat this work, only because there are several similar works that are very similar, but slightly refined, that I would spend time on. Since this work is so similar to several others, I will not detail it any further [4,6,8,18,19,20,23].

“Robust Features for Detecting Evasive Spammers in Twitter” by Karim et al. [18] is on yet another graph based approach for detecting bots. However, this effort brings in a new angle, which is comparing the semantic meaning between tweets. As bots evolve, some may use evasion methods such as tweeting different words, but which mean the same
thing. They explore the use of several machine learning classifiers, and work with a dataset consisting of 750 users. The author employs clever usage of a lexical database to more accurately compare tweets. This is done by basically looking up synonyms for words within tweets, so that if two tweets use different words, but mean the same thing, they can more accurately be classified as similar. They also combine this with several methods often used, such as: friend-follower ratio, number of followers, number of friends, reply ratio, retweet ratio, etc. They also run several experiments using different profile attributes, content attributes, network attributes, and ‘neighbor’ attributes. Overall, this shows a lot of good statistics showing which attributes are the most important to look at. This will certainly be useful if I choose to add weighting to my current metrics. If given the chance, I would repeat this work, just to validate some of the results. I think it would be interesting to compare the individual attributes compared against some of my own.

“CATS: Characterizing Automation of Twitter Spammers” by Amleshwaram et al. [19] propose a new set of features to be used for feature detection (in 2013). They claim a high accuracy, with a minimal number of tweets per user (90% accuracy with only 5 tweets). The authors worked with two data sets: the first consisting of 500,000 users, and 14 million tweets from April 2010 - July 2010. The second consisted of 110,789 Twitter accounts, and 2.27 million tweets from November 2011 to January 2012. As with many other approaches, they are calculating statistics such as: number of mentions, number of hashtags, tweet times, URLs posted, domain names, etc. However, it is worth noting that they are one of the few approaches that are specifically looking for tweet text similarity. This is quite close to my approach, but unfortunately they weight this metric very low, in favor of the ever so common ‘friend-follower’ ratio used in graph/network based analysis. The authors claim a high accuracy percentage. They further their work by clustering bots based on the URLs included in their tweets. Overall, this paper did not particularly provide any new, useful, or unique information beyond the other papers reviewed. It did contain yet another list of possible metrics to be used, most of which overlap with several other research papers. I would not repeat this research only because I would rather do others that are very similar, but more recent. This differs from my work because I will be using fewer
attributes, none of which are graph based. Mine will also have the possibility of running in real-time or near real-time, as mine won’t need to reach out to get profile information for each user.

“Detecting Spam and Promoting Campaigns in the Twitter Social Network” by Zhang et al. [20], as many others, seems to take a basic approach to detecting bots. Their approach appears to be a subset of feature detection used by many others. They primarily focus on URL, timestamps, and relationships between users. They also employ some similarity metrics in regards to the URLs contained in tweets. As apparently using SVMs is the de facto standard for conducting research on bots within Twitter, they use this as well. This one goes a little further by trying to classify ‘groups’ of bots, or campaigns, although it primarily relies on URL similarity to do this. In the current state of Twitter, this would no longer work. Not only do bots use URL shorteners to obfuscate destinations, but Twitter itself automatically shortens URL with its own URL shortener. Overall, this paper is good. It adds reinforcement to the works of others. I would do this research again, if given the chance, because I suspect it would not work in the current state of Twitter. As this paper is short, and is very similar to many others, my review will be short as well.

“Detecting Social Spam Campaigns on Twitter” by Chu et al. [21] is on topic, but limited in scope. They use a dataset consisting of 50 million tweets, posted by 22 million users. This data was collected for three months during 2011. The authors note themselves that this paper is limited to only detecting spam campaigns, where spam is defined as “malicious, phishing, or scam.” I would argue that this is a poor definition, as it leaves out large scale spam campaigns whose purpose is only to sway public opinion. I would also argue that this is a far more important issue to deal with. The authors put an emphasis on detecting bots by analyzing the URL posted by accounts. They address such issues as URL redirection, final URL, URL frequency, and URL similarity, among other metrics. They then use ‘final URL’ to cluster accounts based on similarity. They also analyze ‘activity over time’ within a campaign, noting that bots are more likely to tweet in ‘bursts.’ As with
many other researchers, they look at content attributes (they call it tweet-level features), spam content properties, network attributes, and user information. These attributes are largely in line with the majority of other research works. As always, they put this together using a machine learning classification system. They compare different algorithms, and get about the same results as several other papers. Overall, this was a good paper. This covered and combined more issues than most other papers by going into depth with URL detection and classification, semantic similarity, and of course the list of other attributes that are fairly common between research works. I would like to do this research work again, since it covers so many different issues. I would be interested to see how it holds up today, as it is now four years old. I would also like to add some of my own attributes, and remove some of theirs to see how well it would perform. This work differs from mine as I won’t be taking a machine learning approach, and I will be looking at a different set of attributes.

“Detecting Spam on Twitter via Message-Passing Based on Retweet-Relation” by Chen et al. [22] is in line with many other works, but contributes a unique approach by using graph analysis to determine the flow of a message between accounts. They worked with a dataset of approximately 5 million tweets collected from May 2014 to July 2014. The authors, however, accurately acknowledge that most other efforts are based on finding individual bot accounts. Their approach is designed to detect groups of bots. The paper isn’t quite clear on how they construct their “message-passing graph,” but it alludes to creating a graph of retweets based on time. Although the paper lacks detail, they presumably compare this to what a non-spam retweet graph looks like. They do this by calculating various statistics on the graph structure. Overall, this is a good paper due to its unique approach. I think their approach only detects a very small subset of bots/spammers though, and is likely looking at the wrong information. This may have been a good way to detect bots when the paper was published, but I doubt it would still hold up. Consider a group of bots who don’t retweet, but still act as one when posting messages? What if they don’t communicate with each other at all? I think this method is easily defeated. I would consider repeating this work, only to verify that it would not work well in the current state of Twitter. This work differs from mine because I will be detecting groups of bots based
on simple attributes, not network analysis. Their method, while near real-time, would still be slower than my approach. However, I acknowledge that they are targeting a different subset of bots than myself.

“Spam Detection on Twitter Using Traditional Classifiers” by McCord and Chuah [23] is extremely similar to many others, and due to its age, is likely a precursor to many of the papers published in 2012, and 2013. It features spam detection based on a very small subset of features such as number of URLs, number of replies/mentions, retweets/tweet length, and hashtags. This is a much smaller set of attributes compared to the majority of other works. They calculate statistics showing the range of followers, calculated reputation, and follower’s reputation. Their graphs show that while spammers (used to) fall within these ranges, so did many legitimate users. They further show that the average posts per user is largely similar to legitimate users, as is the average number of URLs. The paper finishes by comparing various classification methods against each other to determine which has the highest accuracy. As many other papers, they compared: RandForest, SMO, NaiveBayes, and KNN. Their results differ slightly, with RandForest being the best and several others had NaiveBayes being the best. Overall, I think this was a good precursor to many other research papers. However, I think the methods used are all outdated. Twitter has since changed many of its policies, such as limiting the number of followers and has better detection algorithms. Conversely, bots have grown more sophisticated as well, using techniques such as URL obfuscation, inflating friend-follower ratio by friending other bots, and randomizing tweets slightly so that they differ from other bots. I would not repeat this research for the reasons mentioned above. It would be highly unlikely that it would still work in the current state of Twitter. This work differs from mine, as I intend to user a slightly larger, and completely different set of attributes to analyze.

“Fake Account Detection in Twitter Based on Minimum Weighted Feature Set” by Ahmed et al. [60], as many before it, utilizes a machine learning approach. They differ in the attributes looked at though, and don’t largely overlap with the works of any others.
They use some interesting metrics such as “Account has at least 30 followers,” or “Account has logged in from an iPhone.” They use a very small, tailored dataset for analysis. This dataset consists of 1,481 human accounts and 3,000 fake accounts. No date was given on when this was collected. They certainly explored some interesting attributes to use for analysis, and is one of the more original approaches that has been reviewed. They still used the often used machine learning approach. The dataset seems quite tailored for their usage, and differs from the “real world.” They have shed light on some new attributes that can be used for bot determination, but an analysis on a real world dataset would have been more conclusive. Possibly more importantly is their analysis of the attributes used by others’ works. This paper is on topic, and the authors achieve what they set out to do. After stating some of the attributes used by others, they show that this list can be reduced to at least 10 of these. They then further reduce this to 7 attributes, and claim that bots can be detected using only this set of attributes. The authors note that they need to prepare a dataset to prove this claim.

“The Case for Repeated Research in Operating Systems” by Matthews [24] claims that repeated research is beneficial not only to the research community, but can also be very beneficial to students. It was an easy read, very straightforward, and short. It also lays out a good, general guide for conducting repeated research. It is a well written, simple, understandable read that makes a good case for repeated research. I particularly like the case for involving students earlier on in research. Repeating a well written, simple experiment is an easy way to get introduced to, and experience with, researching. The only thing to possibly take issue with is the title, as I think a more fitting title may be something along the lines of “The Case for Repeated Research in Computer Science,” or even simply “The Case for Repeated Research.” This paper is on topic for any major research paper, such as a thesis, and is worthy of being included, albeit not bot or Twitter related. Since we all stand on the shoulders of giants in these kinds of works, we are inherently also repeating simpler, more fundamental works.
In summation to all of the research that has been reviewed, most approaches are: outdated, limited in scope, or unnecessarily complex. The majority of research is largely overlapping in regards to what pieces of information are considered when determining whether an account is a bot or legitimate. The most often used attributes are: friends, followers, friend-follower ratio, URLs, and network analysis. This is fine, and undeniably serves its purpose, and I don’t doubt the validity of their research. However, I argue that there are bots that none of these methods may detect.
Chapter 3

Approach

3.1 Equipment

With the given budget constraints, this work was limited to the following equipment. The primary machine used for the majority of programming and data analysis contains the following specifications:

<table>
<thead>
<tr>
<th>Specification</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU Model</strong></td>
<td>Intel Core i7-3770k</td>
</tr>
<tr>
<td><strong>CPU Architecture</strong></td>
<td>64-bit</td>
</tr>
<tr>
<td><strong>CPU Cores</strong></td>
<td>4</td>
</tr>
<tr>
<td><strong>CPU Threads</strong></td>
<td>8</td>
</tr>
<tr>
<td><strong>CPU Clock</strong></td>
<td>3.9 GHz per core</td>
</tr>
<tr>
<td><strong>RAM Size</strong></td>
<td>8 GB</td>
</tr>
<tr>
<td><strong>RAM Speed</strong></td>
<td>1866 MHz</td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td>Windows 10 Pro Insider Edition</td>
</tr>
<tr>
<td><strong>HDD 1</strong></td>
<td>64 GB SSD (OS)</td>
</tr>
<tr>
<td><strong>HDD 2</strong></td>
<td>1 TB (Datasets)</td>
</tr>
</tbody>
</table>

*Table 1: Specifications of desktop*
My secondary machine used occasionally for programming and small dataset analysis contains the following specifications:

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Model</td>
<td>Intel Core i3-5010U</td>
</tr>
<tr>
<td>CPU Architecture</td>
<td>64-bit</td>
</tr>
<tr>
<td>CPU Cores</td>
<td>2</td>
</tr>
<tr>
<td>CPU Threads</td>
<td>4</td>
</tr>
<tr>
<td>CPU Clock</td>
<td>2.1 GHz per core</td>
</tr>
<tr>
<td>RAM Size</td>
<td>4 GB</td>
</tr>
<tr>
<td>RAM Speed</td>
<td>1600 MHz</td>
</tr>
<tr>
<td>OS</td>
<td>Windows 10 Home</td>
</tr>
<tr>
<td>HDD</td>
<td>500 GB</td>
</tr>
</tbody>
</table>

Table 2: Specifications of laptop

3.2 Datasets Used

As previously detailed in 1.11, all data was acquired from Twitter’s public API. There are many datasets, but I will detail the majority of them. The names are fairly arbitrary, and not always descriptive, but I will keep them as I have them to avoid confusion on my part. Any pertinent information is included below each table. There is a notable amount of political-related datasets. This is due to its availability from my work on [35,36,37], and more importantly because it is a topic that is more likely than most to contain the kinds of bots I am attempting to detect [29, 30, 55]. At the time of this writing, with the looming 2016 US election, this is a prime time to be analyzing this kind of data.
Below is a comma separated set of keywords that were used to filter upon. Datasets that used this set will say so in their description. This will be referred to as the “politics keywords.”


<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>20k</td>
<td>13,204</td>
<td>66.2MB</td>
<td>26m 43s</td>
</tr>
</tbody>
</table>

*Table 3: “20k” dataset statistics*

This dataset was obtained from Twitter’s keyword filtered stream. This used the politics keywords as filters, and the language filtered by English. The first tweet was on Sun Jan 31 02:41:47 +0000 2016, and the last was on Sun Jan 31 03:08:31 +0000 2016.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>1k</td>
<td>1,000</td>
<td>4.8MB</td>
<td>51s</td>
</tr>
</tbody>
</table>

*Table 4: “1k” dataset statistics*
This dataset is simply a subset of “20k,” and because of it’s even and small amount, was used often for program testing, and for timing statistics. As such, it also used the politics keywords as filters, and the language filtered by English. The first tweet was on Sun Jan 31 02:41:47 +0000 2016, and the last was on Sun Jan 31 02:42:39 +0000 2016.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>jadine</td>
<td>1,832</td>
<td>8.9MB</td>
<td>13h 50m 27s</td>
</tr>
</tbody>
</table>

*Table 5: “jadine” dataset statistics*

This set was extracted from a subset of the “politics” set. It was extracted based on if it contained the string “jadine.” This was used as a test set, as it is almost entirely spam. The first tweet was on Sat Oct 03 00:18:35 +0000 2015, and the last was on Sat Oct 03 14:09:02 +0000 2015.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>isis</td>
<td>11,038</td>
<td>62.2MB</td>
<td>22m 22s</td>
</tr>
</tbody>
</table>

*Table 6: “isis” dataset statistics*

This set was from Twitter's keyword filtered stream, which only filtered on the keyword “isis”. The first tweet was on Wed Feb 04 07:18:17 +0000 2015 and the last was on Wed Feb 04 07:40:40 +0000 2015.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>politics_250k</td>
<td>249,999</td>
<td>1.1GB</td>
<td>4h 45m 15s</td>
</tr>
</tbody>
</table>

*Table 7: “politics_250k” dataset statistics*

This dataset is the first 250,000 tweets (1 invalid one is subtracted from count) from the “politics” dataset. This set was filtered on the politics keywords, and the language
filtered by English. The first tweet was on Thu Oct 08 18:07:33 +0000 2015, and the last was on Thu Oct 08 22:52:49 +0000 2015.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>red4tum</td>
<td>3,536</td>
<td>13MB</td>
<td>25m 1s</td>
</tr>
</tbody>
</table>

*Table 8: "red4tum" dataset statistics*

This was one of the first datasets I ever collected, simply as a test. It was filtered on the keywords “reddit”, “4chan”, and “tumblr”. The first tweet was on Mon Feb 02 07:54:40 +0000 2015, and the last was on Mon Feb 02 08:19:42 +0000 2015.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample1</td>
<td>52,741</td>
<td>186.7MB</td>
<td>18m 59s</td>
</tr>
</tbody>
</table>

*Table 9: "sample1" dataset statistics*

This dataset was relatively recent, and used as a control for testing. This is an unfiltered set of tweets, taken from Twitter’s Sample stream.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>pure_spam</td>
<td>86</td>
<td>244.3KB</td>
<td>1s</td>
</tr>
</tbody>
</table>

*Table 10: "pure_spam" dataset statistics*

This is a very small dataset, used for testing certain functionality of my program. As the name implies, it contains only spam. It is just a subset of the “jadine” dataset, and thus inherits its properties. The first tweet was on Sun Oct 04 01:17:22 +0000 2015 and the last was on Sun Oct 04 01:17:24 +0000 2015.
<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>politics_10k</td>
<td>9,999</td>
<td>45.1MB</td>
<td>10m 41s</td>
</tr>
</tbody>
</table>

*Table 11: “politics_10k” dataset statistics*

This set was used for more rapid testing of the “politics” dataset. This is a subset of “politics_250k”, and thus inherits its properties. The first tweet was on Thu Oct 08 18:07:33 +0000 2015 and the last was on Thu Oct 08 18:18:15 +0000 2015.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>superbowl1</td>
<td>13,840</td>
<td>53MB</td>
<td>33m 56s</td>
</tr>
</tbody>
</table>

*Table 12: “superbowl1” dataset statistics*

This dataset was collected in early 2015, during the Superbowl, of which analysis was later performed on. This dataset was filtered on the keywords: “super,” “bowl,” “superbowl,” and the language as English. The first tweet was on Sun Feb 01 07:35:29 +0000 2015 and the last was on Sun Feb 01 08:09:26 +0000 2015.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>superbowlive</td>
<td>59,804</td>
<td>237.3MB</td>
<td>5m 15s</td>
</tr>
</tbody>
</table>

*Table 13: “superbowlive” dataset statistics*

This dataset was filtered on the same criteria as “superbowl1.” The first tweet was on Sun Feb 01 23:02:25 +0000 2015 and the last was on Sun Feb 01 23:07:41 +0000 2015.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>2_clean_25k</td>
<td>25,000</td>
<td>100.8MB</td>
<td>29m 11s</td>
</tr>
</tbody>
</table>

*Table 14: “2_clean_25k” dataset statistics*
This is another subset of the “politics” dataset, and as such, inherits its properties. This is another set small enough to work with, from a different section of the “politics” dataset. The first tweet was on Sun Oct 04 01:07:04 +0000 2015 and the last was on Sun Oct 04 01:36:15 +0000 2015.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>ny_primary_250k_1</td>
<td>248,618</td>
<td>1.2GB</td>
<td>3h 46m 21s</td>
</tr>
</tbody>
</table>

Table 15: "ny_primary_250k_1" dataset statistics

This is one of the most recent datasets. This was collected on New York’s voting day for the primary election (4/19/2016). This dataset was filtered on the keywords “vote,” “primaryday,” and “#primaryday.” The phrase “[#]PrimaryDay” was the current trending hashtag at the time of collection. This is part one of two datasets collected on this filter, on that day. The first tweet was on Tue Apr 19 16:30:27 +0000 2016 and the last was on Tue Apr 19 20:16:49 +0000 2016.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>ny_primary_250k_2</td>
<td>248,497</td>
<td>1.3GB</td>
<td>3h 47m 36s</td>
</tr>
</tbody>
</table>

Table 16: "ny_primary_250k_2" dataset statistics

This is the second dataset to the dataset above. It was filtered on the same keywords, on the same day. This chronologically follows immediately after the first. The first tweet was on Tue Apr 19 20:17:34 +0000 2016 and the last was on Wed Apr 20 00:05:11 +0000 2016.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>snl</td>
<td>4,113</td>
<td>23.3MB</td>
<td>32m 9s</td>
</tr>
</tbody>
</table>

Table 17: "snl" dataset statistics
This dataset was filtered on the keywords “snl40” and the language as English. This was collected when that keyword was trending during the 40th anniversary performance of Saturday Night Live (a popular TV show). The first tweet was collected on Mon Feb 16 21:56:08 +0000 2015 and the last was on Mon Feb 16 22:28:17 +0000 2015.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Tweets</th>
<th>File Size</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>politics</td>
<td>9,985,185</td>
<td>48.5GB</td>
<td>16d 15h 14m 55s</td>
</tr>
</tbody>
</table>

Table 18: "politics" dataset statistics

This is the set from which most politic-related subsets originate from. This is my largest dataset, and as such, it is currently too large to analyze on a single machine. This is only due to the program currently reading in entire datasets to memory before processing. It is certainly not required, but it is beyond the scope of this effort to provide a full real-time implementation. It is beyond the scope of this project to optimize my program as such; however, this dataset is worth mentioning since several others are subsets of it. This, of course, was filtered on the politics keywords, and the language as English. The first tweet was on Sat Oct 03 00:18:30 +0000 2015, and the last was on Mon Oct 19 15:32:25 +0000 2015.

There are other datasets, but for the sake of brevity, I will leave it at this. This is primarily to show that there is sufficient variance in datasets analyzed. Between these datasets, there are some that were:

- Collected over a year apart
- Filtered on language
- Filtered on keywords
- Unfiltered
It is worth noting that Twitter, unlike most other social media networks, provides a “random sample” of tweets in its Sample stream [52]. As stated by an employee of Twitter:

“The sampling is based on a hash that is completely agnostic to any substantive metadata, so it should be a fair and proportional representation across all cross-sections, including whether there are links, whether there are hashtags, @-mentions, @-replies, what app/client generated the Tweet, etc.[53]”

However, as shown by Morstatter et al., this may not be entirely true. Regardless of who is correct, it should be sufficiently random for these datasets to be considered valid and of quality in this context. This is in comparison to other OSNs that will likely only allow access to data within your own personal network, or geographical region, which would certainly not be satisfactory in regards to data quality and data validity, for this research. In summation, I intend to demonstrate that my approach will work on any dataset, on a sliding scale from a random sample to a filtered dataset.

### 3.3 Data Treatment & Arrangement

All data is pre-processed before any analysis to remove any malformed, or otherwise any non-empty lines that don’t contain a valid tweet. This is always a very small percentage, but they must be removed. Some datasets are already pre-processed, and some are pre-processed on the fly.

All datasets mentioned, with the exception of “jadine” and “pure_spam” are in the order as given by Twitter, and thus in chronological order.
3.4 Tools & Libraries

As this is a simple approach, there are only 3 libraries used, beyond regular, often used libraries (ex: time, sys, json, math, unicodedata).

The first library used is Genderizer. Gender determination is done by utilizing, optionally, two pieces of data: Name, and Text. If a name is known to be typically strictly for one gender, then that determination is final, and no further analysis is performed. When a name isn’t clearly associated with a specific gender, then the text is analyzed using a Naive Bayesian Classification approach for determining the probability that any given text is associated with a gender. Given its heuristic nature, this also results in indeterminate decisions. This algorithm and approach implemented was derived from Mustafa Atik, and Nejdet Yucesoy’s work on ‘Genderizer’ [49]. This was used to add another attribute to compare against. It is not a “high impact” attribute compared to others, but it helps reinforce a decision made by the scoring system. For example, it would be uncommon for N tweets in a row to all be the same gender. However, since this is a binary value, it is for all intents and purposes likely that N/2 will be the same. This is why it is not a “high impact” attribute, and thusly it is given a reduced weight in regards to determining whether a tweet is from a bot or not.

Next, is the SequenceMatcher module within the ‘difflib’ library. This is a default library provided with any installation of Python. It’s worth discussing, as it plays an important role directly with analysis, rather than other functions such as “sys” which simply help with reading files from disk for example.

“This is a flexible class for comparing pairs of sequences of any type, so long as the sequence elements are hashable. The basic algorithm predates, and is a little fancier than, an algorithm published in the late 1980’s by Ratcliff and Obershemp under the hyperbolic name “gestalt pattern matching.” The idea is to find the longest contiguous matching subsequence that contains no “junk” elements (the Ratcliff and Obershemp algorithm doesn’t address junk). The same idea is then
applied recursively to the pieces of the sequences to the left and to the right of the matching subsequence. This does not yield minimal edit sequences, but does tend to yield matches that “look right” to people.

Timing: The basic Ratcliff-Obershemp algorithm is cubic time in the worst case and quadratic time in the expected case. SequenceMatcher is quadratic time for the worst case and has expected-case behavior dependent in a complicated way on how many elements the sequences have in common; best case time is linear. [50]”

This library plays a crucial role in determining whether a tweet is from a bot or not. This would be one of the “high impact” attributes when evaluating a tweet. This is used in a few places. Intuitively it is used to compute the difference between the text contained within two tweets, and it is also used to compare profile descriptions. Since bots most often will tweet identical or similar messages, this library will shed light on how similar or dissimilar two messages are.

Lastly, is a library called “TextBlob.” This library is utilized for its sentiment analysis functions. TextBlob is a library that provides a simple interface for performing various natural language processing (NLP) functions. While it offers various functionality, I only utilized the sentiment analysis function.

“TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. [51]”

This library performs sentiment analysis by using a Naive Bayesian classifier. This classifier is trained on movie reviews to create its corpus. This is also used to calculate a “high impact” attribute, as it has been shown that bots tend to group around a specific sentiment value [9].
3.5 Method

The research methodology used is quantitative analysis [59]. Specifically, the Experimental research method [57]. This was chosen due to its high evidence of effectiveness [58], and because of its emphasis on generating and using analyzable data. As stated by [57]:

“When we talk about ‘scientific research methods,’ this is what most people immediately think of, because it passes all of the definitions of ‘true science’.”

This approach, as most others, looks at analyzing ‘simple’ attributes. I used the word ‘simple,’ not so much in reference to the attribute, but the comparisons made upon them. While most researchers have utilized machine learning algorithms such as neural networks and SVMs, most of the attributes used in this effort are analyzed by performing a simple comparison to determine whether they are the same as that tweets neighbors’ attributes. What makes it stand apart is not the attributes used, or even the specific combination, but rather that this approach does not analyze the data, history, and network of an account. Instead, it compares most attributes against the nearest N tweets as they appear in the Twitter live stream (chronological order). This approach has both ups and downs. It has the advantage of only requiring the data of a single tweet to make a determination, as the only data considered is that of the nearest N tweets on a chronological scale. As this approach is designed for detecting botnets and spam campaigns, it will not detect a lone bot acting on its own.

This approach has massive advantages by requiring so little data, and also not being computationally difficult. This also allows this program to analyze tweets in real-time. The determination of whether a user is a bot or not is done by a scoring system. For each attribute below, with the exception of entropy and sentiment, the score for each is on a scale of 0 to N, where N is the number of neighboring tweets that are compared against. As there are 10 attributes, the maximum score is N * 10. With the addition of sentiment and entropy, the total maximum score is: (N * 10) + SentimentScore + EntropyScore. If a
bot meets a certain percentage of this, they are classified as a bot. This will be detailed further.

Entropy:

Entropy is a telling feature, as shown by [1] and [2]. Entropy is the calculation of how many bits would be required to represent some piece of data.

\[
H(X) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i)
\]

*Equation 1: Shannon's entropy formula*

For example, the letter “a” would have an entropy of 1, as it would only take 1 bit to represent this. The sequence “abcd” would require 2 bits to encode, as there are 4 possible values. Since tweets are limited to 140 characters, the maximum entropy possible would be \(2^{7.129283016945}\). \(2^7\) would only be able to represent 128 different characters, and \(2^8\) would be too large by representing 256. Bots tend not to use as many different characters as humans, as stated and shown in [1] and [2]. This is one of few attributes that is not compared against the nearest N tweets. Since it is shown that bots tend to have an entropy of lower than humans. From personal analysis in my works on [36], I have determined this to be around ~6. However, this is set to a user-changeable variable in my approach, should a lower threshold be desired. This is one of two metrics, which is not on a scale of 0 to N, as it isn’t compared to neighboring tweets. This number is set as a static variable, between 0 and N. During testing, this has been set to N * 0.5.

Similarity:

There are two metrics calculated in regards to similarity. The first, intuitively, being a summation of the similarity score as determined by [50] on the nearest N tweets. As the similarity score as measured between two pieces of text is on a scale of 0-1, the total score will be on a scale of 0 to N. The second score is also between 0 and N, which represent how many tweets had a similarity score above the user set threshold. This is also a telling
feature as shown by [9]. As mentioned, this is compared to neighboring tweets rather than a list of a user's own tweets, as done by others work.

Sentiment:

This is the second attribute of two, which is not compared against neighboring tweets. As such, it’s score is set as a static variable between 0 and N. Sentiment is calculated using a Naive Bayes Classification algorithm. This uses Bayes Theorem to determine the probability of an event. In this case, it’s used in comparison to a corpus of manually classified pieces of text to determine the probability that a given piece of text is either positive or negative in sentiment.

\[
P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)}
\]

*Equation 2: Baye's theorem*

“P(A) and P(B) are the probabilities of A and B without regard to each other. P(A \mid B), a conditional probability, is the probability of observing event A given that B is true. P(B \mid A) is the probability of observing event B given that A is true [56].”

For testing, this has been set to N * 0.5. This score being added, is determined by meeting a threshold as set by a user.

Language, Gender, User Agent, Time Zone, Location, URL:

These attributes, are all extracted from a tweet, and simply compared to each neighboring N tweets. If any individual one matches, that tweets score is incremented by one. However, as tweets can be filtered by language, this should be set to 0 if the dataset was filtered this way. The program will automatically rebalance the scoring system if this is the case. It is also worth mentioning that while many, if not most, approaches analyze URLs, my approach only looks at a user's profile URL, as compared to a URL contained within a tweet. I do not analyze the URL itself, only if it is the same as neighboring user's profile URLs.
Description:
As most of the others, this score is on a scale of 0 to N. However, this score, rather than having to match exactly as the section above, has to be above a similarity threshold. If the compared profile descriptions are above that threshold, then the score is incremented by one.

Time Difference:
Similar to description, this is only incremented if the tweet it’s compared against, has a similarity score against that tweet that is above the set threshold. The example below shows a representation of how this works.

In this example, we see that Tweet2 and Tweet3 are within each other’s Time Difference threshold. Many tweets are sent every second, so this alone would not be of any value. However, if Tweet2 and Tweet3’s text are above the similarity threshold, then there is a high probability that this is automated behavior, as a human typically cannot write and send a similar or the same tweet from the same, or two different accounts within this amount of time.
An example scoring system, on a dataset that was filtered by language:

\[
N = 40 \quad // \text{Compared against nearest 40 tweets} \\
\text{SentimentScore} = N \times 0.5 \quad // \text{Score to be added if sentiment is above threshold} \\
\text{EntropyScore} = N \times 0.5 \quad // \text{Score to be added if entropy is below threshold} \\
\text{Total\_possible\_score} = (N \times 9) + N \times 0.5 + N \times 0.5 \\
//\text{Total\_possible\_score = 380}
\]

So, the determination could then be made by:

```python
if (tweetScore/maxScore) > threshold:
    User = bot
```

In the results section, specific thresholds for different attributes and final determination will be analyzed and discussed further.

As could be assumed, this approach uses a scoring system. First we must define thresholds for particular attributes which require them:

```python
# This must be an even number. Memory needed for holding tweets ==
# simCompareLimit * 5075 bytes. This is how many neighboring tweets are
# compared against.
N = 30
# Scale of 0-1, threshold for text similarity
textSimThresh = 0.6
# Threshold used for time difference in similar tweets
timeDiffThresh = 4000
# Threshold, for entropy: if entropy < entThres, add points to high_score
entThresh = 5.5
# Threshold for sentiment: if sentiment > sentThresh, add points to
# high_score
sentThresh = 0.5
# Percentage of total_score that is needed to be called a bot
highScoreThresh = 0.35
```
Then we define the static points given for entropy and sentiment, as they are not compared against neighboring tweets:

<table>
<thead>
<tr>
<th>bonus_for_sent = 10</th>
<th># bonus points are added if sentiment &gt; sentThresh</th>
</tr>
</thead>
<tbody>
<tr>
<td>bonus_for_ent = 10</td>
<td># bonus points are added if entropy &lt; entThresh</td>
</tr>
</tbody>
</table>

Next, we define multipliers. All attributes are certainly not equal, and most are only “helpers,” to help avoid false positives. The high impact attributes are of course the most telling attributes, but to avoid the inevitable chance that a legitimate user is classified as a bot, we make sure that several other features are an exact match as well - or at least some of them. These multipliers are just an example, but my results will detail the success of some of these. The segregation between “high impact” and “other” multipliers is determined by the works of most papers cited in this project, who have shown that these attributes are the most telling. Similarity stands out with a multiplier of 2, as this is the feature most often associated with a bot.

```
########### MULTIPLIERS ##############
# High impact multipliers
simScoreMult = 2
sentimentMult = 1.2
entropyMult = 1.2
simCountMult = 1.2

# Other multipliers
locCountMult = 1
uaCountMult = 1
tzCountMult = 1
descCountMult = 1
urlCountMult = 1
timeDiffMult = 1
genderMult = 1
```
Now, we can define the maximum score as:

```python
# If we aren’t using ‘language’ (filtered dataset), adjust max score accordingly
if langCountMult == 0:
    total_score = ((N * 4) * 1.2) + (N * 5) + bonus_for_sent + bonus_for_ent
else:
    total_score = ((N * 4) * 1.2) + (N * 6) + bonus_for_sent + bonus_for_ent
```

Finally, we can compute all comparisons, and sum the scores for each attribute to create a score for a tweet. As stated:

```
if (tweetScore/total_score) > highScoreThresh:
    User is likely a bot
```

Now, an example full walkthrough of the calculation of the score of a tweet:

In the neighboring 40 tweets:

<table>
<thead>
<tr>
<th>How many have a text similarity greater than textSimThresh?</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>- For each of the above, how many were within 4 seconds?</td>
<td>16</td>
</tr>
<tr>
<td>What is the summation of the similarity score between the current tweet and the others?</td>
<td>24.9582</td>
</tr>
<tr>
<td>How many have the same language?</td>
<td>21</td>
</tr>
<tr>
<td>How many have the same gender?</td>
<td>16</td>
</tr>
</tbody>
</table>
How many have the same user agent? | 21
---|---
How many have the same time zone? | 32
How many have the same location? | 27
How many have the same profile URL? | 12
How many have the same profile description? | 15
Is the entropy lower than the entropy threshold? | Yes
Is the sentiment higher than the sentiment threshold? | No

Table 19: Example of a bot’s score

Total tweet score (adding up all scores, and respecting multipliers):

\[
(23 \times 1.2) + (24.9582 \times 1.2) + 15 + 12 + 27 + 32 + 21 + 16 + 21 + 16 \\
= 217.54984
\]

Equation 3: Step 1 of a bot’s score

Since the entropy threshold was met, we add an additional:

\[
= 217.54984 + (40 \times 0.5) \times 1.2 \\
= 241.54984
\]

Equation 4: Step 2 of a bot’s score

This is the total score for this tweet. Now, does it meet our bot threshold?

First, we need the total maximum score:

\[
\text{total\_score} = ((40 \times 4) \times 1.2) + (40 \times 6) + \text{bonus\_for\_sent} \times 1.2 + \text{bonus\_for\_ent} \times 1.2 \\
\text{total\_score} = 192 + 240 + 24 + 24 \\
\text{total\_score} = 480
\]

Equation 5: Step 3 of a bot’s score
Now, the high score as a percent is:

\[
= \frac{241.54984}{480} \\
= 0.50322
\]

*Equation 6: Step 4 of a bot’s score*

Lastly, as our threshold was set to: 0.35, this tweet would be classified as a bot. This should be intuitive, due to the high number of similarities of most attributes given in this example.
Chapter 4: Results

4.1 Thresholds

To begin analysis, we must first select our thresholds. Some ultimately are subjective, such as text similarity. For example, how would you define when two pieces of text are similar? However, we can at least gain insight in most thresholds. First, we’ll use the following thresholds below, which were some preliminary numbers which proved successful. The first analysis will be to determine how many neighbors is optimal to compare against. The following thresholds are only for the purpose of keeping all other numbers the same when comparing, and are not used in the final analysis.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text similarity threshold</td>
<td>0.6</td>
</tr>
<tr>
<td>Time difference threshold</td>
<td>4000 ms</td>
</tr>
<tr>
<td>Entropy threshold</td>
<td>5.5</td>
</tr>
<tr>
<td>Sentiment threshold</td>
<td>0.5</td>
</tr>
<tr>
<td>High score threshold</td>
<td>0.35</td>
</tr>
</tbody>
</table>

*Table 20: Example thresholds*
The number of neighbors to compare against needs to walk a fine line between accuracy and performance.

### 4.1.1 Neighbors

The first data set analyzed was the “1k” dataset. As detailed in Chapter 3, this was taken from Twitter’s sample stream, and has no filters.

<table>
<thead>
<tr>
<th># of neighbors compared</th>
<th>Tweets per second</th>
<th>Time taken</th>
<th>Bots detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>98.6</td>
<td>10s</td>
<td>39</td>
</tr>
<tr>
<td>20</td>
<td>46.9523</td>
<td>21s</td>
<td>41</td>
</tr>
<tr>
<td>30</td>
<td>30.8125</td>
<td>32s</td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td>23.4761</td>
<td>42s</td>
<td>39</td>
</tr>
<tr>
<td>50</td>
<td>18.6037</td>
<td>53s</td>
<td>40</td>
</tr>
<tr>
<td>60</td>
<td>15.4062</td>
<td>1m 4s</td>
<td>40</td>
</tr>
<tr>
<td>70</td>
<td>11.0786</td>
<td>1m 29s</td>
<td>38</td>
</tr>
<tr>
<td>80</td>
<td>9.4807</td>
<td>1m 44s</td>
<td>37</td>
</tr>
<tr>
<td>90</td>
<td>9.6667</td>
<td>1m 42s</td>
<td>34</td>
</tr>
<tr>
<td>100</td>
<td>9.1296</td>
<td>1m 48s</td>
<td>18</td>
</tr>
</tbody>
</table>

*Table 21: Number of neighbors compared for “1k”*

The results were mostly as expected. When the number of neighbors compared goes above 60, there is a decline in accuracy. Otherwise, the number of bots detected is approximately the same. It was expected that when the number of neighbors compared was low, this would
decrease accuracy as there wouldn’t be as many comparisons, but this was not the case. A manual check did not reveal any false positives.

The second dataset analyzed was “red4tum”, which was a dataset filtered on the keywords “reddit”, “4chan”, and “tumblr.” It was limited to English tweets only.

<table>
<thead>
<tr>
<th># of neighbors compared</th>
<th>Tweets per second</th>
<th>Time taken</th>
<th>Bots detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>139.826</td>
<td>23s</td>
<td>47</td>
</tr>
<tr>
<td>20</td>
<td>73.0909</td>
<td>44s</td>
<td>19</td>
</tr>
<tr>
<td>30</td>
<td>48.7272</td>
<td>1m 6s</td>
<td>16</td>
</tr>
<tr>
<td>40</td>
<td>32.4848</td>
<td>1m 29s</td>
<td>12</td>
</tr>
<tr>
<td>50</td>
<td>22.4895</td>
<td>2m 23s</td>
<td>10</td>
</tr>
<tr>
<td>60</td>
<td>18.9176</td>
<td>2m 50s</td>
<td>7</td>
</tr>
<tr>
<td>70</td>
<td>16.9263</td>
<td>3m 10s</td>
<td>4</td>
</tr>
<tr>
<td>80</td>
<td>14.9581</td>
<td>3m 35s</td>
<td>2</td>
</tr>
<tr>
<td>90</td>
<td>13.6271</td>
<td>3m 56s</td>
<td>2</td>
</tr>
<tr>
<td>100</td>
<td>12.6117</td>
<td>4m 15s</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 22: Number of neighbors compared for “red4tum"

These results were the most surprising. It was not expected that when the number of neighbors compared was low, there would be a dramatic increase in the number of bots detected. However, this also introduced 1 false positive when N = 10. I suspect the variance in bots detected has to do with the fact that this was a filtered dataset, and is more likely to capture bots that may be using the keywords it was filtered on, as opposed to a random sample of tweets. It is also worth noting that these were not popular keywords, relatively speaking, when this was collected.
The next dataset analyzed was the “politics10k” set. As detailed in Chapter 3, this was filtered on a list of keywords relevant to politics, and also the English language.

<table>
<thead>
<tr>
<th># of neighbors compared</th>
<th>Tweets per second</th>
<th>Time taken</th>
<th>Bots detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>65.6556</td>
<td>2m 31s</td>
<td>130</td>
</tr>
<tr>
<td>20</td>
<td>34.3044</td>
<td>4m 49s</td>
<td>125</td>
</tr>
<tr>
<td>30</td>
<td>24.299</td>
<td>6m 48s</td>
<td>119</td>
</tr>
<tr>
<td>40</td>
<td>21.0042</td>
<td>7m 52s</td>
<td>115</td>
</tr>
<tr>
<td>50</td>
<td>17.0343</td>
<td>9m 42s</td>
<td>111</td>
</tr>
<tr>
<td>60</td>
<td>14.3889</td>
<td>11m 29s</td>
<td>98</td>
</tr>
<tr>
<td>70</td>
<td>10.558</td>
<td>15m 39s</td>
<td>70</td>
</tr>
<tr>
<td>80</td>
<td>9.2654</td>
<td>17m 50s</td>
<td>58</td>
</tr>
<tr>
<td>90</td>
<td>8.4374</td>
<td>19m 35s</td>
<td>54</td>
</tr>
<tr>
<td>100</td>
<td>7.7272</td>
<td>21m 23s</td>
<td>45</td>
</tr>
</tbody>
</table>

*Table 23: Number of neighbors compared for “politics10k”*

This was closer to the expected results than “red4tum,” although it was still surprising that the number of bots detected decreases when the number of neighbors compared increases. When N=10, there were 3 false positives when compared to N=20. When N=20, there were 0 false positives when compared to N=30. This dataset is likely the most important to consider, as it was filtered on many popular keywords when collected. When a botnet tweets, since they use the same or similar message, it would be more likely that they would fall into a general, more popular topic such as politics.
Overall, I suspect “red4tum” may be an outlier in regards to the dramatic increase when N=10, due to the limited scope the dataset was filtered on. It would be more likely that similar tweets will be together. While N=10 did in fact detect over twice as many bots, this would have likely been detected if it was a larger (“faster”) dataset that was not so limited in scope, and if N was set a little higher.

The takeaways from this would be that N=20 appears to be the best compromise between speed and accuracy, as N=10 introduced a small percentage of false positives (0.02%). It is also worth noting that the “1k” dataset did not show any false positives when comparing different values of N. Of course N=20 is certainly not a perfect number, and will vary between datasets as shown. It should only serve as a general guideline for where this number should be. Different needs may warrant a need for speed over accuracy, in which N=10 may be desirable. I also suspect that if given access to Twitter’s full stream, a high N would be needed as similar tweets would be separated by many other random tweets. When analyzing a dataset filtered on a general, popular topic, N=20 appears to be a good compromise between speed and accuracy.
We can also take away some other statistics:

Figure 4: Neighbors compared vs Tweets per second

We can see that as $N$ grows linearly, and that tweets per second decreases logarithmically.

Figure 5: Neighbors compared vs. Time taken
We can also see that N and Time Taken both grow approximately linearly. In regards to how large N is, the complexity would be O(N * number of tweets), or simplified to just O(N) which explains this graph.

4.1.2 Text Similarity

This is a difficult one to analyze, as it is ultimately subjective as to what the exact threshold for text similarity is. However, below will show some examples to shed some insight.

<table>
<thead>
<tr>
<th>String 1</th>
<th>String 2</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is a test</td>
<td>This is a test</td>
<td>1.0</td>
</tr>
<tr>
<td>abcd</td>
<td>efgh</td>
<td>0.0</td>
</tr>
<tr>
<td>This is a test</td>
<td>This is another test</td>
<td>0.83</td>
</tr>
<tr>
<td>(1) Please check out my awesome link! t.co/50938</td>
<td>(2) Please check out my awesome link! t.co/93452</td>
<td>0.916</td>
</tr>
<tr>
<td>How different do two strings have to be to be different?</td>
<td></td>
<td>0.392</td>
</tr>
<tr>
<td>Apples are good! #VoteForApple</td>
<td>Oranges are bad! #VoteForApple</td>
<td>0.76</td>
</tr>
<tr>
<td>Apples are good! <a href="http://t.co/kjh3sf">http://t.co/kjh3sf</a> #VoteForApple 1</td>
<td>Oranges are bad! <a href="http://t.co/pl5ibq">http://t.co/pl5ibq</a> #VoteForApple 2</td>
<td>0.725</td>
</tr>
<tr>
<td>These two strings should be very different from each other.</td>
<td></td>
<td>0.103</td>
</tr>
<tr>
<td>Apples are good! <a href="http://t.co/kjh3sf">http://t.co/kjh3sf</a> #VoteForApple #ApplesRock 1</td>
<td>Oranges are bad! <a href="http://t.co/pl5ibq">http://t.co/pl5ibq</a> #VoteForApple #OrangesSuck 2</td>
<td>0.677</td>
</tr>
</tbody>
</table>

*Table 24: Text similarity example*
The preliminary threshold was set to 0.6, which would appear to be pretty close to a good threshold to choose. Some of the examples above were modeled after some bot tweets, so this is taken consideration when choosing a threshold. Ultimately, 0.65 - 0.7 appears to be the optimal range, and 0.65 will be chosen as the threshold used in this effort. As stated, this is not easily quantifiable and subjective, but hopefully the comparisons will partly justify and rationalize the chosen threshold.

4.1.3 Time Difference

This threshold is even more difficult to justify and analyze. As with text similarity, it is also subjective. How soon is too soon for similar tweets to be sent? The only partial justification for the chosen number would be that in a distributed botnet, it may take several seconds for a command to propagate across the internet, be interpreted, and acted upon. A threshold of 4 seconds should be enough for this. Any similar tweet sent in under 4 seconds in relation to another similar tweet will be given extra points towards its score.

Again, this is subjective and non-quantifiable. The threshold chosen will be 4,000ms. It is worth saying again at this point, that since multiple attributes are analyzed, no single attribute can change a determination. Only in conjunction with several others can a determination be made.

4.1.4 Entropy

The choosing of an entropy is fairly straightforward as shown by [1] and [2]. In conjunction with my work in [36]. In [2], 4.5 is used as the threshold for bots, and 6 is used for cyborgs. Personal analysis has had similar results.
In the “isis” dataset, which has an unusually high amount of bots (~25%), there is a clear spike in the number of tweets with an entropy around 4.5.

Further k-means clustering performed on the full “politics” dataset shows that the bottom cluster tops out just above 4.5 as well, and the next cluster topping out just below 6.

Figure 6: Entropy of the "isis" dataset
This sufficiently shows that there are clusters of likely bots between 0 - 4.8, and cyborgs and bots would be between approximately 0 - 5.8. For this reason, an entropy threshold of 5.5 has been chosen, so that we include the “worst” cyborgs, and possible bots. This number can of course vary depending on the dataset and desired outcomes, but 5.5 should be an acceptable threshold.
4.1.5 Sentiment

To confirm their work, sentiment analysis was performed on a dataset consisting only of spammers, and another on a random sample of tweets.

This set is a sentiment analysis of a small random sample of tweets to be used as the ground truth. The average polarity (which is the important and most used metric of sentiment) shows to be 0.136 which is similar to a spike in normal user's in [9]'s work.
Next is the analysis of a dataset consisting only of spam. The average polarity is 0.659, which is very similar to the spike in figure 8.
From this analysis, it is clear that sentiment has a strong correlation with spammers, and that this analysis validates [9]'s. As the spike in [9] for spammers is quite sharp, a threshold of 0.5, a value just below has been chosen.

4.1.6 High Score

Finally, there must be a final high score threshold used to determine whether a tweet is from a bot or not. This is a difficult value to quantify and justify, as it’s a fuzzy line where spammers and legitimate users meet on the line of possible scores. However, an analysis of the curve of high scores provides some insight.
Figure 11: High score percentage of “1k”

This is the curve for the 1k dataset. This shows that around 0.22 would be optimal.

Figure 12: High score percentage of “sample1”

This is the curve for the sample1 dataset. This shows about 0.16 would be optimal.
This is for the superbowl1 dataset. This shows around 0.25 as the optimal value.

More graphs are presented in Appendix B. The average is approximately 0.21; however, this can vary between 0.25 and 0.15, depending on the dataset. A value too high will miss some bots, and a value too low will include some legitimate users. It can only be stated that with these datasets, and all other threshold values, this tends to be a slightly conservative value that minimizes false positives while detecting a large number of bots.

However, these graphs don’t show the necessary fidelity to make a final determination. For a better view, KMeans clustering is performed on the high score computed for each tweet within various datasets to visualize this better.
First, we take a look at the politics_250k graph.

![Figure 14: K-Means clustering of high score percentage on "politics_250k"](image)

The X axis is a count of tweets, and as these tweets are in chronological order, the X axis is equivalent to time progression. We see that while clustering helps visualize this, KMeans doesn’t create a specific threshold. In this graph, approximately 0.24 would be optimal.
The next graph looked at is on the superbowl1 dataset.

This graph shows the threshold being a bit higher, at around 0.28.

In Appendix C, there are several more datasets analyzed. There is some conflict between what the optimal threshold should be. It appears that the Super Bowl related datasets are outliers with a slightly higher threshold, around 0.28 - 0.3. Since a solid number is desired for this effort, I resort to analyzing the sample1 dataset as it’s an unfiltered, random sample of tweets.
To confirm this choice, I record and analyze a second dataset from Twitter’s sample stream.
The results are in line with the ‘sample1’ dataset, with a value of 0.25.

Of course, this number varies, but overall, 0.25 seems to be the best choice. Of course this is user-configurable, and can change between datasets. For this effort, as this is a fuzzy line, a value of 0.25 will based on the mean optimal value in the analysis shown in Appendix C, and also strongly considering the results from datasets containing unfiltered, random tweets.

4.2 Analysis

The following sections will be an analysis of all the datasets detailed in Chapter 3, except for “politics” due to its size. The following table is the final values used for the numbers of neighbors compared against, and the thresholds. It is also worth noting that numbers reported for total tweets may differ from the numbers described for each dataset in Chapter 3. This is only due to a pre-processing step that is required to remove “delete messages” (messages that appear in tweet streams instructing clients that a tweet has been deleted), malformed, or other any other non-empty line that did not include a valid tweet.

<table>
<thead>
<tr>
<th>Neighbors compared against</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text similarity threshold</td>
<td>0.65</td>
</tr>
<tr>
<td>Time difference threshold</td>
<td>4,000ms</td>
</tr>
<tr>
<td>Entropy threshold</td>
<td>5.5</td>
</tr>
<tr>
<td>Sentiment threshold</td>
<td>0.5</td>
</tr>
</tbody>
</table>
### High score threshold

| High score threshold | 0.25 |

*Table 25: Example of thresholds used*

### 4.3 Results

The below graph is an example of what an organic network looks like. This is included for reference, and as an example of a ground truth for comparison against the network analysis of botnets. Most importantly, it is clear that there are many distinct groups, with a central group around the topic this dataset was filtered on.

Also, due to hardware limitations, network graphs have been limited in size. The current hardware used can only support approximately 250,000 nodes and 250,000 edges at most. Collection was limited to include most connections, for most users. This only came into effect for the largest numbers of bots detected.
Figure 18: Example of an organic network graph
4.3.1 Dataset 1k

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>1k</td>
<td>47</td>
<td>53</td>
<td>986</td>
<td>51s</td>
<td>25s</td>
</tr>
</tbody>
</table>

*Table 26: Bot statistics on "1k"

Since this was a small dataset, a full analysis was performed on the network of each user. This resulted in a fully connected graph of each bot. The first analysis was on the first 100 followers, and first 100 friends of each account.

This is highly irregular and clearly automated behavior. Each bot analyzed have the same friends and followers, without a single account outside this group. It is worth noting that
clustering was determined by the calculated modularity of this graph. Only one cluster was computed, and as such, there is only one color representing this.

A full analysis results in a different graph, but still extremely highly connected.

![Full bot network graph of "1k" (Figure 20)](image)

While the calculated modularity shows several different groups, it is still so highly connected that there are no clear or distinct groups.
Below is a snippet from the raw dataset, showing some of the spammers’ tweets.

"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "#FeelTheBern well done bernie, you're all class. https://t.co/Alm3SjJmwe", "source": "\u0
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "R7 @FFTPROJECT: Cop Running for Political Office, Found Wasted on Duty in Hio Patrol Car Wi"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "R7 @BernieSanders: When we began we were 50 pts behind the u201cinevitable Dem nominee,u2"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "You probably shouldn't talk about politics in front of me.", "source": "\u003ca href='http:\
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"
"text": "so im gonna vote a little bit for the top50fans", "source": "\u003ca href='https://about.t"

Figure 21: Example of detected spam in "1k"

Again, this is clearly automated behavior as this snippet shows tweets that were all sent at the same time, down to the second (Sun Jan 31 02:42:05 +00 2016).

Even as this is quite clearly a botnet on a campaign, analysis shows that Twitter has yet to detect and suspend these accounts. Almost 90 days after these tweets were sent, not a single detected account has been suspended.
Table 27: Bot statistics on "2_clean_25k"

First, an analysis of the bots network is performed. These results still show a very highly connected graph, although not as much as “1k” for example.

Figure 22: Full bot network graph of "2_clean_25k"
A further look shows more conclusive evidence. These two snippets from the raw Twitter data shows automated behavior. Again, these are on a chronological scale, and all sent within 2 seconds of each other.

Figure 23: Example of detected spam in "2_clean_25k"
Figure 24: Example of detected spam in "2_clean_25k"

Even some of the bot names in the detected bot list are a clear indication of a botnet.

| ken_liza1 | ken_liza10 | ken_liza11 | ken_liza12, |
| ken_liza13 | ken_liza14 | ken_liza15 | ken_liza16 |
| ken_liza17 | ken_liza18 | ken_liza19 | ken_liza2 |
| ken_liza20 | ken_liza21 | ken_liza22 | ken_liza23 |
| ken_liza24 | ken_liza25 | ken_liza26 | ken_liza27 |
| ken_liza28 | ken_liza29 | ken_liza3 | ken_liza30 |
A total of 87 accounts have been banned since analysis.
### 4.3.3 Dataset 20k

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>20k</td>
<td>77</td>
<td>135</td>
<td>13,106</td>
<td>26m 43s</td>
<td>5m 20s</td>
</tr>
</tbody>
</table>

*Table 29: Bot statistics on "20k"

*Figure 25: Full bot network graph of "20k"*
4.3.4 Dataset isis

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>isis</td>
<td>988</td>
<td>2,795</td>
<td>8,710</td>
<td>22m 22s</td>
<td>2m 28s</td>
</tr>
</tbody>
</table>

Table 30: Bot statistics on "isis"

A total of 6 accounts have been banned since analysis.
Figure 27: Full bot network graph of "isis"
Figure 28: Example of detected spam in "isis"

As the text similarity library is language agnostic, even if the text isn’t English, we can compute the similarity. This is shown above, as there is clearly automation when sending these tweets.

A total of 803 accounts have been banned since analysis.
4.3.5 Dataset jadine

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>jadine</td>
<td>371</td>
<td>1,382</td>
<td>1,829</td>
<td>13h 50m 27s</td>
<td>29s</td>
</tr>
</tbody>
</table>

*Table 31: Bot statistics on “jadine”*

The graph analysis produces the following result.

*Figure 29: Full bot network graph of “jadine”*
This, as the other botnets, is a highly connected graph, with very little distinction between each cluster.

A smaller look at only a few accounts shows a clearer picture.

*Figure 30: Subset of bot network graph "jadine*
The modularity calculated is 1, which means there is only one group detected. As can be seen, each account is friends with not only each other, but also has the same friends and followers. This is highly irregular, and conclusive evidence of a botnet.

Furthermore, a closer look into the raw data confirms this.

A total of 79 accounts have been banned since analysis.
4.3.6 Dataset ny_primary_250k_1

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>ny_primary_250k_1</td>
<td>1,781</td>
<td>9,477</td>
<td>248,618</td>
<td>3h 46m 21s</td>
<td>1h 39m 39s</td>
</tr>
</tbody>
</table>

*Table 32: Bot statistics on "ny_primary_250k_1"*

A network analysis provides the following results.

*Figure 32: Full bot network graph of "ny_primary_250k_1"*
As with other datasets, this shows a highly connected graph, with no clear distinction between clusters.

A total of 26 accounts have been banned since analysis.

4.3.7 Dataset ny_primary_250k_2

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>ny_primary_250k_2</td>
<td>2,119</td>
<td>9,327</td>
<td>248,497</td>
<td>3h 47m 36s</td>
<td>1h 41m 40s</td>
</tr>
</tbody>
</table>

*Table 33: Bot statistics on "ny_primary_250k_2"*
Again, a very dense cluster can be seen in the middle, showing that the majority of these detected bots are connected to each other.

A total of 43 accounts have been banned since analysis.
4.3.8 Dataset politics_10k

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>politics_10k</td>
<td>181</td>
<td>424</td>
<td>9,914</td>
<td>10m 41s</td>
<td>4m 8s</td>
</tr>
</tbody>
</table>

*Table 34: Bot statistics on "politics_10k"*

A network analysis shows the following results:

*Figure 34: Full bot network graph of "politics_10k"*
This graph isn’t quite as highly connected as suspected, as there are clear distinctions between each group. While most clusters are highly connected with each other, not all are. However, a closer look at the raw data shows some insight. It is worth noting that this method isn’t perfect, and will occasionally have false positives. The cluster around “Clear99” is one of these false positives, for example.

However, a closer look shows some automated behavior in the raw Twitter data.

Furthermore, the list of bots contains clear indication of botnet behavior.
A total of 43 accounts have been banned since analysis.
4.3.9 Dataset politics_250k

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>politics_250k</td>
<td>2,916</td>
<td>14,613</td>
<td>247,858</td>
<td>4h 45m 15s</td>
<td>1h 33m 20s</td>
</tr>
</tbody>
</table>

Table 36: Bot statistics on "politics_250k"

Figure 36: Full bot network graph of "politics_250k"

A total of 681 accounts have been banned since analysis.
4.3.10 Dataset red4tum

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>red4tum</td>
<td>188</td>
<td>775</td>
<td>3,216</td>
<td>25m 1s</td>
<td>48s</td>
</tr>
</tbody>
</table>

*Table 37: Bot statistics on "red4tum"*

*Figure 37: Full bot network graph of "red4tum"*
This group of bots seems to have taken an alternative approach of segmenting themselves. Most clusters are still extremely dense though, which is not organic.

A total of 44 accounts have been banned since analysis.

### 4.3.11 Dataset sample1

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample1</td>
<td>208</td>
<td>242</td>
<td>36,254</td>
<td>18m 58s</td>
<td>8m 41s</td>
</tr>
</tbody>
</table>

*Table 38: Bot statistics on "sample1"*

*Figure 38: Full bot network graph of "sample1"*
As with most bot graphs, this shows an un-organic mesh of accounts that are highly connected. The top blue/orange section seems to be separate, which may be a false positive, or possibly just a different bot net.

An example of some of the spam tweets from this dataset can be seen above.

A total of 10 accounts have been banned since analysis.
4.3.12 Dataset snl

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>snl</td>
<td>1,584</td>
<td>1660</td>
<td>4,091</td>
<td>32m 9s</td>
<td>1m 27s</td>
</tr>
</tbody>
</table>

*Table 39: Bot statistics on "snl"*

*Figure 40: Full bot network graph of "snl"*
This graph resulted in a unique result. It would appear fairly organic, with many small users with an occasional popular account, however a closer look at the list of usernames reveal a common pattern that could only come from automation.

\begin{verbatim}
zedunysis is likely a bot!
zagajipe is likely a bot!
zhaecria is likely a bot!
zahemevyxc is likely a bot!
Zaki Tamani is likely a bot!
zakofizulec is likely a bot!
Zana Bartlow is likely a bot!
zandramaasbeo is likely a bot!
zatipeqal is likely a bot!
Zeafuodn is likely a bot!
zebyjocyleg is likely a bot!
zefivyse is likely a bot!
zegeasabam is likely a bot!
zehugtygu is likely a bot!
zeloryfilum is likely a bot!
zemonaqim is likely a bot!
zezitavixc is likely a bot!
Zeus_Manda is likely a bot!
Ziakvdpj is likely a bot!
ziatutuyv is likely a bot!
ziufvuykif is likely a bot!
ziximitafan is likely a bot!
zo_e_febiray is likely a bot!
zofanxywyn is likely a bot!
Zondrakbybr is likely a bot!
zogotodu is likely a bot!
ZOZO_ALEX60 is likely a bot!
Zsarsarvzedg is likely a bot!
zybykeza is likely a bot!
zuodezux is likely a bot!
zuhaazugarzyp is likely a bot!
zukawizojug is likely a bot!
zukiwuzzaco is likely a bot!
zuynbudexco is likely a bot!
zuipixwail is likely a bot!
zuvasokola is likely a bot!
zuwivowojeg is likely a bot!
zuzaebey is likely a bot!
zuzenykomip is likely a bot!
zuzylaty is likely a bot!
zybehepuq is likely a bot!
zybesynew is likely a bot!
zyheghahy is likely a bot!
zyxlaytavbiv is likely a bot!
\end{verbatim}

Table 40: Example of detected bot names in "snl"
As can be seen, the majority of names in this snippet (and is present throughout the entire list of bot names), follow a specific pattern of consonant-vowel-consonant-vowel etc.

A total of 149 accounts have been banned since analysis.

4.3.13 Dataset superbowl1

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>superbowl1</td>
<td>987</td>
<td>1,231</td>
<td>13,522</td>
<td>33m 56s</td>
<td>3m 57s</td>
</tr>
</tbody>
</table>

Table 41: Bot statistics on "superbowl1"

Figure 41: Full bot network graph of "superbowl1"
Very clear example of a botnet cluster in the center, with likely false positives on the outside. Again, it’s a very un-organic network graph.

Some examples of the spam tweets can be seen above.

A total of 293 accounts have been banned since analysis.

### 4.3.14 Dataset superbowllive

<table>
<thead>
<tr>
<th>Set name</th>
<th>Bots</th>
<th>Bot Tweets</th>
<th>Total Tweets</th>
<th>Period</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>superbowllive</td>
<td>2,567</td>
<td>2,968</td>
<td>58,623</td>
<td>5m 15s</td>
<td>15m 34s</td>
</tr>
</tbody>
</table>

Table 42: Bot statistics on “superbowllive”
This yielded similar results to the ‘red4tum’ graph, in that there are many distinct clusters, yet most are very dense. Again, this may simply be multiple botnets, or a different approach at bots attempting to appear organic.
An example of some spam tweets can be seen above.

A total of 388 accounts have been banned since analysis.

### 4.4.1 Bots

Some bots use specific algorithm for name creation as can be seen in the ‘snl,’ ‘politics_10k,’ and ‘2_clean_25k’ results. The most common methods used was to use the same name and simply increment a number at the end, or alternatively a less obvious approach was to randomly generate names. The latter is easily caught with a close inspection, as shown in the ‘snl’ dataset.
An analysis of every detected bot’s creation time shows some interesting results as well. In the graph above, it is clear that there are occasionally up to hundreds of bots created within a short period of each other.

A total of 2,586 accounts have been banned since analysis, out of 10,994 detected unique bots.
Chapter 5:

Summary

5.1 Conclusions

This work has shown a very strong argument for this type of approach, and has shown a high success rate. While unquantifiable, as it is unfeasible to manually verify each tweet, its accuracy is strongly estimated to be above 90%, and likely closer to 95%, if not slightly higher.

It’s also evident that a complicated, resource intensive approach is not always optimal. A simple heuristic looking at the correct pieces of information in conjunction with each other can yield accurate results, while being extremely efficient, both resource and time wise. It’s estimated that this approach could analyze 100% of all tweets in real-time, with only a few high end commercial grade computers, based off of the performance of the commodity hardware used in this effort. This is not surprising, as there is no operation performed in this analysis that is resource intensive. All analysis is based around simple comparisons, equality checks, and calculations.
Some final statistics:

<table>
<thead>
<tr>
<th>Bots (non-unique)</th>
<th>Bot-Tweets</th>
<th>Total Tweets</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>14,585</td>
<td>53,503</td>
<td>920,008</td>
<td>~4,401.3 MB</td>
</tr>
</tbody>
</table>

Table 43: Final statistics 1

<table>
<thead>
<tr>
<th>Time Taken to Collect</th>
<th>Time Taken to Analyze</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 day 5 hours 33 minutes 55 seconds</td>
<td>5 hours 45 minutes 17 seconds</td>
</tr>
</tbody>
</table>

Table 44: Final statistics 2

Which comes to a ratio of 5.14:1 when comparing time taking to collect vs time taken to analyze. Another way of stating this is that it takes five times longer to collect tweets than it does to analyze them. This is subjective, as not all datasets were equal in popularity, but most were collected around popular terms of the time. It was found that the attributes chosen based on previous research by myself and others appear to be strongly correlated to bots and when combined accordingly, and can yield accurate results for detecting botnets.

5.2 Recommendations

Based on my results, I recommend that this technique be applied to similar social networks to determine its effectiveness. The attributes used for comparison are purposefully OSN agnostic, so that this approach is not limited in scope to only Twitter.

I also recommend, if possible, to analyze 100% of a social networks traffic to attempt to gain insight into the larger picture of botnet activity on an OSN as a whole.

This technique may be more effective if some of the attributes were better analyzed to conclude what thresholds should be set, if any. It may be possible to remove some attributes altogether. I would recommend analyzing the “value” of each attribute in a more quantifiable way.
It would be worthwhile to recreate or expand upon “Social Spammer Detection with Sentiment Information” by Hu et al. [9] as an alternative to this approach, as it is closely related. “Detecting Spam and Promoting Campaigns in the Twitter Social Network” by Zhang et al. [20] is another approach worth investigating, as they have a heavy focus on analyzing URLs which could also be very telling.

I would like to continue this work to determine whether the same accuracy of results can be obtained, with less attributes.
REFERENCES


[33] V. S. Subrahmanian, Amos Azaria, Skylar Durst, Vadim Kagan, Aram Galstyan, Kristina Lerman, Lihong Zhu, Emilio Ferrara, Alessandro Flammini, Filippo Menczer,


APPENDICIES

Appendix A:

Source code for main program. Some parts have been removed for brevity. These portions are denoted by “REDACTED”, and subsequent pseudo code has been added.

```python
import time
import sys
from mastersLibrary import *

###################################################################
# VARIABLES
###################################################################
simCompareLimit = 30  # This must be an even number. Memory needed for holding tweets
# = simCompareLimit * 5075 bytes
textSimThresh = 0.6  # Scale of 0-1, threshold for similarity
timeDiffThresh = 4000  # This is crappy, need to use this better
entThres = 5.5  # Threshold, used like: if entropy < entThres, add points to high_score
sentThresh = 0.5  # Threshold for sentiment. Used like above^
highScoreThresh = 0.35  # Percentage of total_score that is needed to be called a bot
debug_level = 0

# File Paths
inputFile = "/path/to/input.file"
outputFile = "/path/to/output.file"

# MULTIPLIERS
# High impact multipliers
simScoreMult = 1.2
sentimentMult = 1.2
entropyMult = 1.2
simCountMult = 1.2

# Other multipliers
locCountMult = 1
uaCountMult = 1
tzCountMult = 1
descCountMult = 1
urlCountMult = 1
timeDiffMult = 1
genderMult = 1
langCountMult = 0  # If tweets were filtered by language (all are the same), this should be 0, otherwise 1

# Bonus points
bonus_for_sent = 10  # bonus points are added if sentiment > sentThresh
bonus_for_ent = 10  # bonus points are added if entropy < entThres
```

122
simPlusMinus = simCompareLimit / 2
if langCountMult == 0:
    total_score = (simCompareLimit * 9) + bonus_for_sent + bonus_for_ent
else:
    total_score = (simCompareLimit * 10) + bonus_for_sent + bonus_for_ent

are_we_done = False

with open(inputFile) as f:
    tweets = f.readlines()

print "Total tweet count is: " + str(len(tweets))
print "Preprocessing now..."
while not are_we_done:
    tweets, are_we_done = pre_process_json(tweets)
    are_we_done = False

while not are_we_done:
    tweets, are_we_done = pre_process_ent(tweets)
    are_we_done = False

while not are_we_done:
    tweets, are_we_done = pre_process_gender(tweets)
    are_we_done = False

while not are_we_done:
    tweets, are_we_done = pre_process_sent(tweets)
print "Total tweet count is now: " + str(len(tweets))

start_time = int(time.time() * 1000)

# REDACTED: Declare Arrays

firstTweetTime = json.loads(tweets[0]).get('timestamp_ms')
lastTweetTime = json.loads(tweets[-1]).get('timestamp_ms')
totalTweetTime = int(lastTweetTime) - int(firstTweetTime)
totalTweetTime = int(totalTweetTime / 1000)

print "Filling arrays..."
# Extract important info from tweet, and build arrays from it.
for tweet in tweets:
    # REDACTED: Fill arrays with data

print "Entering main loop..."
# Aggregate similarity score of 50 nearest tweets. Possible high score of 50.
# Also counts HOW many were similar
prog = 0
for x in range(0, len(text)):

    perc = 100. * prog / len(text)
    percCounter = int(2 * perc / 5)
    print '\rCalculating... [' + '#' * percCounter + ' ' * (40 - percCounter) + ']',
    print '%.2f%% % perc, (%d/%d) % (prog, len(text)), sys.stdout.flush() #
    flush io
    prog += 1
    simScoreCount, tzCount, simCount, langCount, genCount, uaCount = 0, 0, 0, 0, 0, 0
urlCount, locCount, descCount = 0, 0, 0
timeCount = 0

    # REDACTED: Set upper bound and lower bound for sliding window

for y in range(lowerBound, upperBound):
    # REDACTED: Loop through sliding windows tweets
    # and compare all attributes

    # REDACTED: Open CSV file, and write the header.
    list_of_likely_bots = []

for i in range(0, len(tweets)):
    # REDACTED:
    # Compute the high score for each tweet
    # Add bonus points if necessary
    # If score is above threshold, add to list for later printing
    # Write all scores to CSV

    botTweetCount = 0
for i in range(0, len(tweets)):
    if sname[i] in list_of_likely_bots:
        botTweetCount += 1

print "$\n#### OUTPUT ####\n"
for bot in sorted([x.encode('UTF-8') for x in list(set(list_of_likely_bots))],
key=str.lower):
    print str(bot) + " is likely a bot!"

print "$\nThere were " + str(len(list(set(list_of_likely_bots)))) + " bots, which
tweeted a total of " + str(botTweetCount) + " times out of " + str(len(tweets)) + " total tweets over
a period of " + str(int(totalTweetTime / 60 / 60)) + "h " + str(int(totalTweetTime / 60 % 60))
+ "m " + str(int(totalTweetTime % 60)) + "s"

end_time = int(time.time() * 1000)
total_time = (end_time - start_time) / 1000
print "$\nTotal time taken: " + str(int(total_time / 60)) + "m " + str(total_time % 60) + "s"
Appendix B:

Graphs showing the high score curve on various datasets.
Appendix C:

Politics_250k

Red4tum