Coalmine: An Experience in Building a System for Social Media Analytics

Joshua S. White, Jeanna N. Matthews, John L. Stacy
Clarkson University, Potsdam, New York 13699
{whitejs, jnm, stacyjl} @clarkson.edu

ABSTRACT

Social media networks make up a large percentage of the content available on the Internet and most of the time users spend online today is in interacting with them. All of the seemingly small pieces of information added by billions of people result in a enormous rapidly changing dataset. Searching, correlating, and understanding billions of individual posts is a significant technical problem; even the data from a single site such as Twitter can be difficult to manage. In this paper, we present Coalmine a social network data-mining system. We describe the overall architecture of Coalmine including the capture, storage and search components. We also describe our experience with pulling 150-350 GB of Twitter data per day through their REST API. Specifically, we discuss our experience with the evolution of the Twitter data APIs from 2011 to 2012 and present strategies for maximizing the amount of data collected. Finally, we describe our experiences looking for evidence of botnet command and control channels and examining patterns of SPAM in the Twitter dataset.

Keywords: Social Media, Data Collection, Data Mining, SPAM, Botnet

1. INTRODUCTION

Social media networks make up a large percentage of the content available on the Internet and most of the time users spend online today is interacting with social media networks, e.g. reading or contributing blog posts, tweets, status updates, etc.

Social media networks all share one thing in common: they allow for users of the systems to contribute their own content. However that's where the similarity stops, these systems come in many forms, including;

- General search and browsing sites that allow users to recommend or suggest content for other users.
- Video, audio and image sharing sites that allow users to share their own content or review others. Popular examples include Youtube, Flickr, Vimeo and Hulu.
- Geo-mapping services allow that users to share maps about things going on in their communities on a wide range of topics from politics to health. Once such example is Google Flu which tracks users reports of flu outbreaks.
- Blogging and micro-blogging sites that allow users to post free-form updates on topics of their choice. These include everything from individual personal blogs to mass micro blog aggregation sites such as Twitter.
- Profile Sharing Sites that encourage users to share personal information with individuals they know only casually or not at all. Popular examples include Facebook and LinkedIn.

These categories for social media networks are regularly accepted classifications based on a number of works, the most controversial of which is arguably entitled “Publicly Available Social Media Monitoring and Situational Awareness Initiative” which was published by the Department for Homeland Security in 2010. It outlined a potential program to use social network data to monitor potential threats to the United States [18].

While an analysis of the privacy concerns that social media network data mining poses is out side the scope of this work, it is worth pointing out that we are continually monitoring the literature and law surrounding this. All the seemingly small pieces of information added by billions of people result in a huge rapidly changing dataset. Searching, correlating, and understanding billions of individual posts is a significant technical problem, even all the data from a single site such as Twitter can be difficult to manage.

2. COALMINE

Coalmine was designed to be a flexible data mining architecture that can process large amounts of streaming social media data, for data-mining applications. The initial focus of Coalmine was to process live Twitter updates, or “tweets”,but a similar architecture could be used for other social media sites that export their data through a defined API.

Each tweet consists of up to 140 characters of free form text. However, when processing a stream of Twitter traffic, there's a lot more than the 140 characters that make up a user’s message. Each tweet represents about 1K of data including both the text of the tweet and a rich set of meta-data about it. This additional tweet meta-data, includes information about the user, the user's account, and potentially the user's location at the time of the tweet. Since Twitter is a social networking site, many of the connections between users can be gleamed from the included post data.

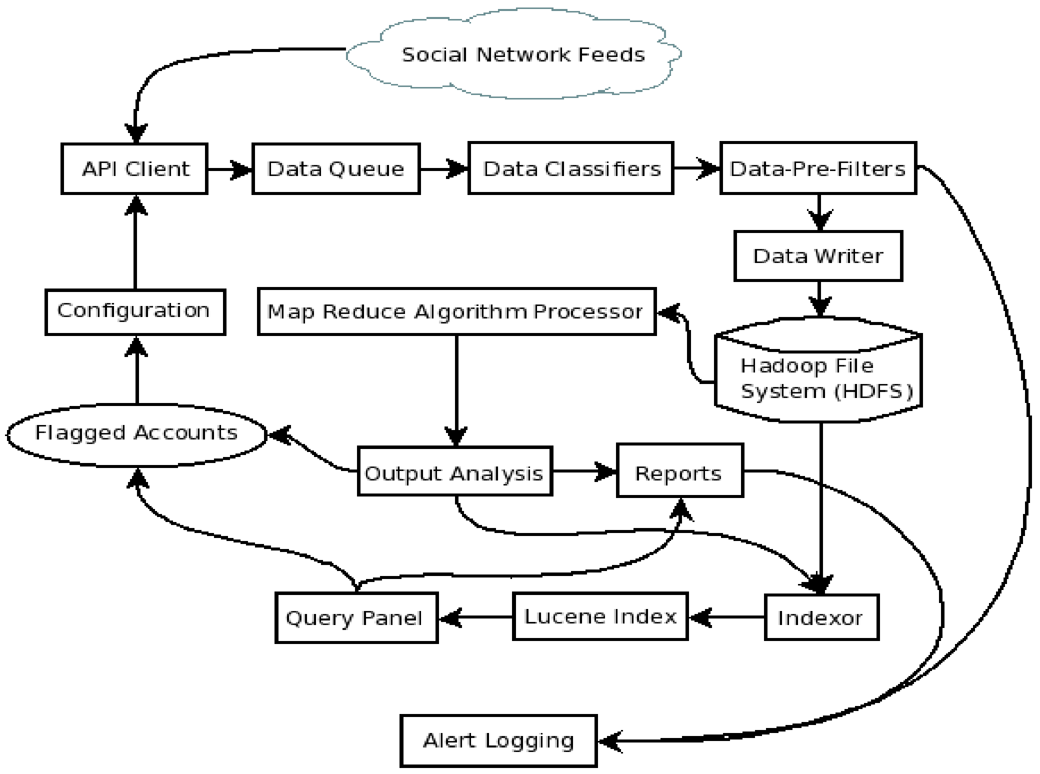


Figure 1: Coalmine - System Diagram

Our system consists of both a back end spanning multiple headless servers that captures and indexes the Twitter data stream and a front end which can be used to access or query the indexed data. Each component of our system is expressed in Figure 2. The four primary components are: (1) the data collection and storage component, (2) Ad-Hoc query tool, (3) batch processing component, (4) and live filtering, classification, and alerting.

1. The data collection and storage component consists of third party API's, such as the Twitter REST interface. Our system has been developed in both Java and Python using Tweetstreamer as an initial connection library. This client provides the basic wrapper around the Twitter API and facilitates authentication to the Twitter servers [30].

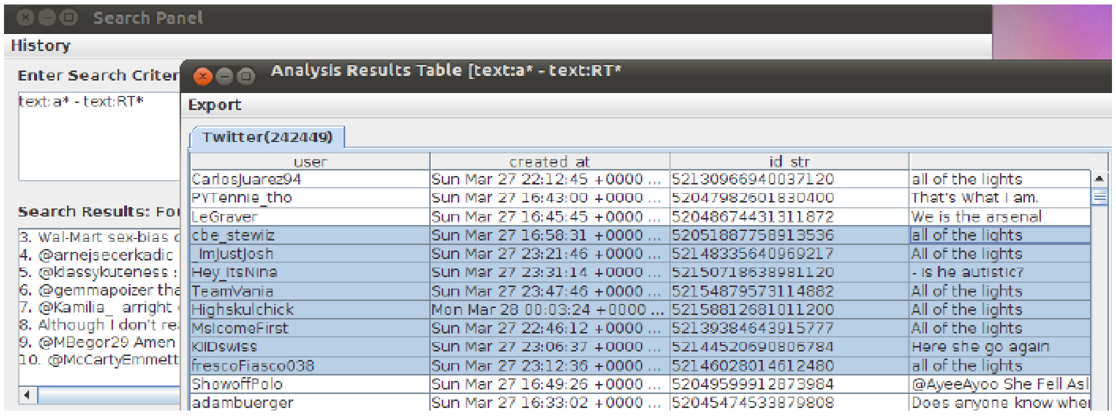
   The development team extended the simple TwitterClient to provide file IO, error handling, and multiple client connections to Twitter, based on keyword and specific user. The data is received from Twitter as a stream of JSON objects. As the individual streams are received, they are combined and buffered for file output in gzip format.

   At this time, the data is passed through an inline compression routine, before it is written to the output file. The compression ratio is high, averaging a 1:4 ratio. Based on this compression ratio, it is calculated that, in the case of Twitter, all of the tweets in a given day amount to ~150 GB of compressed data.

2. The query tool provides a user with the ability to access one or more data files through a Google-like search interface.

   The core of the query tool component, uses the Apache Lucene data indexing library. Each field of the tweet meta-data is stored within a large indexed repository, created when the query tool processed the data file. The user then queries the repository by selecting a meta-data field, such as "text", for the text of the tweet and provides a term to match.

   For example, if a user is looking for a tweet containing the word "Twitter", the user would enter the string "text:Twitter" into the query text box. The query syntax is very similar to that of Google and allows for order-of-operations within the query. Figure 1 depicts an example of the query panel and the resulting table generated from a query.



```
text:a* - text:RT*
```

```
3. Wal-Mart sex-bias
4. @arnejsecerkadic
5. @klassykuteness :
6. @gemmapoizer tha
7. @Kamilia_  arright
8. Although I don't re
9. @MBegor29 Amen
10. @McCartyEmmett
```

user	created at	id str	
CarlosJuarez94	Sun Mar 27 22:12:45 +0000 ...	52130966940037120	all of the lights
PYTennie_tho	Sun Mar 27 16:43:00 +0000 ...	52047982601830400	That's what I am.
LeGraver	Sun Mar 27 16:45:45 +0000 ...	52048674431311872	We is the arsenal
cbe_stewiz	Sun Mar 27 16:58:31 +0000 ...	52051887758913536	all of the lights
_ImJustJosh	Sun Mar 27 23:21:46 +0000 ...	52148335640969217	All of the lights
Hey_ItsNina	Sun Mar 27 23:31:14 +0000 ...	52150718638981120	- is he autistic?
TeamVania	Sun Mar 27 23:47:46 +0000 ...	52154879573114882	All of the lights
Highskulchick	Mon Mar 28 00:03:24 +0000 ...	52158812681011200	All of the lights
MsIcomeFirst	Sun Mar 27 22:46:12 +0000 ...	52139384643915777	All of the lights
KIIDswiss	Sun Mar 27 23:06:37 +0000 ...	52144520690806784	Here she go again
frescoFiasco038	Sun Mar 27 23:12:36 +0000 ...	52146028014612480	all of the lights
ShowoffPolo	Sun Mar 27 16:49:26 +0000 ...	52049599912873984	@AyeeAyoo She Fell Asl
adambuerger	Sun Mar 27 16:33:02 +0000 ...	52045474533879808	Does anyone know whe

Figure 2: Example Query Table

As discussed previously, Coalmine supports manual searching similar to a typical Boolean search engine. We have added the ability to specify the fields from which to search as well as logical operators. Table 1 denotes the exact fields that the system currently handles. In addition null and non-null operators expressed as "None" which is useful for mass filtering of some criteria such as Latitude and Longitudinal coordinates. For instance a search for "Geo-Lat:!=None" would return all posts where the Latitude field was present meaning the post could be attributed to an actual physical location.

To support these search terms we have implemented a standard Lucene index rather than some more custom systems like TweetQL which in our studies did not scale as well when performing queries across billions of records [28]. Lucene is a Java based solution for fast searching of indexed data. Currently the Coalmine front end interface creates queries through Lucene using PyLucene [29].

Contributions	Location	Followers_count	Profile_image_url
Coordinates	Retweeted Count	Friends_count	Protected
Created_at	Retweeted	Geo_enabled	Screen_name
Hashtags	Retweet Source	ID	Show_all_inline_media
URLS	Text	Translator_enabled	Statuses_count
Mentions	Truncated	Language	Time_zone
Favorited	Contributions	User_name	Text-URL
Geo-Lat	Application	Notifications_enabled	UTC_offset
Geo-Long	Description	Profile_background_color	verified_user
In_Reply_To	Favorites_count	Profile_background_image	Mobile_device_enabled

Table 1: Coalmine - Initial Search Criteria

3. The batch processing environment allows for more complex queries and post-processing. The core of this component, is the Apache Hadoop open-source framework [27].

   Hadoop allows for the efficient processing of large amounts of data. It provides developers with two key sub-components. The first being a distributed file system for the organization, storage, and transfer of data associated with data processing. The second is a distributed process execution environment based on the MapReduce method [7].

   This environment allows for efficient processing of data over multiple machines that are aligned in a cluster configuration, and it can scale as new machines are added to the cluster. Map reduce allows for data to be chopped into tiny pieces and processed by any machine within the cluster. The results of this can be recombined when the processing is complete. To use the framework, the developer has to implement the processing logic specific to their data-sets. The execution process is handled completely within Hadoop.

   On small batches, Hadoop can add considerable overhead, and is not recommended. However, in our work where batches are 10-100 gigabytes in size, the Hadoop execution environment is the only system that we have found that is flexible and efficient at processing such large amounts of data.

4. The final component of the Coalmine system is the live filtering, classification, and alerting functionality. Once a user has made a query, the front end returns the intended results, the results can then be used to form a filter. These filters can be set to run all the time on all incoming data streams. To do so we have created a rudimentary classification system for the incoming data. Since it takes too much time and computational power to filter across multiple posts, our filtering system for now only works on a post by post basis. That is to say, we can setup a filter that will look for all instances of a certain word, but we can not setup a live filter that correlates all instances of a specific tweet across multiple accounts. To do so, would require keeping an enormous amount of state in memory and we leave this for the batch processing component of our system. Currently when our system does find a match using one of our live filters, we report that match using a text based log file in syslog format. It is our plan eventually to produce logs can be read by a standard SIEM (Security Information and Event Manager).

While our work on automated analysis is only in its initial phases we have determined the basic approach for doing so. We first use the NLTK (Natural Language ToolKit)[1] to define the following objects:

- User Profiles: These contain lists of unigrams, bigrams, and complete messages. Unigrams refer to single uses of meaningful words that a user has posted, similarly bigrams are a combination of two words, while complete messages are just that.

1 Natural Language ToolKit a Python module for processing linguistic data: http://www.nltk.org/

- Account Profiles: These contain the complete historical record for an individual user. These profiles consist of only the posts that we have recorded with our tool and as such may not be a complete record of everything an individual user has ever posted.
- Corpus: This consists of all posts that we have ever recorded in our entire data-set.

To make use of this data for automated analysis we typically list all unigrams and bigrams by the frequency at which they occur within a particular account profile. While the exact count of each is useful for determining how often a particular user discusses a topic it is not necessarily all that interesting if we are trying to look for more covert communications.

Using the statistics from an individual users account profile we can start to determine a trend in speech pattern and thus develop a system to look for anomalies in those trends. The same can be accomplished using the complete Corpus as we start to analyze the sentiment of the entire Corpus collected we start to see fluctuations in the usage of specific terms which could eventually lead us to an automated system for analyzing the sentiment of large groups of people.

One of the issues we face as we continue the development of Coalmine is the determination of what's relevant and what's not. Current estimates have stated that SPAM accounts for as much as 50% of all Twitter messages all though that number appears to be receding in recent months [23, 24, 25]. A number of works have been published recently on just this problem. Our current approach utilizes information diversity and user cognition to sort weed out the noise [26]. This method utilizes entropy based calculations which we have already added to the output of Coalmine. These calculations can be used to calculate the diversity of the complete corpus or the posts of an individual user. Our next steps include the addition of entropy distortion minimization algorithms to reduce the number of results. We have a lot of work left to completely integrate this technique, however initial results are promising.

3. DATA COLLECTION ISSUES

Data collection is by far one of the most complicated tasks in our work. We aim to support a number of social media sites eventually, but focused on Twitter at this time. We learned Twitter posed significant problems all on their own, due to both the ever changing methods that they use to distribute and format data, as well as the amount of out of date documentation that does not seem to be maintained by Twitter.

During the course of our work, we have seen the death of Twitter’s Fire-Hose API and White-listing Service, the addition of 0Auth, streaming gzip API, paid API access, new fields, and many more [1, 2, 3, 4, 5]. These changes have created a number of issues for our once simple data collection system. In mid 2011, when Twitter switched to it's REST API, it started limiting the number of requests per hour that a single IP/Username combination was able to make, to 350 per hour. This results in approximately 200 tweets per request for a grand total of 70,000 per application. An application is denoted as any tool written and distributed to multiple users, but using the same account credentials. Twitter also imposes a cap of 20,000 tweet responses an hour for any single IP address. This service is known as the Spritzer REST API. This limitation means that based on a current estimated rate of 200,000,000 tweets occurring daily, and the average application+user+IP combination, being able to receive no more then 480,000 tweets daily (20,000 * 24 hours), normal users can only collect .24% of all tweets. This is a woefully small sample size on which to perform meaningful analysis.

After considerable searching for alternative, free Twitter data sources [19, 20, 21], we eventually developed a system of our own which is now able, using an offset sampling method, to gather nearly all tweets in near real time, without relying on a scrapper. Our development of our own acquisition method was a direct reaction to changes in Twitters terms of service which prevents researchers from redistributing tweet captures [22]. This allows us the benefit of gathering all addition meta-data associated with each tweet, which a typical html scrapper would not provide. We discuss this further in the section on Twitter data structure.

Twitter uses a fairly simplistic method for sampling the stream it gives to users. Twitter says:

> "The status id modulo 100 is taken on each public status, that is, from the Firehose. Modulus value 0 is delivered to Spritzer, and values 0-10 are delivered to Gardenhose. Over a significant period, a 1% and a 10% sample of public statuses is approached. This algorithm, in conjunction with the status id assignment algorithm, will tend to produce a random selection." [6]

What this boils down to, is that for the Spritzer streaming API feed, twitter starts a counter when you connect. The counter starts at 1 and counts to 100 before starting over. The first tweet "1" is sampled and returned to the connecting client. After some basic experimentation we realized that by putting in a simple "sleep 1" command between the start-up of two or more subscriber scripts,we can receive a completely different set of tweets between accounts. To prove this point, we initially set up 5 accounts and manually started our script using each accounts credentials. After running for 24 hours, these 5 subscriber script account combinations collected 16,599,674 unique tweets While it depends greatly on the timing of the startup script and the number of accounts used, the maximum amount of overlapping tweets collected by our scripts has been no greater then 20% This overlap constitutes duplicate tweets that one account may have received that had been previously collected by another account. Since tweets from the streaming API are only available when they are posted we don't need to deal with historical overlap. Running 30 simultaneous subscribing accounts and staggering collection time by 1 second each has reduced the total collection duplication across all accounts after the first two months of collection to be no greater then 12%.

We have since automated our collection system and developed a network architecture using 30 individual IP's in a round robin proxy fashion and a centralized proxy to support the collection of nearly 100% of all tweets. Our assumption of near 100% collection is based on the measured number of non-duplicate tweets versus the sporadically published numbers on number of tweets posted by twitter. Twitter has grown regularly since the first reported numbers in November of 2009 which stated that the site received 27.3 million tweets a day to it's busiest day ever in May, 2 2011 with an estimated 250,000,000 tweets [31, 32, 33, 34]. Our system collects around 260,000,000 unique tweets a day, while by our estimates it is capable of collecting up to 300,000,000. We have also started adding feeds from other services such as Facebook, Youtube, and Linkedin.

4. COALMINE: BOTNET C2 DETECTION CASE STUDY

One of our initial focuses for Coalmine use was the detection of Botnet Command and Control Channels on Twitter. A botnet is a network of infected systems known as “Bots”, based on the word “Robots,” that are remotely controlled for a malicious individual known as the “Botmaster”. Botnets allow attackers to wage war on a much larger scale, which potentially, can be lead from a single system. Typically, they are not employed by the average attacker, but instead by so called nation states, criminal organizations, hacktavists, and those seeking profit [8, 9]. Botnets proliferate for both technical and non-technical reasons. The primary reason is often that, owners of these malicious networks, sell access to them as a service. Like any good businessman, they attempt to attract sales through “beating out the competition” on stats and features. They must therefore grow their networks, by taking over more systems to use as bots.

Another popular reason is perhaps the simplest, revenge. A botmaster will grow their “armies” in order to carry out a vengeful attack. This was proven true during the Low Orbit ION Cannon (LOIC) attacks carried out by the group Anonymous. These occurred during the first quarter of 2010. [10] While that specific incident was non-traditional, in that systems were infected because their owners purposefully downloaded the malicious software, it none-the-less proves the point.

Before we go further, it is prudent to discuss the following common terms, which are used throughout this section.

- ***Botnet***: A network of infected machines known as bots, that are controlled by a malicious entity. This network is used to attack victim systems by exploiting a weakness, such as a maximum services connection limit.
- ***Bot***: Once known as zombie computers, bots are the infected systems that do a botmasters bidding. Bots are modular programs that can be updated with new commands and payloads to help them carry out orders. The difference between a bot and a typical remote controlled system, is the legitimacy of access. Bots spread through infection, while the infection means may vary, the end result is that the owner of a system has not authorized the control.
- ***Botmaster***: Once known as a bothearder, the botmaster is the human or group that controls the activities of a botnet. This individual(s) may not be the one(s) who originally created the malicious software or the botnet itself. These individuals/groups were once motivated by the traditional hacker cultural directive of seeing what can be done. Nowadays, money, power, revenge and politics, are the primary drivers.

- ***Command and Control Channel:*** The command and control channels are the primary systems, sites, and protocols, that allow a botmaster to communicate orders to their army. The channels are typically obscured from normal view through means of obfuscations, encryptions, and a-typical (for the average user) protocols. A botmaster doesn't communicate directly with each node in the army because doing so would be easily traceable.

Like any other network build-out, a botnet has a fairly standard creation procedure consisting of what Zhu, et al., discuss in their work “A Botnet Research Survey” [11] as a four phased approach. While this approach is very basic, it aids in understanding of the process by which a botnet is created.

1. ***Initial Infection***: The exploitation of a system, that allows for it to become infected by a bot program. This stage results in the basic ability to remote control a victim.
2. ***Secondary Infection***: This usually consists of the botmaster sending a command that instructs the bot to download a more sophisticated version of its self, a payload of some sort, or even carry out a simplistic initial attack.
3. ***Malicious Activity***: This is the act that the bot carries out, due to the command and/or payload that was sent. This may be as simple as spreading the infection, or as complex as a distributed computing [5] function.
4. ***Maintenance and Upgrade***: More often then not, a botmaster will send a command to a bot, that includes instructions for upgrading or regular maintenance reporting.

From the previous statements, we now have a botnet ontology of sorts, whereby in common terms we have defined our actors, assets, access methods, motivations, attacks on, and outcomes, using the typical format as shown in Table 2.

Actor	Asset	Access Method
Botmaster	Bot (Zombie)	Infection
Motivation	**Outcomes**	**Attacks On**
Profit, Power, Revenge, Cause	An Army of Malicious Drones	Victim systems/Services/Networks

Table 2: Botnet Ontology Overview

4.1 Coalmine: Botnet Case Study - Detection

As we discuss later, a distinction can be made as to not only the methods used for botnet detection, but also the data sources that feed these methods. In Bailey's, et al. (A Survey of Botnet Technology and Defenses) this distinction has been made for us [12]. The paper describes data-sources as being limited, depending on sensor location. In this way, the internal enterprise network has more access to data-sources, such as DHCP and DNS resolves, then the carrier network provider would normally allow. While we disagree with Bailey, et al. that one network location is better then another, it is better to think of each location as having unique insight into the problem and the fusion of multiple data sources as being a better solution for traditional detection tools.

Currently the most popular tools on the market for detecting botnets don't actually detect the bots at all. Instead, they focus on their command and control channels. Bailey, et al. defines a number of C2 channel detection methodologies. They make the same points that we had in turn discovered, through our own initial testing. The three major competing detection technologies are available, (1) Bothunter, (2) Botsniffer, and (3) Botminer. They all use the same method of “Detection through Cooperative Behaviors.” In their own ways, each of these tools looked for the grouping patterns that occur on these malicious networks, based on either statistical methods or comparative models.

1. Bothunter was funded by the Army Research Office and developed by SRI International. It is a tool which detects bots using Snort as a dialog event generation engine. These events are fed into Bothunters modeling and comparison programs, which look for significant correlations between real events and recorded malicious events [13].

2. Botsniffer attempts to detect botnets using network temporal-spacial detection and passing the output though a series of statistical algorithms that look for grouping behavior [14].
3. Botminer is really an add-on for Botsniffer that allow cross-correlation to be done on the data by breaking it into cluster series. This is very useful for finding channels if you know what an existing one looks like, but can generate a high number of false positives if you do not [15].

4.2 Coalmine: Botnet Case Study – Approach

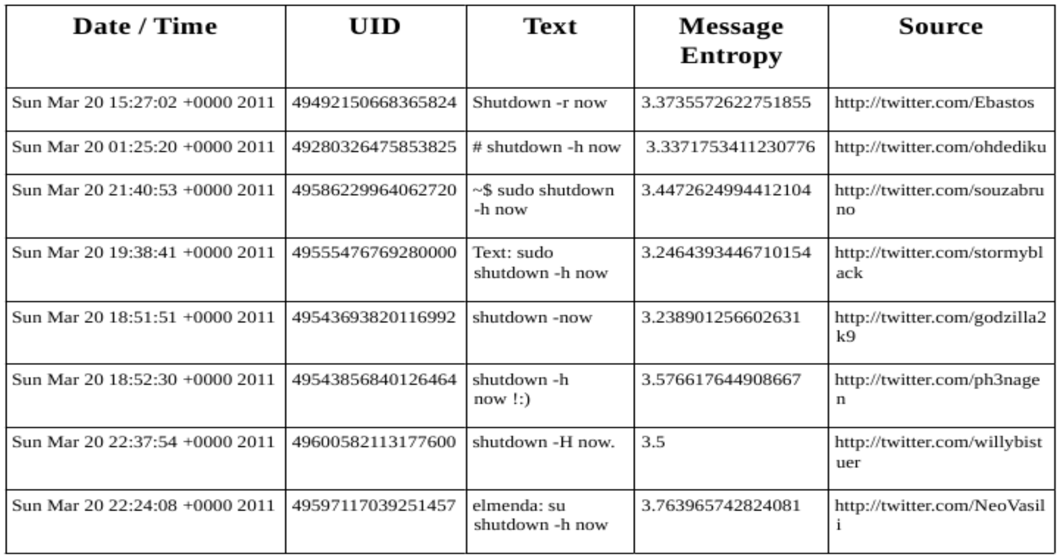
Like most security mechanisms on the market today, current botnet detection methods rely on detection on a local scale. However, as a global Internet community, security professionals must also develop tools that protect the environment as a whole, not just their own small corners. Following this idea, we have focused on Social Networking sites as a C2 Channel data source. Truly, data sources such as Twitter provide a global environment for these malicious networks to communicate and for us to test our tool. The detection/shutdown of these Botnet communications on such networks would, potentially impact many millions of people. Given the scoped initial timeline of our work, we have focused our first attempts on using Coalmine in a way that would detect these C2 channels on Twitter without any modification to our underling code.

Recent trends have shown that Twitter has actually been used as a botnet communication system [16, 17]. Previous works have hypothesized that while careful analysis of individual tweets and Twitter accounts can be done on a small scale, to detect such C2 Channels, it should be possible to potentially automate the process. The approach we used actively gathered social network feeds from Twitter and parsed them, for the tell-tale signs of botnet communications.

4.3 Coalmine: Botnet Case Study – Preliminary Results

The following, Table 3, depicts data that was collected for a 12 hour period on March 20th using the keyword "shutdown." The column for message entropy was calculated by Coalmine on just the contents of the message field using a standard Shannon's Entropy formula. Table 4 shows control messages from a few accounts taken at similar times on the same day. We can verify within reasonable certainty that the messages in Table 3 do not contain botnet C2 traffic because we know the owners of the accounts.

When you compare the entropy values from tables 3 and 4, it is readily obvious that not enough variation occurs between the controls and the potential botnet commands to be considered definitive. More normalization of the messages needs to occur and the entropy scale of 0-8 may need to be tuned to provide more significant differences.



Date / Time	UID	Text	Message Entropy	Source
Sun Mar 20 15:27:02 +0000 2011	49492150668365824	Shutdown -r now	3.3735572622751855	http://twitter.com/Ebastos
Sun Mar 20 01:25:20 +0000 2011	49280326475853825	# shutdown -h now	3.3371753411230776	http://twitter.com/ohdediku
Sun Mar 20 21:40:53 +0000 2011	49586229964062720	~$ sudo shutdown -h now	3.4472624994412104	http://twitter.com/souzabruno
Sun Mar 20 19:38:41 +0000 2011	49555476769280000	Text: sudo shutdown -h now	3.2464393446710154	http://twitter.com/stormyblack
Sun Mar 20 18:51:51 +0000 2011	49543693820116992	shutdown -now	3.238901256602631	http://twitter.com/godzilla2k9
Sun Mar 20 18:52:30 +0000 2011	49543856840126464	shutdown -h now !:)	3.576617644908667	http://twitter.com/ph3nagen
Sun Mar 20 22:37:54 +0000 2011	49600582113177600	shutdown -H now.	3.5	http://twitter.com/willybistuer
Sun Mar 20 22:24:08 +0000 2011	49597117039251457	elmenda: su shutdown -h now	3.763965742824081	http://twitter.com/NeoVasili

Table 3: Coalmine: Botnet Case Study - First Set Query Results

Date / Time	UID	Text	Message Entropy	Source
Sun Mar 20 22:23:50 +0000 2011	49600582113177871	Going to be another late night	3.60341234326	http://twitter.com/ (removed)
Sun Mar 20 22:33:12 +0000 2011	49600582113180800	dslr cam lens cleaning	3.52205520887	http://twitter.com/ (removed)
Sun Mar 20 22:40:20 +0000 2011	49600582113201100	Have to get this report done soon	3.6427741507	http://twitter.com/ (removed)
Sun Mar 20 22:41:01 +0000 2011	49600582113271260	@inlin I hear that	3.47135448701	http://twitter.com/ (removed)

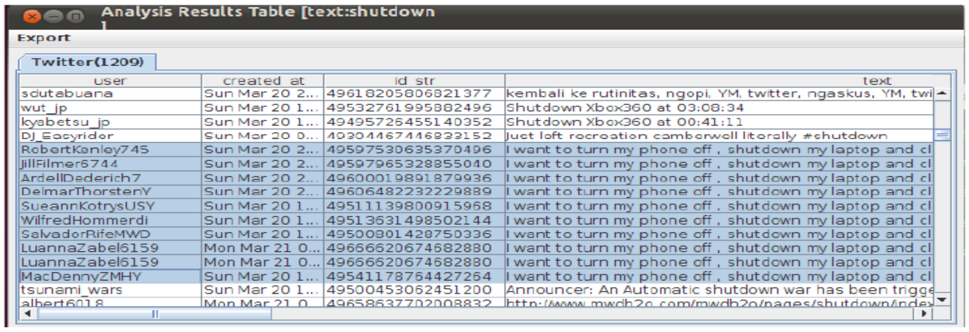
Table 4: Coalmine: Botnet Case Study - First Set Control Group

5. COALMINE: SPAM RESULTS BASED ON BOTNET STUDY

When performing our Botnet C2 Channel discovery case study, we noticed a high volume of identical Tweets from different accounts. These messages were not re-tweets. They were individual posts from what we believed to be unique users. Figure 5 shows one such instance as we looked for the term "Shutdown."

We can assume that these messages with the exact same wording all occurring within a few hours are not a coincidence. Upon manual inspection of the webpages for each of these Twitter accounts, it became readily obvious that these were indeed SPAM related messages. In fact the exact messages that we found were what we believe to be filler messages used to fool some of the anti-SPAM techniques that Twitter uses. Manual inspection showed that all messages within these 9 accounts share the following:

- All messages are similar between accounts, though the order they were posted in varied
- Posts appear to occur about once every minute
- Account Photos all look like professional photographs. Further inspection using Google's image comparison showed that they were from stock photo sites.
- Each account had a largely disproportionate number of followers and following for each account.
- The wording of each tweet consisted of very choppy English.
- Accounts either had no locations associated with them or were set to European and South American countries, where English was not the primary language.
- Finally all of the accounts had one or more tweets with links to http://pyq.me which redirected to an IPAD related phishing sub-page called ipad2a.



user	created_at	id_str	text
sdutabuana	Sun Mar 20 2...	49618205806821377	kembali ke rutinitas, ngopi, YM, twitter, ngaskus, YM, twi
wut_jp	Sun Mar 20 1...	49532761995882496	Shutdown Xbox360 at 03:08:34
kyabetsu_jp	Sun Mar 20 1...	49495726455140352	Shutdown Xbox360 at 00:41:11
DJ_Easyrider	Sun Mar 20 0...	49304467446833152	Just left recreation camberwell literally #shutdown
RobertKenley745	Sun Mar 20 2...	49597530635370496	I want to turn my phone off , shutdown my laptop and cl
JillFilmer6744	Sun Mar 20 2...	49597965328855040	I want to turn my phone off , shutdown my laptop and cl
ArdellDederich7	Sun Mar 20 2...	49600019891879936	I want to turn my phone off , shutdown my laptop and cl
DelmarThorstenY	Sun Mar 20 2...	49606482232229889	I want to turn my phone off , shutdown my laptop and cl
SueannKotrysUSY	Sun Mar 20 1...	49511139800915968	I want to turn my phone off , shutdown my laptop and cl
WilfredHommerdi	Sun Mar 20 1...	49513631498502144	I want to turn my phone off , shutdown my laptop and cl
SalvadorRifeMWD	Sun Mar 20 1...	49500801428750336	I want to turn my phone off , shutdown my laptop and cl
LuannaZabel6159	Mon Mar 21 0...	49666620674682880	I want to turn my phone off , shutdown my laptop and cl
LuannaZabel6159	Mon Mar 21 0...	49666620674682880	I want to turn my phone off , shutdown my laptop and cl
MacDennyZMHY	Sun Mar 20 1...	49541178764427264	I want to turn my phone off , shutdown my laptop and cl
tsunami_wars	Sun Mar 20 1...	49500453062451200	Announcer: An Automatic shutdown war has been trigge
albert6018	Mon Mar 21 0...	49658637702008832	http://www.mwdb2o.com/mwdb2o/pages/shutdown/index

Figure 5: Coalmine: SPAM Case Study - Sample Search Results for Term "Shutdown"

6. CONCLUSION

In this paper, we present Coalmine a social network data-mining system. We described the overall architecture of Coalmine including the capture, storage and search components. Coalmine has shown to be a promising security tool for Social Network data-mining. It currently provides a means for large scale collection of Twitter data, in a manageable format, as well as a framework for analyzing this data. Our next major steps will incorporate a number of techniques to better analyze the connectivity of users and the interaction of opinion leaders with the masses. In addition, we will continue our develop of a method that will incorporate some of the interesting things that we learned from our discovery of SPAM channels, such as the large disproportions of followers and following, into a sophisticated map reducing algorithm.

7. CITATIONS

[1] Christina Warren, “Measureing Social Media: Who Has Access to the Firehose?,” Machable.com, March 13 2011, http://mashable.com/2011/03/13/sxsw-smaroi/

[2] Mike Melanson, “Twitter Kills the API Whitelist: What it Means for Developers and Innovation,” February 11 2011, http://www.readwriteweb.com/archives/twitter_kills_the_api_whitelist_what_it_means_for.php

[3] Joab Jackson, “Twitter Now Using Oauth authentication for Third Party Apps,” Computer World UK, September 1, 2010, http://www.computerworlduk.com/news/security/3237659/twitter-now-using-oauth-authentication-for-third-party-apps/

[4] Arne Roomann-Kurrik, “Announcing gzip Compression for Streaming API's,” Twitter Developers Feed, Jan 20 2012, https://dev.twitter.com/blog/announcing-gzip-compression-streaming-apis

[5] Ryan Sarver, “Twitter + Gnip Partnership,” Twitter API Announcments, Nov 17 2011, http://groups.google.com/group/twitter-api-announce/browse_thread/thread/e0a2006b0698f880?pli=1

[6] Twitter, “Streaming API Concepts: Sampling,” Twitter.com, Dec 2012, https://dev.twitter.com/docs/streaming-api/concepts#sampling

[7] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design \& Implementation - Volume 6* (OSDI'04), Vol. 6. USENIX Association, Berkeley, CA, USA, 10-10.

[8] B. Saha and A, Gairola, “Botnet: An overview,” CERT-In White PaperCIWP-2005-05, 2005.

[9] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, “A multifaceted approach to understanding the botnet phenomenon,” in Proc. 6th ACM SIGCOMM Conference on Internet Measurement (IMC’06), 2006, pp. 41–52.

[10] A. Pras, A. Sperotto, et. al., “Attacks by Anonymous WikiLeaks Proponents not Anonymous,” Design and Analysis of Communication Systems Group (DACS) CTIT Technical Report, 2010, pp. 1-10

[11] Zhaosheng Zhu, Guohan Lu, Yan Chen, Z.J. Fu, P. Roberts, and Keesook Han. “Botnet Research Survey,” pages 967–972, 28 2008-Aug. 1 2008.

[12] M. Bailey, E. Cooke, et al. “A Survey of Botnet Technology and Defenses,” Cyber Applications & Technology Conference for Homeland Security, Conference Proceedings, IEEE, 2009

[13] G. Gu, P. Porras, V. Yegneswaran, M. Frog, W. Lee. “BotHunter: Detecting malware infection through ids-driven dialog correlation”. In Proceedings of the 16th USENIX Security Symposium (Security’07), Boston, MA, August 2007.

[14] G. Gu, J. Zhang, and W. Lee. “BotSniffer: Detecting botnet command and control channels in network traffic,” Proceedings, 15th Annual Network & Distributed System Security Symposium (NDSS’08), San Diego, CA, 2008.

[15] G. Gu, R. Perdisci, junjie Zhang, and W. Lee. “BotMiner: Clustering analysis of network traffic for protocol and structure-independent botnet detection,” Proceedings, 17th USENIX Security Symposium (Security’08), San Jose, CA, 2008.

[16] Ryan Singel, “Hackers Use Twitter to Control Botnet,” Wired Magazine, August 2009. http://www.wired.com/threatlevel/2009/08/botnet-tweets/

[17] Jose Nazario, “Twitter-Based Botnet Command Channel,” Arbor Networks: Arbor SERT, DDoS and Security Reports: The Arbor Networks Security Blog, August 2009. http://ddos.arbornetworks.com/2009/08/twitter-based-botnet-command-channel/

[18] Donald Triner, “Publicaly Available Social Media Monitoring and Situational Awareness Initiative,” Office of Operations Coordination and Planning: Departmetn of Homeland Security, June 22 2010.

[19] Graeme McMillan, “Tweet Eternal: Pros and Cons of the Library of Congress Twitter Archive,” Time Magazine: Technland, December 8, 2011

[20] J. Yang, J. Leskovec, “Stanford Twitter data-set archive 6-2009 to 12-2009,” Stanford University, 2011: http://snap.stanford.edu/data/bigdata/twitter7/

[21] Minas Gjoka, Maciej Kurant, Carter T. Butts, Athina Markopoulou, “Walking in Facebook: A Case Study of Unbiased Sampling of OSNs: Data-set.” Proceedings of IEEE INFOCOM '10, San Diego CA, March 2010: http://odysseas.calit2.uci.edu/doku.php/public:online_social_networks

[22] Audrey Watters, “How Recent Changes to Twitter's Terms of Service Might Hurt Academic Research,” Read Write Web, March 3rd 2011: http://www.readwriteweb.com/archives/how_recent_changes_to_twitters_terms_of_service_mi.php

[23] Douglas Main, “How Much of Twitter Is Spam?,” Popular Mechanics, August 4 2011: http://www.popularmechanics.com/technology/how-much-of-twitter-is-spam

[24] Chao Yang, Robert Harkreader, Guofei Gu. "Die Free or Live Hard? Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers." Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID 2011), Menlo Park, California, September 2011.

[25] Jonghyuk Song, Sangho Lee, Jong Kim. “Spam Filtering in Twitter using Sender-Receiver Relationship,” Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID 2011), Menlo Park, California, September 2011.

[26] Munmun De Choudhury, Scott Counts, and Mary Czerwinski. 2011. Identifying relevant social media content: leveraging information diversity and user cognition. In Proceedings of the 22nd ACM conference on Hypertext and hypermedia (HT '11). ACM, New York, NY, USA, 161-170. DOI=10.1145/1995966.1995990 http://doi.acm.org/10.1145/1995966.1995990

[27] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler., “The Hadoop File System,” IEEE 978-1-4244-7153-9 2010

[28] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. 2012. Processing and visualizing the data in tweets. *SIGMOD Rec.* 40, 4 (January 2012), 21-27. DOI=10.1145/2094114.2094120 http://doi.acm.org/10.1145/2094114.2094120

[29] Yonik Seeley, “Full-Text Search with Lucene,” ApacheCon, May 2, 2007

[30] Rune Halvorsen, Christopher Schierkolk., “Tweetstream: simple Twitter Streaming API Access,” Version 1.1.1, 2011: http://pypi.python.org/pypi/tweetstream

[31] “Osama bin Laden's death sets Twitter record for sustained rate of Tweets”, LA Times, May 2, 2011: http://latimesblogs.latimes.com/technology/2011/05/twitter-sets-tweets-per-second-record-on-news-of-osama-bin-ladens-death.html

[32] Kevin Weil, “Measuring Tweets,” Twitters Blog, February 22, 2010: http://blog.twitter.com/2010/02/measuring-tweets.html

[33] Brian Solis, “Guess How Many Tweets Fly Across Twitter Each Day,” Blog, November 15, 2009: http://www.briansolis.com/2009/11/guess-how-many-tweets-fly-across-twitter-each-day/

[34] Twitter, “#Numbers,” Twitters Blog, March 14, 2011: http://blog.twitter.com/2011/03/numbers.html