Analyzing Twitter Activity to Identify Spam Accounts

Stephen Giuliani
Indiana University
Virginia, USA
sgiulian@iu.edu

ABSTRACT

This paper provides an analysis of Twitter account activity to identify potential spamming bots as opposed to organic user activity. The approach uses a combination of web scraping in python, data storage within the MongoDB Atlas Cloud, and implementing a REST service to pull data from the cloud, expose it on a local system, and then archive the data for analysis.

KEYWORDS

hid-sp18-507, Twitter, MongoDB, REST, TweePy

1 INTRODUCTION

The 2016 US Presidential election is a prime example of how social media can be used to push an agenda or advertise specific points of view in such a volume that discrediting the masses of inaccurate posts, shares, and re-tweets is a near impossible feat. Many individuals, organizations, as well as government entities, have utilized social media to flood various social mediums with repetetive posts and comments at the hands of automated bots, rather than real-user activity [11]. Since the unethical use of social media in this manor has come to light, post election, the US Department of State (DoS) and the Federal Bureau of Investigation (FBI) has been tasked to identify and address the use of fake accounts to push political agendas and manipulate the media as a whole [12].

Luckily for investigators, account activity for automated or fake users is typically characteristically different from the activity of organic or real users [3]. This paper will provide an analysis on Twitter users' accounts and user-habits to identify potential bots. This paper will also identify certain topics that are more likely to fall victim to the spamming tactics than less polar-opinion subject matter.

2 BOTS, SPAM, AND THE IMPACT

Automation is a norm in technology, especially social media. The purpose of the automated account, or bot, is what determines whether the account is considered spam versus a service. Examples of more favorable automated accounts include the 'Earthquake Bot', which automatically tweets warning information to areas where a 5 or greater magnitude earthquake is detected, the 'Netflix Bot', which will notify you each morning of the latest Netflix releases, and the 'Big Ben Clock', which just tweets 'BONG' every time London's Big Ben chimes [8]. The fun and useful bots are just a few examples of how Twitter accounts can be used for a collectively positive purpose. In contrast, there are accounts that serve only to like, re-tweet, or even produce original content, at a higher daily rate and often focus on subject that are hot-topics in the news or polarized views in politics. Within this analysis, we have concluded that certain topics are more likely to have a spam bot presence than others. For instance, '#LockHerUp', a notorious hashtag used in favor of putting Hillary Clinton in jail, is used by bots significantly more than '#SundayBrunch'.

The accounts that can be considered spam-based bots have similar account behavior to other bots and can be classified collectively via just a few data points. A spam bot is more likely to post at a much higher volume than a real user. Some bots rack up high counts of re-tweets with very few original posts, have a very unbalanced ratio of followers to people who follow back, and may achieve a high volume of activity within a few days of creation.

The use of these spaming accounts creates artificial interest, or hype in specific subjects. This tactic can keep topics in the limelight or falsely overpower reputable news sources ultimately controlling what we see online or in mainstream media [1]. Instances where the masses latched onto the volume of information rather without investigating source or information independently can lead to dangerous conclusions and sometimes unjustified courses of action. In 2012, the moments immediately following the unfortunate incident at Sandy Hook lead to an online hunt and prosecution of Ryan Lanza. Varieties of social mediums were quick to post and share than Ryan was the perpetrator of the terrorist attack, including death threats and pledges of revenge. However, as a short time passed and sources were verified and facts pushed through the social hype, Adam Lanza, Ryan's brother, was identified as the shooter–well after news outlets followed and falsely reported on socially-based conclusions [5].

3 ANALYTICAL APPROACH

In this section we will provide an overview of the primary steps involved in acquiring, storing, querying, and analyzing the subject Twitter data.

Note to the reader: In order to create, and therefore replicate, this paper, independent Twitter, Gmail, and Twitter Developer accounts were created. These accounts and credentials are necessary for access to Twitter's API. Additionally, a temporary access account was set up so that instructors and reviewers could access and replicate data storage and pulls within the MongoDB Atlas cloud service. The credentials and access information is provided within the project code and no changes should be made.

3.1 Technologies Used

Twitter API

Twitter uses a publicly available API for all of its users' Twitter activity and profile information. Free developer accounts are available and come with the ability to pull data from the past 7 days. Increased scope is available through paid-tiers of developer accounts.

Python: TweePy

Tweepy is a package developed for the purpose of interacting with Twitter's online API [9]. Queries are returned in JSON format for analysis or direct storage.

MongoDB Atlas

MongoDB is a popular NoSQL document database program that is open source and can be deployed almost anywhere. MongoDB Atlas is a cloud-based MongoDB service that can host documents through the services of Amazon Web Services (AWS), Microsoft's Azure, or Google's Cloud Platform (GCP) [6]. For purposes of this paper and analysis, we have used MongoDB Atlas to host the scraped Twitter data, using AWS, and expose the data through a RESTful service.

Python: Flask (REST)

Flask is a micro framework for Python that can be used to design and make RESTful queries. We use flask to make GET requests to the MongoDB Atlas Cloud data [4]. The data requested is used for the analysis itself. Once the data is scraped and stored within the cloud, python and Flask expose, or makes available, the data on a predetermined web address, or local endpoint for purposes of this paper.

3.2 Web Scraping, Twitter Data

In order to pull data from Twitter's API, you must first register for a developer account (free). Once registered, and after creating a default application, the account keys and credentials are necessary for authentication when using tweepy in python. A free developer's account has the ability to pull twitter data from the past 7 days only. For purposes of this analysis, these seven days are sufficient. Documentation on the data Twitter makes available to the public is available here [13]. Using tweepy, we are able to query Twitter's data via a number of characteristics: hashtags, words present in tweets, usernames, dates, and more. The JSON object that is returned (dependent on the structure of the query) includes information on the user's account since creation as well as meta-data on the tweet itself (geotags, platform used to post, links used, and more). The entirety of the returned JSON object is what is stored on the MongoDB Atlas cloud.

A typical Twitter query JSON object is less than 8 kB at a length of about 212 lines of code, based on a JSON dump of 100 tweet-data returns. Therefore, a high volume of twitter data can be pulled in stored within the cloud for analysis. However, according to Twitter's API policy [14], only 500 instances can be queried at once with a registered development user (paid options available for higher volume queries).

3.3 MongoDB Atlas, Storing Twitter Data

MongoDB's online web-hosting cloud service is available for free for testing purposes. The limitations of this service is a 3-cluster database with a storage limit of 512 Mb. Any requirements to use more than 3 clusters or storage needs greater than 512 Mb requires paid subscriptions, which are hourly-usage based. The data scraped from Twitter will be stored on this cloud and will be available to readers/reviewers after the analysis is concluded. For purposes of demonstration and this paper, the query max of 500 instances will be stored and used for the analysis. However, at an average of 8kb per tweet query, over 64,000 instances can be stored within the free 512 Mb of data MongoDB Atlas offers.

The ability to store data on a MongoDB database comes from the use of pymongo, a Python package developed for the administration of Mongo databases [15]. However, for testing purposes, we also used the mongo-shell capabilities to clear, start, stop, and monitor the cloud-server status. Documentation on pymongo and mongo-shell are available at https://api.mongodb.com/python/current/.

Connection to the cloud database requires ssl authentication, which will be published within this paper for testing purposes. Administration or viewing permissions will also be created. The JSON objects produced within Python using tweepy are stored in a created database collection on the Atlas cloud. This cloud database is also use as the connection for the RESTful service to pull specific data-points.

3.4 Flask, RESTful Service in Pulling from the Cloud

The data or JSON-format documents stored on the Atlas cloud can be connected to and authenticated via pymongo in python. Flask is used to bridge the connection and expose an available address for data to be displayed/used for analysis [10]. The RESTful API and paths created were created from scratch, grouping specific JSON attributes of the stored data. For instance, in order to determine the daily rate of posts per user, the GET request will pull the data for the user's total posts and the data including the creation date of the account. The JSON object produced via the GET request can be parsed and regrouped prior to exposing the data on flask's default local endpoint. See the code-documentation for the REST API used to query the Atlas cloud.

3.5 Data Analysis

As mentioned above, the data on the cloud can be pulled via GET requests. The data required is pulled via specific GET requests so that analysis can be done. MongoDB's Atlas cloud service has the ability to perform analysis on the cloud; however a free testing account is not able to perform this. Therefore, the analysis is conducted locally, via python objects. For purposes of data familiarization, the data is pulled and stored in a comma-separated values (CSV) format. This enables seamles interpretation by python's many analytical and visual packages. Metrics such as daily rates for friending other users and posting, friends-to-followers ratios, and account age are calculated. Medium, a popular online publication and editorial provides "Twelve Ways to Spot a Bot" [2] on Twitter; many of these points can be verified via Twitter's public API data. For analytical purposes, the GET request, hosted by flask through localhost endpoints, is downloaded as a raw JSON object and converted and stored locally as a CSV. This paper will analyze the scraped users' average posts per day, average favorites per day, as well as the ratio of friends to followers. Spam bots are known to constantly produce tweets or retweets and therefore we can expect a high number of average daily posts or favorites per bot-users. Similarly, a spam bot often will follow, or friend, a high volume of other twitter users so that its account expose increases. In contrary to the number of friends a bot may have, the number of users that follow back are usually low; therefore a high ratio of friends to followers can indicate an automated account.

4 RESULTS

Evaluating the average daily posts, average daily favorites, and ratio of friends to followers revealed a majority of organic users; however there is still a significant number of users identified as a possible spam bot. The following tables, Table 1, Table 2, and Table 3, show the summary information on the daily post, daily favorites, and friend to follower ratio data, respectively, pulled from Twitter's user data.

[Table 1 about here.]

[Table 2 about here.]

[Table 3 about here.]

Note that the number of instances within the CSV files totals only 460, rather than the 500 available within the GET requests. This is because 40 posts are associated with duplicate accounts that are already a part of the queried data; therefore their data is duplicative and only requires a single instance for evaluation.

Users' post, favorites, and ratio data indicates right-skewed data points because the minimum and maximum values are so far apart. The averages produced are likely inaccurate for organic-only accounts. The standard deviation from each of the means also indicates the wide range of account activity.

The histograms from Figures 1, 2, and 3 support the summary data and the conclusion that some accounts are not indicative of an average user.

[Figure 1 about here.]

[Figure 2 about here.]

[Figure 3 about here.]

Arbitrary metrics for differentiating activity of a spam-bot versus an organic user we used to re-evaluate the data to filter out potential bots, or outliers, in the queried data.

Daily posts of an organic user is considered to be at most 96 posts per day (a user who posts a single tweet every 10 minutes for 16 hours each day). Favorites per day was evaluated at a similar cap, only it was increased by 25 percent to account for the simplicity and efficiency of a favorite over a produced tweet. The ratio of friends to followers was capped at 2 times the standard deviation of the original data, or about 3.5. This indicates that an account may follow three and a half times as many users as are willing to follow them.

After filtering out the users that don't fall within these thresholds, Table 4, Table 5, and Table 6 show the delta, or change in the data as impacted by the potential bot accounts.

[Table 4 about here.]

[Table 5 about here.]

[Table 6 about here.]

As shown in Table 7, Table 8, and Table 9, the average for daily posts and daily favorites fell by more than half the original mean. Of the 59, 47, and 28 accounts that were filtered out per each category, the top ten and the values that flagged them are shown below.

[Table 7 about here.]

[Table 8 about here.]

[Table 9 about here.]

The analysis in each category was conducted independently. However, merging the Top 10 candidate bots based on daily activity shows 5 accounts sharing the top ten in both categories. When comparing the top 10 based on friend-follower ratios, none of these users share the top 10 with the daily-activity-based top 10 accounts.

5 CHALLENGES AND OPPORTUNITIES

The purposes of this paper are to demonstrate the ability to scrape, store, access, and analyze data; however the analysis itself is only part of a much larger potential research topic; one that is outside the scope of this course.

5.1 Challenges

This paper demonstrates the ability to scrape and store data on the cloud where the data can be pulled, exposed locally, and then downloaded into analytical-friendly formats. Limitations on this analysis include Twitter's policy for query frequency and size of the data returned as well as MongoDB Atlas's limit on data storage. Both of these can be mitigated via paid respective accounts. Additionally, in order to perform the analysis independent of the provided credentials, a user must register themselves for these services. Also, depending on the query or subject matter of the requested information, the freely available 7-day history might not suffice. Sentiment analysis and bot filtering on past events would require the ability to pull tweets or archived user data.

5.2 Opportunities

In order to truly conclude a spam-bot over atypical organic user further analysis should be conducted. Analysis can include re-querying Twitter for specific users (such as the 5 that shared the top tens for daily activity) where the data on a specific-user's tweets can be evaluated with sentiment analysis, analysis on the repetition of tweets, other subjects the user promotes, and the platforms and languages used to tweet. Bots typically are based on programs that are not run using a mobile platform, such as the Twitter app on a phone; contrarily, most organic twitter users use their phone as the primary posting/sharing/favoring platform. Deep learning, out of scope of this paper is a great opportunity to parse through the dense JSON object pulled per user.

Additionally, some cloud services and platforms come with analytical capabilities in house. The lowest tier within the MongoDB Atlas cloud service does not offer data analytics; however should a cloud database platform be used with these capabilities it would eliminate the need to re-pull data to a local machine or server as the analytics, or even the necessary python files can be hosted and executed directly from the cloud.

6 CONCLUSION

Clearly, automation within Twitter is prevalent. According to a study conducted by the University of Southern California and Indiana University, up to 15% of twitter accounts could be bots [7]. Lately, the news, as well as the FBI, have focused on the use of these bots as a way of manipulating media and public perception. By pulling data from just the past seven days from Twitter, based on a query for a hot-topic hashtag, '#LockHerUp', and determined by activity that deviates from typical organic user activity, over 100 accounts appear to be automated and in support of the politically-charged campaign.

The ability to scrape, and directly store-then-pull data on MongoDB's Atlas cloud service was seamless, efficient, and was possible at no cost to the user. The 500 records pulled from Twitter and stored in the cloud occupied less than 1% of the available memory cap and therefore allows for opportunities to vastly increase data volumes for analysis.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Gregor von Laszewski and the Associate Instructors for their support and suggestions while writing this paper.

REFERENCES

[1] Wikipedia contributors. 2018. Twitter bot — Wikipedia. Web Page. (Apr 2018). https://en.wikipedia.org/wiki/Twitter_bot Accessed: 2018-04-05.
[2] Digital Forensic Research Lab. 2017. #BotSpot: Twelve Ways to Spot a Bot. Web Page. (Aug 2017). https://medium.com/dfrlab/botspot-twelve-ways-to-spot-a-bot-aedc7d9c110c Accessed: 2018-04-01.
[3] Supraja Gurajala, Joshua White, Brian Hudson, Brian Voter, and Jeanna Matthews. 2016. Profile characteristics of fake Twitter accounts. Web Journal. (2016). http://journals.sagepub.com/doi/pdf/10.1177/2053951716674236 Accessed: 2018-04-08.
[4] Anthony Herbert. 2017. MongoDB Backed RESTful — mongo.py. Code Repository. (Jan 2017). https://github.com/PrettyPrinted/mongodb_backed_restful/blob/master/mongo.py Accessed: 2018-04-05.
[5] Kashmir Hill. 2012. Blaming The Wrong Lanza: How Media Got It Wrong In Newtown. Web Page. (Dec 2012). https://www.forbes.com/sites/kashmirhill/2012/12/17/blaming-the-wrong-lanza-how-media-got-it-wrong-in-newtown/#55e6a9327601 Accessed: 2018-04-01.
[6] MongoDB, Inc. 2018. MongoDB Atlas. Web Page. (2018). https://docs.atlas.mongodb.com/ Accessed: 2018-04-05.
[7] Michael Newberg. 2017. As many as 48 million Twitter accounts aren't people, says study. Web Page. (Mar 2017). https://www.cnbc.com/2017/03/10/nearly-48-million-twitter-accounts-could-be-bots-says-study.html Accessed: 2018-04-08.
[8] Jannis Redmann. 2017. Awesome Twitter Bots. Code Repository. (Aug 2017). https://gist.github.com/derhuerst/1cb20598b692aa87d9bb Accessed: 2018-04-01.
[9] Joshua Roesslein. 2009. Tweepy. Web Page. (2009). http://www.tweepy.org/ Accessed: 2018-04-01.
[10] Armin Ronacher. 2018. Welcome — Flask (A Python Microframework). Web Page. (2018). http://flask.pocoo.org/ Accessed: 2018-04-01.
[11] Eli Rosenberg. 2018. Twitter to tell 677,000 users they were had by the Russians. Some signs show the problem continues. — Washington Post. Web Page. (Jan 2018). https://www.washingtonpost.com/news/the-switch/wp/2018/01/19/twitter-to-tell-677000-users-they-were-had-by-the-russians-some-signs-show-the-problem-continues/ Accessed: 2018-04-08.
[12] Chris Strohm. 2018. FBI Task Force to Expose Russian Social Media Manipulation. Web Page. (Jan 2018). https://www.bloomberg.com/news/articles/2018-01-10/fbi-plans-task-force-to-expose-russian-social-media-manipulation Accessed: 2018-04-08.
[13] Twitter, Inc. 2018. API Reference Index — Twitter Developers. Web Page. (2018). https://developer.twitter.com/en/docs/api-reference-index Accessed: 2018-04-01.
[14] Twitter, Inc. 2018. Rate Limiting — Twitter Developers. Web Page. (2018). https://developer.twitter.com/en/docs/basics/rate-limiting.html Accessed: 2018-04-01.
[15] Robert Walters. 2017. Getting Started with Python and MongoDB. Blog. (Apr 2017). https://www.mongodb.com/blog/post/getting-started-with-python-and-mongodb Accessed: 2018-04-01.

A CHKTEX

```
cd ../../hid-sample; git pull
Already up to date.
cp ../../hid-sample/paper/Makefile .
cp ../../hid-sample/paper/report.tex .
```

List of Figures

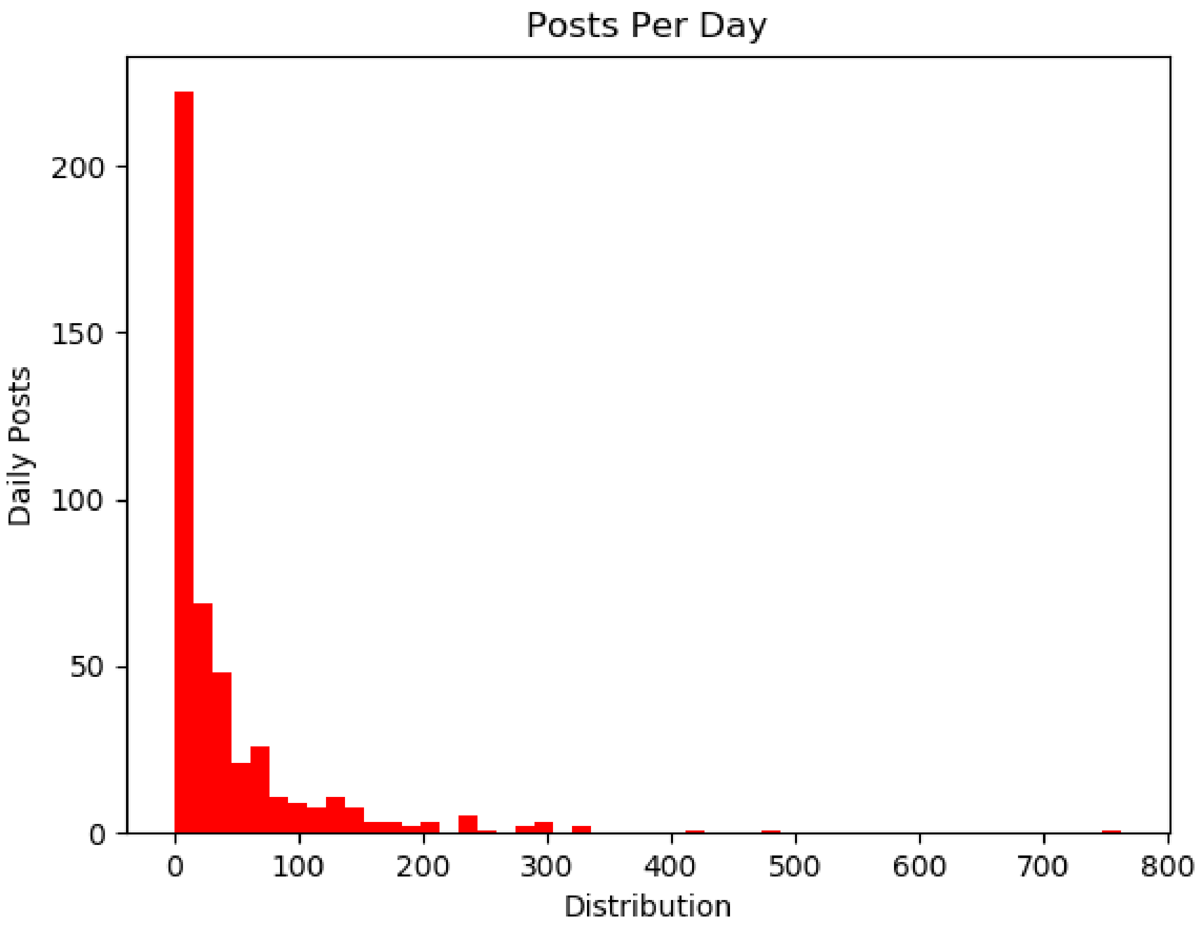

Figure 1: Histogram: Posts per Day

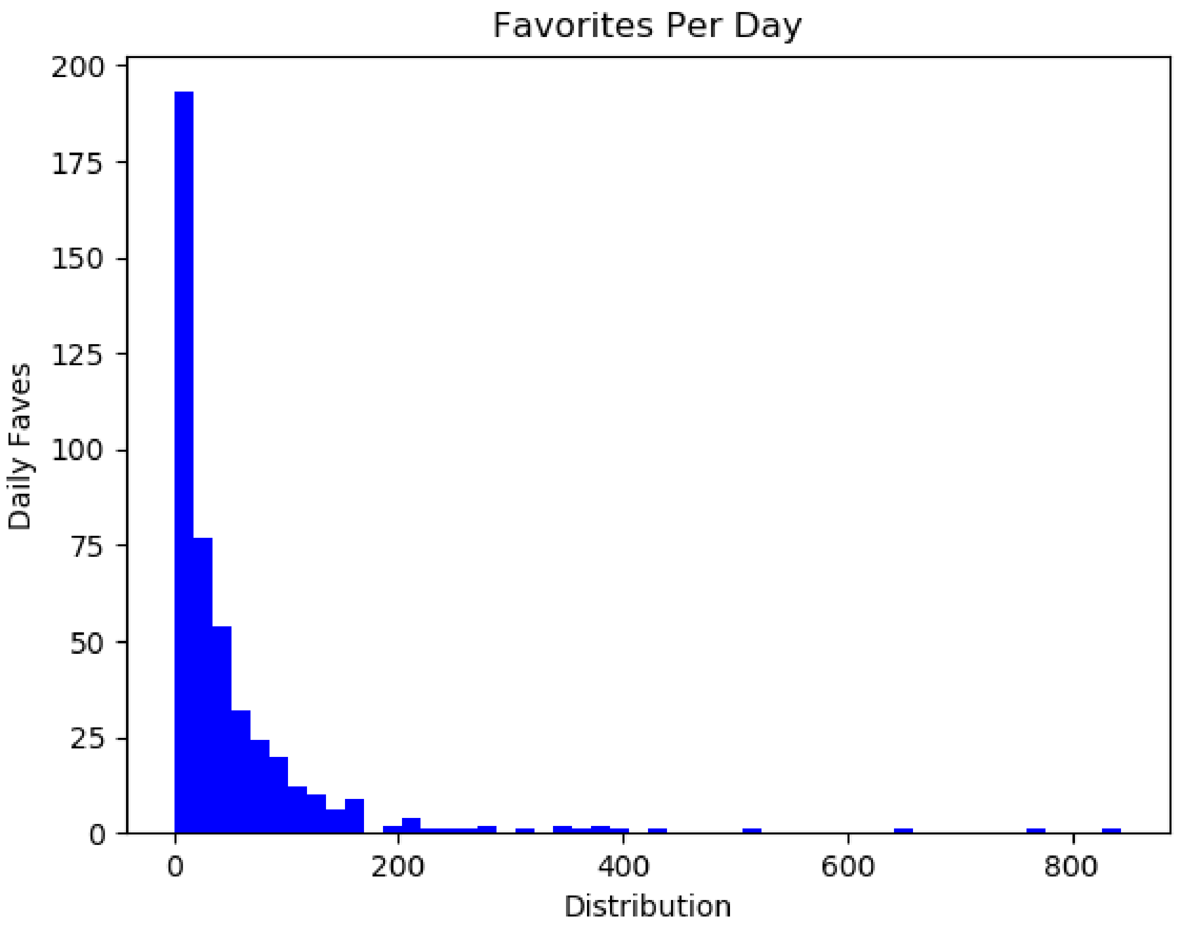

Figure 2: Histogram: Favorites per Day

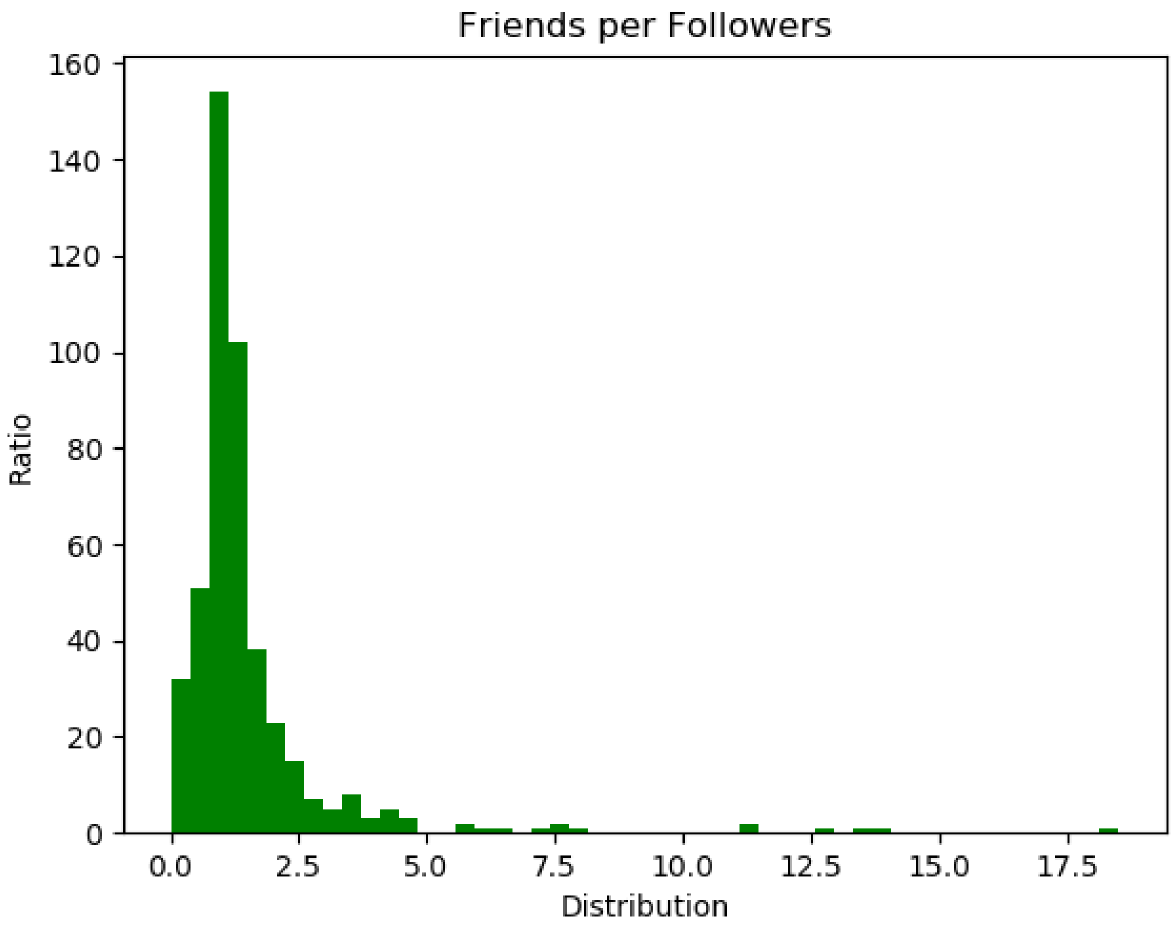

Figure 3: Histogram: Friends per Followers

List of Tables

Table 1: Summary: Daily Posts

Count	460
Mean	43.60
Std	72.19
Min	0.0014
25%	4.95
50%	17.49
75%	50.57
Max	763.09

Table 2: Summary: Daily Favorites

Count	460
Mean	52.23
Std	89.20
Min	0.0024
25%	7.84
50%	23.02
75%	59.53
Max	844.67

Table 3: Summary: Friends per Followers

Count	460
Mean	1.53
Std	1.77
Min	0.00
25%	0.88
50%	1.09
75%	1.50
Max	18.50

Table 4: Adjusted: Daily Posts

Count	59
Mean	21.59
Std	48.79
Min	0.00
25%	1.12
50%	4.32
75%	17.88
Max	667.62

Table 5: Adjusted: Daily Favorites

Count	47
Mean	22.34
Std	59.60
Min	0.00
25%	1.55
50%	3.67
75%	13.18
Max	726.27

Table 6: Adjusted: Friends per Followers

Count	28
Mean	0.35
Std	1.14
Min	0.00
25%	0.04
50%	0.03
75%	0.09
Max	14.98

Table 7: Potential Bots: Criteria = Daily Posts

Screen Name	Avg. Daily Posts
MaheshC78848209	763.09
KatTheHammer1	475.44
GaryWil29548846	421.00
DonGibs22787443	330.10
ARealPrincesa	327.32
notbuyingthat54	304.66
nice-centurion	294.95
lynn85706529	291.30
FrankJManrique3	287.52
R98250729	279.81

Table 8: Potential Bots: Criteria = Daily Favorites

Screen Name	Avg. Daily Favorites
siminuteman1776	844.67
ImmoralReport	772.75
MaheshC78848209	644.52
KatTheHammer1	522.71
GaryWil29548846	427.00
4Freedom4ever	395.86
mickeyh54099794	380.63
FrankJManrique3	374.21
lynn85706529	368.90
melinda63312935	350.65

Table 9: Potential Bots: Criteria = Friends per Followers

Screen Name	Friends per Followers
Andy75213017	18.50
KenSR-802VT	13.75
wesleyh23420956	13.44
JamesChhetri7	12.58
john316stv	11.46
Ikram-35	11.25
JaredFu88957380	7.87
LDuvall19	7.59
nateweyand	7.48
DJLPX	7.31