MMM Social Financial Network: **Machine Learning in Fraud Detection**

Joshua Uduehi
School of Informatics & Computing
Indiana University-Bloomington
Bloomington, IN
juduehi@umail.iu.edu

Jasmine Bowers, Pankaj Chand, Kevin Butler
FICS Lab
University of Florida
Gainesville, FL
jdbowers08@gmail.com, pchand@ufl.edu,
butler@ufl.edu,

Abstract - Financial fraud is a prevalent issue online. Ponzi schemes in particular can range on a scale from small and benign to large and heinous enough to cripple a nation's GDP. Combatting Ponzi schemes effectively means understanding how they are structured and function. One way to do that is using the framework of a network of well known Ponzi scheme sites like the MMM Social Financial Network and analysis of the sites in the network with a machine learning approach. Through the use of two processing pipelines, a generated list of domain names were scraped for content and input to a Naïve Bayes *Scikit-learn* classifier as either positively identified as a Ponzi site or negatively for a non-Ponzi site.
Keywords *- Machine learning, Financial fraud, Naïve Bayes, Ponzi scheme*

1. INTRODUCTION

Ponzi schemes are defined as fraudulent investments, in which the operator pays returns to investors from new money paid by new investors. This type of scheme was first practiced by Charles Ponzi the 1920s but still occur around the world, where they can have disastrous effects on a nation's economy. Ponzi schemes often share common characteristics. One of those is the promise of well above-average return on investment (ROI). Another characteristic is the use vague verbal guises, language that attempts to dissuade doubts about the site's legitimacy. An example of this is a high-yield investment program (HYIP), which is quite similar to the aforementioned ROI in terms of functionality. Advertising a wide array of investment vehicles is yet another characteristic common in Ponzi schemes.

Detecting Ponzi schemes is important to the financial security on developing nations. That is, investigating the global impact on third world nations and current state of well-known Ponzi sites helps in combatting such sites like the MMM Social Financial Network. Doing so provides a case example of what a Ponzi scheme would usually look and operate like. More so, provides a framework in which to understand Ponzi schemes.

However, properly detecting can prove to be difficult for a two reasons. First, establishing a ground truth as a baseline is dependent on experts in the field. Second, working from that backed ground truth to identifying what constitutes a Ponzi scheme depends on how extensive the classification goes. Both these rationale play a part in characterization of known Ponzi sites and potential Ponzi sites.

The problem of Ponzi schemes has likely not been solved before due to many factors.
One is that Ponzi schemes are ever evolving and dynamically altering the structure of their site content and financial infrastructure, which can difficult to track. Another is that at times, it is too late to identify them once they have grown to a national scale. Either reason contributes to the ever-elusive issue of cracking down on all Ponzi schemes outright.

In this paper, we seek to identity Ponzi schemes based on a textual analysis of known fraudulent schemes identified and collected from domain experts and non-malicious sites. Then, we use these sites as training and testing data in the *Scikit-learn* module. Following this, we generate a list of possible domain names. This list was subsequently used to scrape the domain names for their content so as to identify them against the first set of sites as either Ponzi or non-Ponzi sites. There were limitations to this paper. One included the variations in domain name structure, like prefixes and suffixes. Another was the use of MMM sites as the primary Ponzi scheme samples and not other known or documented Ponzi schemes. Lastly, using the index pages of each site and not the entire site itself was another limitation.

The remainder of the paper is organized as follows. In Section 2, we discuss the process through which the Naïve Bayes classifier was built. Section 3 details the results of the classifier and its metrics. In Section 4, we discuss the meaning of the results and how related work compares to the results.

Sections 5 and 6 are where we outline the conclusion and future work, respectively.

II. SCIKIT-LEARN LIBRARY: PYTHON MACHINE LEARNING

Naïve Bayes classification serves as a baseline methodology for categorizing text. In terms of machine learning, this classifier is a simple approach in probability based on assumptions. Functionally, classification occurs when pre-processing data is trained and then compared against test data so as to result in a measure of its accuracy. How much data to train and test is dependent on the situation, but usually there is a threshold to train at least 60% of data against the remainder of data.

In performing the classification properly for this paper, construction consisted of 2 joint pipelines. The first pipeline started with 30 URLs and output a classifier based on the Naïve Bayes approach. The second pipeline consisted of a generated list of possible domain names and output text content of valid domains and whether or not these sites content is similar or identical to a Ponzi scheme site or not. As a result through both of these pipelines, the accuracy of the *Scikit-learn* classifier against test data will be evaluated.

A. Building the classifier

A major component of constructing the classifier is formatting the corpus in a way that is readable and iterable. Doing so requires the use of *html2text*. *Html2text* is a Python script that converts HTML files to text ('.txt') files. This conversion proves very helpful given the structure of the module.

1) Retrieval: Before using the *Scikit-learn* model, we used another model. That model is the natural language Toolkit, or *NLTK*, in Python using the Naïve Bayes approach. This was done to show the most informative features present in the preliminary data, which is comprised of 30 websites. The results of this model are shown in *Table 3*.

Following this, we collected URLs of 30 websites. More specifically, collect the landing/index page of each site. Of those 30 sites, 15 of them are non-Ponzi sites, such as the Lion Cub Scouts and AMC Theaters sites, and other 15 are divided into different types of Ponzi schemes sites: 8 are of the MMM Social Financial Network, which are verified by domain experts in the field, 7 are of similar language and text structure to the MMM sites, but each from different organizations [1].

2) Parsing: After retrieving the HTML of the index page of each site, we saved them to file. Then, the retrieved HTML was converted to text ('.txt') file using *html2text* python script. Subsequently, we saved them to a specified folder in the 'nltk_data/corpora' directory. The specified corpus was a folder named 'ponzi_schemes'.

3) Analysis: We arranged the files through the *Scikit-learn* module involved feature extraction and text tokenization. More so, transforming documents into feature vectors, dividing the number of occurrences of a word in a document by the total number of words in a document, which can be downscaled, as well as filtering out stopwords. This involved tokenizing files based on which folder they were in based on binary designation. Designation was either a value of 0 or 1.

4) Classification: In training the classifier, the multinomial variation of the Naïve Bayes approach (*MultinomialNB*) in *Scikit-learn* was imported to handle word counting. Another step in the training process was configuring classifier with training data (80% of corpus) and test data (20% of corpus). Assembling a pipeline that consists of the division of each word occurrence in a document by the total words in that document (*CountVectorizer*), the weights of these words were downscaled with high frequencies across documents in the corpus (*TfidfTransformer*), and prediction input classification most suitable for word counts (*MultinomialNB*). Metrics of the precision and recall capabilities and a confusion matrix are in the *Table 1* and *Table 2*, respectively.

B. Testing the classifier

There are various avenues through which to retrieve DNS information. *Nslookup* is a command line tool that inputs the argument (nslookup, *name_of_url*) and outputs the DNS available via that URL. The format is *nslookup*, and then the URL without the prefixed protocol requests HTTP/HTTPS. Although usually used in the Terminal application in MacOS and Windows OS, there is a way to call *nslookup* in Python.

Attaining specific information from web page is a task many program can perform. *Scrapy* is a tool does so via Python interface. That is, using Python files that are then called in the terminal command line through the syntax *scrapy crawl file_name* to start crawling the specified URLs in the Python file. The output can then be saved to file formats such as '.json', '.jl', '.csv', etc. and transferred to a text ('.txt') file.

1) Retrieval: To begin, we generated possible domain name prefixes and suffixes via Microsoft Excel based on domain name variations of 8 MMM sites used building the classifier and do so for 196 countries around the world. Working from a Microsoft Excel workbook, we used *openpyxl*, a Python library to read and write Excel files, to import document with cell values into Python. Then, we iterated each value in the 8 columns through the *nslookup* function in the Python file. Lastly, we specified output to be DNS output from domain name. Furthermore, that if output's argument length is less than or equal to 2, then to the result would return 0. If not, then it returns the country's domain

name in that column. These values were put on a separate sheet, but same workbook, as the first sheet and copied to a '.txt' file. To account for different protocols in the addresses, both HTTP and HTTPS request versions of each of the 160 domain names was constructed, resulting in 320 possible domain names.

2) Parsing: Spiders are tools of *Scrapy* that retrieve specified elements of HTML of URL(s). Essentially, spiders in *Scrapy* are the Python files with defined tags and elements to grab from the retrieved URL(s). For the 320 possible domains, we used spider file to crawl the domains for title, resolved URL, and paragraph text content. This was done to more easily identify each domain's content.

3) Analysis: Domains recognized by *Scrapy* were parsed and saved to file. Scraped sites were consolidated form a JSON file a to single '.txt' file. Files that were not scraped were converted from HTML files to '.txt' files (*html2text*) and grouped into a single corpus folder. Afterwards, we input the frequency of verified Ponzi domains into a Microsoft Excel worksheet with a list of 196 countries. The map of this frequency is shown in *Figure 1*.

4) Classification: After the analysis phase, we reformatted the data using triple quotes so as to assimilate to the list function in the classifier. Ultimately, this was done to test the corpus of valid and verified sites' content against the data already present in Naïve Bayes classifier. Distribution of the positive and negative classification values can be seen in *Figure 2*.

III. RESULTS

	Predicted NO	**Predicted YES**
Actual NO	TN = 10	FP = 2
Actual YES	FN = 0	TP = 12
TN = True negative **FN = False negative**		**FP = False positive** **TP = True positive**

Table 1. Confusion matrix for Scikit-learn classifier

	Precision	**Recall**	**F-1 score**	**Support**
Negative	1.00	0.83	0.91	12
Positive	0.86	1.00	0.92	12
Avg./Total	0.93	0.92	0.92	24

Table 2. Precision metrics of Scikit-learn classifier

1.0	*Most Informative features*	
	simple = True	*pos : neg = 8.0 : 1.0*
	good = True	*pos : neg = 8.0 : 1.0*
	member = True	*pos : neg = 6.9 : 1.0*
	sure = True	*pos : neg = 6.9 : 1.0*
	times = True	*pos : neg = 6.9 : 1.0*
	possible = True	*pos : neg = 6.9 : 1.0*
	something = True	*pos : neg = 5.9 : 1.0*
	several = True	*pos : neg = 5.9 : 1.0*
	country = True	*pos : neg = 5.9: 1.0*
	read = True	*pos : neg = 5.9 : 1.0*

Table 3. Most informative features of NLTK classifier

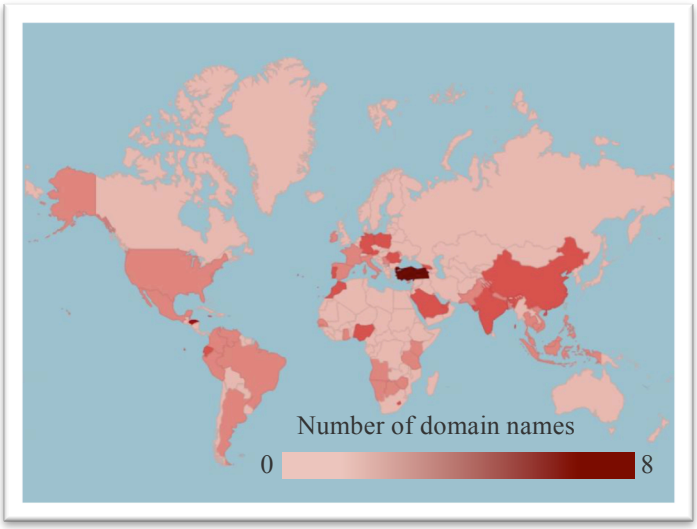

Figure 1. Map of Ponzi site domain name frequency

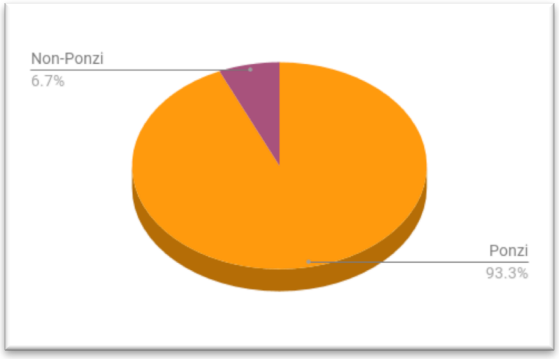

Figure 2. Naïve Bayes classification of 211 valid domains

From the 320 possible domain combination bred out of he HTTP/HTTPS combinations, 222 URLs were output to a JSON file. Out of the 222 domains found by Scrapy, excluding duplicates, 76 verified domains were MMM sites, 5 were similar to MMM sites, and 44 were deemed undefined because their content was non-Ponzi related, and 11 domain names did not exist. The remaining was found to be duplicate URLs.

IV. DISCUSSION AND RELATED WORK

A. Discussion

From the results shown, the MMM sites are present in a sizable portion of the world, according to the data. Although not entirely comprehensive, this provides an ample overview of the frequency of the sites in different countries and the identical/similar structure of the each site's text content. Determining how to further detect these types of online schemes on a larger scale can aid in combatting their presence in developing countries and possibly preventing these Ponzi schemes from growing so far as to cripple that nation's GDP or at the very least, maintain its infrastructure for a lengthy amount of time.

However, there were problems that occurred in this paper. One was the instance of false positives in the second pipeline. At the end of the process, there was an issue of unregistered domains being classified as positive for Ponzi schemes. Another issue was the output of the *nslookup* function and the length for which to properly measure valid DNS output. Both can be deemed as possible sources of error. Nevertheless, I was able to perform predictive analytics using the *Scikit-learn* and *NLTK* modules.

B. Related Work

Characterization of fraudulent online content has ben performed in previous literature before. The *Mallodroid* framework served as a way to survey a large array of content, specifically Android applications, for TLS vulnerability. What the framework found was that approximately 57% of the analyzed applications overrode default certificate verification routines, rendering users in the countries these applications are vulnerable when it comes to mobile banking because of a lack of robust security [4]. Similarly, surveying countries for stationed MMM sites can serve as a framework for statistical analysis of these sites.

In addition to identifying how prevalent Ponzi scheme sites can be, it is equally important to acquire a snapshot of the current state of such business. Specifically, the current state of High Yield Investment Programs (HYIPs) should be attained. While there is the common thread of investigating how HYIP and the like work, this paper does not delve into much depth about the financial analytics and transactions that occur as a result of these fraudulent vendors as much as Moore's et al. work [3]. Even so, the overview of HYIPs specifically points out the different types of cryptocurrencies and the financial risk Ponzi scheme sites take when utilized. That is, describing the cryptocurrency climate as it stands currently.

From the work proposed by Kanich et al., spam-marketing profits are a problem that requires understanding of its function before mitigation can occur. That is, the conversion rate and delivery of spam needs to be understood. This reason is why the *Storm* malware was constructed to act as a botnet to exemplify such behavior through the use of two distinct campaigns: one disguised as a postcard site, the other as a pharmacy site through 500 million spam messages. *Storm* spreads itself from user to user, providing each with spam. As a result, the botnet was able to acquire approximately $7,000 per day from the pharmacy campaign and able to release around 3,500 new bots per day in the postcard campaign. The scale of this project [2] differs from this paper in that the examination of the presence of these sites cast a wide net, but was ultimately smaller scale. Additionally, there was not an analysis of spam messages and propagation, but rather Ponzi scheme sites themselves and their site content.

V. CONCLUSION

This paper focused on taking a machine learning approach to identifying Ponzi schemes in site content. Identification of these Ponzi scheme sites was based on common characteristics analyzed between the each sites' online content. Doing so provides a ground truth for textual analysis of fraudulent sites like the MMM Social Financial Network. Ultimately the data shown can serve as a jumping off point for further analysis of Ponzi schemes.

VI. FUTURE WORK

In future research, what could be explored is the analysis of how cryptocurrencies such as Bitcoin and Ethereum are incorporated into the infrastructure of the sites' financial transactions. Moreover, a more thorough examination of the eCommerce functionality of the Ponzi scheme sites. Additionally, a more comprehensive survey of known Ponzi scheme sites would aid in classification of sites in a larger scale web crawler, like crawling more than the index page. Having more prefix and suffix variations in domain names

would be frugal in not just scouring for MMM sites, but other Ponzi sites as well. Casting a wider, more intricate net would be interesting to see.

ACKNOWLEDGMENTS:

We wish to thank Logan Blue, Joseph Choi, and Luis Vargas for their technical expertise and scientific guidance. We also thank Christopher Patton and Grant Hernandez for their sample code in Python. The CRA-W DREU program and the National Science Foundation under grant CNS-1540217 have supported this research.

REFERENCES:

[1] Ibenegbu, George (2017, March). Latest Ponzi Schemes in Nigeria 2017s[Web blog post]. Retrieved June 7, 2017, from https://www.naij.com/1094594-latest-ponzi-schemes-nigeria-2017.html.

[2] Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., & Savage, S. (2008, October). Spamalytics: An empirical analysis of spam marketing conversion. In CCS'08 (pp. 3-14). ACM.

[3] Moore, T., Han, J., & Clayton, R. (2012, February). The Postmodern Ponzi Scheme: Empirical Analysis of High-Yield Investment Programs. In Financial Cryptography (Vol. 7397, pp. 41-56).

[4] Reaves, B., Scaife, N., Bates, A. M., Traynor, P., & Butler, K. R. (2015, August). Mo(bile) Money, Mo (bile) Problems: Analysis of Branchless Banking Applications in the Developing World. In USENIX Security Symposium (pp. 17-32).