

Spam Detection: A Syntax and Semantic-based Approach

Ray R. Hashemi¹, Mahmood Bahar², Hai Nguyen¹ and Kongdon D. Tift¹

¹Department of Computer Science
Armstrong Atlantic State University
11935 Abercorn Street
Savannah, GA 31419
hashemra@mail.armstrong.edu

²Department of Physics
Teachers Training University
Tehran, Iran

Abstract

Spams are electronic junk mails that are costly for the individuals and organizations who receive them and Internet service providers who handle them. They also highly contribute to the lack of trust in e-mail service by the users, intrusion of privacy, and fraud. Spam detectors with a wide range of effectiveness are exist in the market. To the best of our knowledge, none of the existing spam detectors uses a semantic-based approach in detection of spams. In this paper, we present Semantix-Plus which uses both semantic-based indications and syntax-based indications to convert a given e-mail into a 10-variable record. The detector then uses a backpropagation neural network for the classification of the e-mail. We have compared the behavior of the Semantix-Plus with the spam detectors used by Google, Yahoo, and AASU mail servers for a limited number of spams. The results reveal that the Semantix-Plus has a superior detection power to its counterparts.

Keywords: *Spam, Spam Detector, Semantic-based patterns, and Syntax-based patterns.*

1. Introduction

Spam is referred to as “electronic junk mail” [1]. Over 76 billion spams are floating in the Internet that

collectively make 60% of all the internet e-mails. The storage need of spams is close to 10 petabytes [2, 3]. The reader needs to be reminded that these statistics is nearly 2 years old. Although spams are almost cost free to the spammers, but they are costly for the individuals and organizations who receive them and Internet service providers who handle them [4]. The cost is more compounding when the spams are received through mobile devices[5] Spams are also to be blamed for (1) lack of trust in e-mail service by the users, (2) intrusion of privacy, and (3) fraud –to name a few.

One of the tools available to fight back the spammers is use of spam detectors. Spam detector is a software that is able to identify a spam and then the detected spam may be stopped in different levels of its travel through the Internet. A good number of spam detectors exist and they may be grouped in five general categories of *token-based*, *list-based*, *Initiation-based*, *Sender authentication-based*, and *Social Network-based* [6, 7] . The categorization is based on the methodologies employed by the spam detector.

Token-based detectors check the body and/or the subject of the e-mail for specific tokens or a group of tokens. A token of interest is usually a keyword or a string of keywords that are used commonly in spams.

List-based detectors check the header of a given e-mail to determine whether to include the e-mail address of the given e-mail in a white, gray, or black list for future use. All lists are dynamically built and can grow. Any incoming e-mail that its address is in the black list is filtered by the spam detector. The e-mail addresses of those e-mails that are suspicious but the detector is not sure yet, are placed in the gray list. Gradually, the e-mail addresses in the gray list migrate to either black or white list.

Initiation-based detectors demand some extra manual work (initiation) from any one who wants to send an e-mail to the owner of the detector. Of course, the initiation takes place only for the first e-mail of every sender. After the initiation process is successfully complete, the e-mail address of the sender is kept in a white list.

Sender Authentication-based detectors register a set of legitimate e-mail servers and accept e-mails only from the registered e-mail servers.

Social Network-based detectors build a directed-edge graph out of the incoming and outgoing e-mails of a user. The clustering coefficient of the graph's subcomponents are calculated. The spammer graph's subcomponents have a low coefficient where as the non-spammer graph's subcomponents have a high coefficient [8]

Recently, there are some voices in the scientific community suggesting that the best solution for detecting a spam is to treat an e-mail as a document and then try to use document classification techniques. Our spam detector that is named *Semantix-Plus* is influenced by this suggestion and, first, it tries to identify semantic-based patterns and then quantify a given e-mail based on discovery of those patterns. To the best of our knowledge, a semantic-based attempt has not been exercised in detection of spams. In addition to semantic patterns, our methodology uses syntax-based indications for the quantification of an e-mail (that is the reason for adding "Plus" to the name of the methodology.)

The results of e-mail quantification are used to classify the e-mail. To do so, a Neural Network with a backpropagation training paradigm is built and used [9, 10].

The rest of the paper is organized as follow. The methodology is presented in section two. The results are the subject of section three. The conclusion and future research are covered in section four.

2. Methodology

Before any action be taken on the e-mail, we create a *cleaned* version of the e-mail. The cleaned e-mail is created in two steps. In the first step, all the html tags, if any, are removed from the e-mail. The outcome is referred to as a *filtered* e-mail. In the second step all the tokens that include a punctuation mark (in this paper, punctuation mark refers to all the non-alphanumeric characters that can be found on a keyboard) are removed from the filtered e-mail and the outcome is a cleaned e-mail. In an e-mail, there are syntax-based and semantic-based indicators that may indicate that an e-mail is a spam. We use filtered e-mail to find some of these indicators and cleaned e-mail for others. The details of both types of indicators and their identification process are discussed in the following subsections.

3.1 Syntax-based Indicators

The syntax-based indicators deal with the properties of individual tokens in an e-mail. We are interested in those tokens that:

1. start with a punctuation mark continues with all capital letters (PAC indicators),
2. include at least one punctuation mark and no capital letters (PM indicators),
3. made up of all capital letters (AC indicators)
4. include at least one alphabet letter form a non-English alphabet set (NE indicators),
5. are misspelled (MS indicators).

The filtered version of a given e-mail is used to find the first two types of tokens (above) and the cleaned version of the e-mail is used to identify the rest of them. Finding the first four indicators in the list is straight forward and they can be found using regular expressions. However, the last one is different and need an explanation.

It is a common practice for the author of a spam to misspell a token (or tokens), break a token into two ore more pieces, or connect two or more tokens together, purposely. The end results of all the cases are a set of tokens that are not valid English words (almost all the time). The safest way to identify these tokens is to check them against an English dictionary. This is the path that is taken in this paper.

One may ask that why PM, PAC, and NE indicators are needed while they are identifiable by MS? The answer is that our observations suggest the otherwise. Reader needs to be reminded that the MS indicators

are identified after PM, PAC, and NE indicators are removed.

Each syntax-based indicator has two properties: type and weight. The type represents the nature of the indicator and weight represents its significance. For the above five syntax-based indicators the types and weights are: (AC, w_{ac}), (PAC, w_{pac}), (PM, w_{pm}), (NE, w_{ne}), and (MS, w_{ms}), respectively.

3.2 Semantic-based Indicators

These indicators set apart the Syntax-Plus detector from the existing detectors and they deal with the semantic of every sentence in a cleaned e-mail. To explain it further, we start with some terminology and then explain how to capture the semantic of a sentence in form of a pattern.

Equivalent set of a token. Let t_i be a token in sentence S . Token t_j is equivalent of t_i , if t_j can replace t_i without changing the message conveyed by S . Suppose the general message conveyed by S is “selling jewelry”. In such a sentence the token “diamond” in a phrase “diamonds are for sale”, for example, may have the equivalent set of: “gold”, “precious stone”, “emerald”, “ruby”, etc. If the token is a verb, then all of its synonyms make its equivalent set.

The phrase that contains a token dictates an equivalent set for the token. However, identifying a subset from all the possible equivalents of the token that is proper for a given phrase is very difficult if not impossible. As a result, replacing token t_i with some of its equivalents may not have any meaning in the context of a given phrase, while it makes perfect sense for other equivalents of t_i .

Pattern. Let S be a sentence that is made up of N tokens, $S = \{t_1, \dots, t_n\}$. Let S' be a proper subset of tokens in S and it is identified by a trained individual (a lighter version of domain expert) to capture the semantics of S . If every token in set $(S-S')$ be removed from S and be replaced by a “-” to preserve the gaps among tokens in S , then S is changed into a *pattern*. Therefore, pattern is a spaced set of tokens in a sentence that collectively reflect the semantics of the sentence. The first token of the pattern is named the *head* of the pattern.

Let P be a pattern of “ $—g_i—g_k—g_m—$ ”, where “ $—$ ” represents a gap made up of a number of contiguous

spaces and g_i represents a group of contiguous tokens. Let groups g_i , g_k , and g_m be also made up of tokens $\{t_i^1, t_i^2\}$, $\{t_k^1\}$, and $\{t_m^1, t_m^2, t_m^3\}$, respectively.

Table 1: A set of patterns generated from a given pattern and the equivalent set of the tokens participated in the original pattern.

Patterns
$—t_i^1, t_i^2—t_k^1—t_m^1, t_m^2, t_m^3—$ (original)
$—t_i^1, t_i^2—t_k^1—t_m^1, y_{t_i^1}^1, t_m^3—$
$—y_{t_i^1}^1, t_i^2—t_k^1—t_m^1, t_m^2, t_m^3—$
$—y_{t_i^1}^2, t_i^2—t_k^1—t_m^1, t_m^2, t_m^3—$
$—y_{t_i^1}^3, t_i^2—t_k^1—t_m^1, t_m^2, t_m^3—$
$—y_{t_i^1}^1, t_i^2—t_k^1—t_m^1, y_{t_i^2}^1, t_m^3—$
$—y_{t_i^1}^2, t_i^2—t_k^1—t_m^1, y_{t_i^2}^1, t_m^3—$
$—y_{t_i^1}^3, t_i^2—t_k^1—t_m^1, y_{t_i^2}^1, t_m^3—$

Furthermore, let the equivalent sets for tokens t_i^1 , t_i^2 , t_k^1 , t_m^1 , t_m^2 , and t_m^3 be $\{y_{t_i^1}^1, y_{t_i^1}^2, y_{t_i^1}^3\}$, \emptyset , \emptyset , \emptyset , $\{y_{t_i^2}^1\}$, and \emptyset , respectively. Seven new patterns may be created from P by generating all permutations of the elements in the equivalent sets for tokens in P , Table 1. We generate all these permutations with the risk that some of the new patterns may not have any meaning.

The following algorithm sums up the creation of a pattern for a given sentence. A home grown parser (PARS) is used to generate a parse tree for the sentence automatically. PARS that employs an “incremental parsing technique [11] is imperfect and limited. Nevertheless, it is very helpful to our goal.

Algorithm Pattern

Given: A sentence, S , and PARS.

Objective: Creating a pattern for S .

Step 1- Create a parse tree for S using PARS.

Step 2- Build a crude pattern by keeping all the leaves of the grammar tree that are either NP (noun phrase) or ADJP (Adjective phrase)
Replace the non-selected leaves by dashes.

Step 3- Convert The dashes to gaps by replacing all the contiguous dashes by one large dash.

Step 4- Refine the “crude pattern” using a trained individual.

Step 5- End;

Example, the sentence of “The real content of this trunk box is the sum of eighteen million United States dollars which was his share from a secret sale of diamond” is borrowed from an actual spam. The grammatical components of the sentence are shown in a tree format in Figure 1.

secret sale—diamond” has two entries of “secret sale” and “diamond” with the same P-ID in this relation. The second relation, LTable(P-ID, HEAD), is a look-up table for the head of the patterns. The third relation, PType(P-ID, TYPE), carries type for each pattern. The fourth relation, PWeight(TYPE, WEIGHT), carries weight for each type and it has only four tuples.

3.3. Identification of Spam Indicators

The following algorithm identifies the syntax-based and semantic-based indicators for a given e-mail.

Algorithm Syntax-Indicator

Given: An e-mail, an English dictionary, weights for the five syntax-based indicators (w_{pm} , w_{pac} , w_{ac} , w_{ne} , and w_{ms}), Patterns-DB, and The five counters of $count_{pm}$, $count_{pac}$, $count_{ac}$, $count_{ne}$, and $count_{ms}$.

Objective: Identification of the syntax-based indicators and their quantifications.

- Step 1. Create a filtered e-mail;
- Step 2. Tokenize the filtered e-mail;
- Step 2. Repeat steps 2 and 3 for each token t_i ;
 If t_i is a PM indicator,
 Then Increment $count_{pm}$;
 If t_i is a PAC indicator,
 Then Increment $count_{pac}$;
- Step 3. Create a cleaned-email out of the filtered e-mail;
 If t_i is an AC indicator,
 Then Increment $count_{ac}$;
 Remove t_i from e-mail;
 If t_i is an NE indicator,
 Then Increment $count_{ne}$;
 Remove t_i from e-mail;
 If t_i is an MS indicator,
 Then Increment $count_{ms}$;
- Step 4. $W_{\bullet} = count_{\bullet} * w_{\bullet}$;
- Step 5. End;

Algorithm Semantic-Indicator

Given: An e-mail, weights for the four semantic-based indicators (w_v , w_p , w_h , and w_m), Patterns-DB that includes Patterns(P-ID, GROUP), LTable(P-ID, HEAD), PType(P-ID, TYPE), and PWeight(TYPE, WEIGHT). The four counters of $count_v$, $count_p$, $count_h$, and $count_m$.

Objective: Identification of the semantic-based indicators and their quantifications.

- Step 1. Create a cleaned e-mail;
- Step 2. Repeat steps 2 through 6 for each sentence of the e-mail

- Step 3. Tokenize the sentence;
- Step 4. Repeat steps 4 and 5 for each token, t_i ;
 Location = position t_i in the sentence (POS);
 $R1 = \text{Select P-ID from LTable}$
 where HEAD = t_i ;
- Step 5. If ($R1 \neq \emptyset$) Then
 Repeat for each R1.P-ID
 Select GROUP from Patterns
 where Patterns.P-ID = R1.P-ID;
 Repeat for each GROUP of the pattern
 If (GROUP is found in the sentence
 and its POS > location),
 Then location=POS;
 FLAG=True
 Else FLAG = False;
 If FLAG = True,
 Then
 Select PTypes.TYPE from R1, PTypes
 where PTypes.P-ID = R1.P-ID;
 Increment $count_{PTypes.TYPE}$;
- Step 6. $W_{\bullet} = count_{\bullet} * w_{\bullet}$;//The weight for the pattern //type is retrieved from the relation PWeight.
- Step 7. End;

The above two algorithms identify the syntax and semantic based indicators and they quantify a given e-mail by providing nine weight values (five weights and four weights for syntax and semantic based indicators respectively.)

3. Results

We have total of 300 spam and non-spam e-mails (150 of each). Each e-mail is quantified as a record of ten variables. The first nine variables are the following weights for the e-mail: W_{ac} , W_{pac} , W_{pm} , W_{ne} , W_{ms} , W_v , W_p , W_h , and W_m . The last variable carries the class of the e-mail which is shown by values 1 and 0. The value one means that the e-mail is spam and zero means otherwise. The dataset is normalized and the attribute W_{pm} is deleted because more than 95% of the values for this attribute are zero.

The 15% of the records (e-mails) that are chosen randomly from each class make a test set and the remaining records make a training set. This process is repeated three times to generate three pairs of training and test sets. The randomly generated test sets do not have any common records.

A neural network is built to be trained by a training set and to be used for classification of the records in the test set. The neural net includes an input layer with nine nodes, one hidden layer with ten nodes, and

an output layer with one node. The number of nodes in the hidden layer is determined using a trial and error approach. The percent of correct classification for the test sets along with the calculation of sensitivity and specificity measures are given in Table 2.

To provide a base for comparison of Semantix-Plus behavior, we use the G-mail (Google mailer), Yahoo, and AASU's (Armstrong Atlantic State University) spam detectors to determine the class of the same test sets. To do so, we simply send each e-mail of the test set separately as a new e-mail to three new e-mail addresses at Gmail.com, mail.yahoo.com, and mail.armstrong.edu to observe the behavior of the three e-mailers. Thirty senders send these messages to their destinations. To avoid introducing a bias in the process, the senders and receivers are not on the same mailer system. The average for the percentage of the correct classifications for the same test sets done by G-mail, Yahoo, and AASU mailers along with the calculation of sensitivity and specificity measures are also shown in Table 2. In addition, the false positive (F+) and false negative (F-) for each test set is displayed in Table 3.

4. Conclusion and future Research

The Semantix-Plus is not able to detect 5% (on average) of the spams whereas Google, AASU, and Yahoo mailers can not detect 20%, 33.4%, and 42.5% of the spasm, respectively.

The G-mail outperforms the Semantix-Plus by 1.7% (on average) when it comes to detect non-spam e-mails. However, Semantix-Plus outperforms the other two mailers. The outperforming of Syntax-Plus by Google mailer in this category is caused by the fact that Semantix-Plus works with a limited number of patterns. For the correct classification of spams and non-spams, the Semantix-Plus outperforms Google, AASU, and Yahoo mailers with a high margin of 13.4%, 34.2%, and 38.8%, respectively. We believe that detection power of the Semantix-Plus increases when the number of patterns is increased.

As future research a great deal of efforts is in progress to (1) automate the process of pattern creation and (2) test the Semantix-Plus for a high volume of e-mails.

5. Acknowledgement

We would like to thank Dr. Carol Jamison, at the department of language, literature, and philosophy at AASU for her valuable suggestions in building the patterns.

6. References

1. T. Fawcett, "In Vivo" spam filtering: A Challenge problem for KDD", SIGKDD explorations, Vol. 5, Issue 2, pp 140-148.

Table 2. The percentage of correct classifications of three different test sets using Semantix-Plus, Yahoo, G-mail, and AASU, spam detectors.

Test set and parameters of measurement		Semantix-Plus	Yahoo	G-mail	AASU
1	Correct Classific.	90	52.5	85	65
	Sensitivity	94.4	60	100	68.7
	Specificity	86.3	51.4	77	62.5
2	Correct Classific.	97.5	57.5	70	57.5
	Sensitivity	95.2	80	100	67
	Specificity	100	54.5	62.5	54.8
3	Correct Classific.	92.5	55	85	55
	Sensitivity	100	100	95	75
	Specificity	87	52.6	91	52.7
AVG	Correct Classific.	93.4	54.6	80	59.2
	Sensitivity	96.6	80	98.3	70.2
	Specificity	91.1	52.8	76.8	56.7

Table 3. The percentage of correct classifications of three different test sets using Semantix-Plus, Yahoo, G-mail, and AASU spam detectors.

	Semantix-Plus		Yahoo		G-mail		AASU	
	F ⁺ (%)	F ⁻ (%)	F ⁺ (%)	F ⁻ (%)	F ⁺ (%)	F ⁻ (%)	F ⁺ (%)	F ⁻ (%)
1	2.5	7.5	5	42.5	0	30	12.5	22.5
2	2.5	0	2.5	40	0	15	7.5	35
3	0	7.5	0	45	0	15	2.5	42.5
AVG.	1.7	5	2.5	42.5	0	20	7.5	33.4

2. J.C. Sipiør, B. T. Ward, and P. Gregory Bonner, Communications of the ACM, Vol. 47, No. 6, June 2004/ pp. 59-63.
3. R.D. Gopal, Z. Walter, and A. K. Tripathi, "Admediation: New horizons in effective e-mail advertising, Communications of ACM, Vol. 44, No. 12, Dec. 2001, pp. 91-96
4. D. Fallows, "Spam: How it is hurting email and degrading life on the Internet, Pew Internet & American association Life, Oct. 22, 2003, www.pewinternet.org/reports/toc.asp?Report=102.
5. A. Brandt, "Wireless industry moves to can the spam, PC World Vol. 20, No. 7, July 2002.
6. M. Perone, "An overview of spam blocking techniques. Technical report, Barracuda Networks, 2004.
7. P. A. Chirita, J. Diederich, and w. Nejdl, "MailRank: Using ranking for spam detection", CIKM'05, 2005, Bremen, Germany, pp.373-380.
8. P.O. Boykin and V. Roychowdhury, "Leveraging social networks to fight spam", IEEE Computer, 2005, Vol. 38, No. 4, pp. 61-68.
9. Fausett L. "Fundamentals of Neural Networks: Architectures, Algorithms, and Applications", Prentice Hall, Englewood Cliffs, 1994.
10. Lippmann, R. P. "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, 4:4-22, 1987.
11. Allen J., "Natural Language Understanding", 2nd Ed., Menlo Park, CA, Benjamin/Cummings.