International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/IJSRCSEIT

Apache Spam Assassin

Prof. Nagaraj Telkar, Miss. Annapurna Belamakar
Department of Computer Science and Engineering, SKSVMACET, Laxmeshwar, Karnataka, India

Article Info
Volume 8, Issue 3
Page Number : 378-385

Publication Issue :
May-June-2022

Article History
Accepted: 03 June 2022
Published: 15 June 2022

ABSTRACT

Spam is any form of annoying and unsought digital communication sent in bulk and may contain offensive content feasting viruses and cyber-attacks. The voluminous increase in spam has necessitated developing more reliable and vigorous artificial intelligence-based anti-spam filters. Besides text, an email sometimes contains multimedia content such as audio, video, and images. However, text-centric email spam filtering employing text classification techniques remains today's preferred choice. In this seminar, the text pre-processing techniques nullify the detection of malicious contents in an obscure communication framework. The use SpamAssassin corpus with and without text pre-processing and examined it using machine learning (ML) and deep learning (DL) algorithms to classify these as ham or spam emails. Results show the supremacy of DL algorithms over the standard ones in filtering spam. However, the effects are unsatisfactory for detecting encrypted communication for both forms of ML Algorithms.

Keywords: Text Preprocessing, Machine Learning Algorithms, Deep Learning, SpamAssassin.

I. INTRODUCTION

E-mail is an essential form of communication in our digital world. Unfortunately, it is often abused by people wishing to use it for advertising their products. Unsolicited bulk e-mail, better known to users as "spam" is a huge problem. One estimate suggests that American companies are losing $71 billion per year in productivity due to spam. One of the most widely used spam filters, Apache SpamAssassinTM1, is developed as an open-source project by people across the world. SpamAssassin works by searching through an e-mail message for certain patterns or characteristics of a message, specified as "rules". Each rule has an associated score, and the scores for each rule that matched on a message are summed. If this total score is greater than a threshold value, the message is marked as spam, otherwise, it is marked as non-spam (often referred to as "ham" in the anti-spam world). Choosing the proper scores for each rule is essential for the accuracy of the software. One of the biggest challenges for the Apache SpamAssassin project is to optimize the scores for each rule. The scores must be chosen so that spam messages have a total score for the message that is greater than or equal to the default threshold of 5.0, and non-spam messages have a smaller score. In order to optimize scores, volunteers keep collections of old spam and

ham e-mails they have received. Before a major release of SpamAssassin, these volunteers scan all these messages with SpamAssassin and keep logs of which rules hit on which message.

This e-mail data can then be fed into an algorithm to generate scores. The project currently uses a custom-made algorithm known as the Genetic Algorithm (GA) to optimize scores. It takes hours to run in order to generate scores. In the past, a much faster algorithm, known as the Perceptron, was used, but the implementation is currently broken and unable to produce good scores. This project focuses on the method used to generate the scores for SpamAssassin. While the current score generation method is good, there is lots of room for improvement. No spam filter is perfect; every filter will have false negatives (spam messages not caught by the filter) and false positives (legitimate messages marked as spam). If we improve the score generation method, we can reduce the number of misclassifications, leading to better performance.

SpamAssassin is a well-known and widely used open-source spam filter. It is made up of about nine hundred binary valued "tests", each of which is related to a characteristic of spam or ham emails like the presence of a given keyword in the emails text or malformed email headers. Each test is associated with a score: the score is zero if the related characteristic is not present, while it is non-zero if such characteristic is present. Scores associated with spam and ham features are respectively positive and negative. For a given email, the score of all tests is summed up. If their sum is greater than a predefined threshold the email is labelled as spam, otherwise it is labelled as ham. The default threshold is 5.0 (note that the scores are not normalised to any interval like [0, 1]). Among the tests, nine are associated to disjoint intervals of the continuous valued output of a text classifier ("naive bayes" classifier). The remaining tests are simple feature detectors which look at the emails' body and header, including DNS block-lists and collaborative filtering databases. Default values are provided for the score of each test and for the decision threshold. These values can be modified by the user, either automatically, using a tool named "mass check" and a training set of labelled emails, or manually. Note that the naive bayes classifier and the mass-check tool are the only components of SpamAssassin based on machine learning techniques. This structure makes it easy to update SpamAssassin by adding or removing tests, as spam email characteristics change.

The organization of this document is as follows. In Section 2 (**Methods and Material**), I'll give detail of any modifications to equipment or equipment constructed specifically for the study and, if pertinent, provide illustrations of the modifications. In Section 3 (**Result and Discussion**), present your research findings and your analysis of those findings. Discussed in Section 4(**Conclusion**) a conclusion is the last part of something, its end or result.

II. BACKGROUND AND THEORY

SPAM

In 1936, Hormel Foods came up with the recipe. The company initiated the contest for the product name by offering $100 as the winning prize. Kenneth Daigneau won the contest by combining two words together, "SP" from spiced and "AM" from ham to create the word SPAM (SPiced HAM). In 1975, Monty Python's Flying Circus created the comedy skit wherein Vikings sing, "Spam, spam, spam, spam . . ." in a restaurant that included SPAM in each menu item. Afterwards, Internet users of Multi-User Dungeons (MUDs), bulletin boards, chat rooms and Usenet message boards began using the term SPAM for annoying, repetitive and unwanted messages. (Liana Technologies, 2012a.) According to Dictionary.com, spam is to send unsolicited electronic mail or text messages simultaneously to a number of e-mail addresses or mobile phones (Dictionary.com, 2013). However, if a long-lost brother finds and sends to your email address and sends you a message, this

could hardly be called spam, even though it is unsolicited (Webopedia, 2013).

Mainly, Spam is used for advertising. Normally, various products and services are getting advertised through mass mailing. Furthermore, Spam is used for traffic grabbing and less common for sending viruses or malware. But all of them have the same goal of reaching the most recipients with minimum costs. Moreover, quite often senders do not care about the target audience; the number of recipients is the key thing for them. Current estimates say that somewhere between 65 percent and 80 percent of all e-mail on the Internet is spam (Gregory, P. and Simon, M. 2005, 20).

The harmfulness of spam

First, the harmfulness of spam is the usage of alien resources (computers, networks, work, and money), also, breaking the national law by creating inappropriate advertisements, interference with privacy, moral damage, blocking off needed emails, spreading of viruses, etcetera. Thus, considering the amount of damage by spam it can be confidentially said that spam have been merged from the problem of individuals,

companies and governmental organisations to the problem of the society as a whole. There are three main damages caused by spam. Firstly, the traffic of inbound emails can cause the network of recipients or companies to be flooded and to have a reduction in the needed working speed. Secondly, the loss of the working hours and productivity is another damage of spam. Without spam protection users receive more like 400 email messages each day, based on the statistic that 90% of global email traffic is spam. Then, each employee categorizes spam and not spam emails that takes around 10 to 20 minutes of working time a day. Thirdly, vulnerability holes in the information security. As emails are capable of handling attachments, many Word or Excel documents can be affected by viruses. Moreover, spam can be tied to the hacking attack for breaching a security system. If the attack is succeeded, the company will have a bad reflection on the brand as well as the sensitive data will be deleted, falsified or stolen.

There are some known spamming techniques, for example, paid calls, when with the product advertisement comes in the phone number, calling to that number will result in an answer from an answering machine and you will get a connection fee. Another example is the advertisement of money pyramids. In that case, you will be promised to get rich after sending small amount of money to a certain account. Information gathering is also an example of Spam in which you will be asked to complete a survey and send it. Then the information could be used for identity theft, fraud or any other falsification. Sending of Trojans is another way to get the information, containing passwords, phone numbers, etcetera) from an unsuspecting recipient. In addition to various technical solutions, the on-going analysis of the language structure, it's changes as well as the most used spam words must be considered.

Spam does not have language barriers. Although spam written in English is the most common, it comes in all languages including Chinese, Korean and other Asian languages. 50% of spam falls into the categories of adult content, health, IT, personal finance, education and training. Before describing the approach and techniques employed in this work, it is helpful to look at the type of spam comments that we will face. The following elucidates the groundwork supporting our assertion that state-of-the-art spam filtering algorithms founded on NLP are likely to fall prey to even rudimentary steganography algorithms.

1. Dataset

The need for spam and phishing email parting is quite noticeable in email datasets. For example, the Enron dataset contains legitimate ham emails. In contrast, the University of California, Irvine (UCI) machine learning repository contains a dataset for spam emails and the Nazario dataset stores phishing emails. However, **the 'SpamAssassin' dataset has ham, hard**

Ham (a bit trickier contents), and spam emails and hence remained a preferred choice as a dataset for the author. The details about the dataset appear in Table 1, which are downloaded from the SpamAssassin website.

E-Mail type	Description
Spam	500 spam messages
Easy ham	2500 non-spam messages
Hard ham	250 non-spam messages which are closer in many respects to typical spam
Easy ham	21400 recent non-spam messages
Spam 2	1397 recent spam messages
Total	6047 messages having around 31% spam ratio

2. Natural language text pre-processing techniques

The necessary stages in the mining of data from email messages have the following categorization in the text pre-processing framework:

i. Tokenization and Segmentation: It divides the given text into minor tokens, where words, punctuation marks, numbers, and others constitute tokens.

ii. Normalization: These are the steps needed for translating text from human language (English) to machine-readable format for further processing.

The process starts with:

- Changing all alphabets to lower or upper case
- Expanding abbreviations
- Excluding numbers or changing those to words
- Removing white spaces
- Removing punctuation, inflection marks, and other circumflexes
- Removing stop words, sparse terms, and particular words
- Text canonicalization

iii. Noise Removal: It is a task-specific section that can occur before or after the tokenization and normalization or in between. Examples include removing HTML Tags, CSS, and similar other traits. Dimensionality reduction creates a new set of attributes, whereas feature selection methods include and exclude details present in the data without alteration.

iv. Feature selection: Following the pre-processing stage is the feature selection. Unlike dimensionality reduction, feature selection effectively typifies email message fragments as a compressed feature vector. The technique comes in handy for large datasets.

3. Studies on the User Action Prediction:

Problem User action prediction aids users in processing daily emails by automatically determine the action that should be done on an email. If a user knows which email should be replied, 162 H.-N. Thanh et al. read or deleted, he will be able to take suitable actions in order to save time. There have been a few studies to tackle this problem. The problem was first introduced in 2005. Authors of evaluated the factors that affect user's responses to incoming emails in order to produce a method to predict user action. Their study was done through an online survey in which questions are divided into three parts. The first part is to collect users' work environment information and the characteristics of their jobs. The second part asks about users' email usage and habits. The last part gathers information on content characteristics, level of importance, characteristics of the sender and associated actions on emails. The study used a self-built dataset with 1100 email messages. 124 persons took part of the survey. 10 features were used to predict the importance of an email. This study found that there is correlation between the importance/probability to get replied of an email and user's relationship with the sender as well as the email's content. In [2], the authors proposed an email user action recommender system. The system is essentially a Bayesian multiclass classifier where each class represents a user action.

There are three user actions: “reply”, “read” and “delete”.

SpamAssassin, or more specifically, a SpamAssassin ruleset, can separate spam from ham. Therefore, it is a binary classifier. In this paper, we make SpamAssassin a multiclass classifier to predict user action (REPLY, READ, DELETE) on an email. Our proposed method can be easily configured to build a system which classifies emails into more than 3 classes. We apply three different approaches of building multiclass classifiers: OVA, OVO and DAG.

4. Preferred algorithms

- Naive Bayes:

Strong independence assumptions between the features exist for Naive Bayes. They have a simple design, easy to implement and train. This algorithm needs small training data to estimate the parameters required for classification purposes. This classifier shows aptness to real-life situations. The math behind the Naive Bayes algorithm is as follows:

$P(A \mid B) = P(A \cap B)/ P(B)$

$P(A \cap B) = P(A \mid B)\, P(B)$

Likewise, $P(B \mid A) = P(B \cap A)/ P(A)$

$P(B \cap A) = P(B \mid A)\, P(A)$

$P(A \mid B)\, P(B) = P(B \mid A)\, P(A)$

$P(A \mid B) = P(B \mid A)\, P(A)/ P(B)$

- Gaussian Naive Bayes: A variant of Naive Bayes (NB), Gaussian Naive follows Gaussian normal distribution and differs in provisioning support for continuous type data. The prospect of the features is assumed to be:
- LSTM Recurrent Neural Network (RNN): are devoid of short-term memory, as shown in Figure 1. They cannot synchronize with previous time steps, such as text processing, for extended sequences of information. They suffer from the waning gradient problem during backpropagation, where gradients are values that update the weights of a neural network. The vanishing gradient problem occurs when the gradient shrinks while propagating back through time and loses contribution in learning for minimal values.

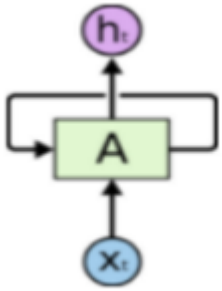

Figure 1. Illustration of Recurrent Neural Network

III. METHODOLOGY

Software development methodology refers to **structured processes involved when working on a project**. It is a blend of design philosophies and pragmatic realism that stretches back to the early days of computing. The goal is to provide a systematic approach to software development.

What SpamAssassin looks for in E-mail?

SpamAssassin “reads” emails and runs tests to look for any attributes and patterns associated with spam. Every time it “reads” an email message, hundreds of tests are performed. They look for things like:

- ✓ Spammy email content
- ✓ Presence on blocklists
- ✓ DKIM and SPF record configuration
- ✓ Suspicious links and attachments
- ✓ Spam related terms
- ✓ Disallowed scripts

The test also aims to verify the source of the message, and that the message hasn’t been altered in transit, by looking at the DKIM.

It also considers the historical engagement of recipients who received emails sent by you. If there is a history of recipients not opening your emails, but you keep sending to them anyway, this will have a negative effect on your SpamAssassin score.

3.1 SpamAssassin

SpamAssassin uses a rule-based detection method that compares different parts of email with many pre-defined rules. Each rule adds or removes points from an email's score. An email with a high enough score is spam. Most of SpamAssassin rules are basically Regular Expressions used to find textual structures which indicate that a message is spam. Figure 1 shows an example of Spam Assassin's rule.

Typical SpamAssassin body rule.

body MONEY_BACK /money back guarantee/i
describe MONEY_BACK Money back guarantee
score MONEY_BACK 2.910

There is a body rule named MONEY_BACK. This rule checks if the body of an email contains a string that matches the Regex "/money back guarantee/i" and adds a score of 2.910 to the total spam score of that email. The higher the total score, the more likely a message is spam. By default, all emails whose total score is equal to or higher than a threshold T = 5.0 are considered spam by SpamAssassin. The threshold T can be adjusted by user. Equation (1) shows how SpamAssassin calculates the total score for each message.

$$Score_R(m) = \sum_{i=1}^{k} Match(R_i, m) \times w_i \quad \text{-- (1)}$$

where:

a. ScoreR(m) returns the total score against ruleset R for the message m.
b. R is a set consisting of k rules (R1… Rk).
c. Match(Ri ,m) returns 1 if m contains a string that matches rule Ri , 0 otherwise.
d. w is a set of k scores (w1…wk) corresponding to k rules

SpamAssassin provides a built-in module to score its rules. The scoring module works as a single-objective optimization method. It sets the threshold to a fixed value, then optimizes the scores to decrease the error rate over a given training dataset. SpamAssassin provides a built-in module to score its rules. The scoring module works as a single-objective optimization method. It sets the threshold to a fixed value, then optimizes the scores to decrease the error rate over a given training dataset. SpamAssassin uses the Stochastic Gradient Descent algorithm to of training a single-layer neural network with a transfer function and a logic activation function. Each node of the neural network represents a rule of SpamAssassin. The input of each node represents whether or not the rule is activated by an email. The weight of each node is respected to the score of that rule. SpamAssassin uses a linear function to map the weights to the score space.

The score return by ScoreR(m) is compared to the threshold T to determine if m is spam using Eq. (2).

$$Spam_R(m) = \begin{cases} 1, & Score_R(m) \geq T \\ 0, & Score_R(m) < T \end{cases} \quad \text{--(2)}$$

3.2 Automatic Generation of SpamAssassin Rules

The method for automatic generation of SpamAssassin rules to detect spam in Vietnamese. That process can be summarized as follows:

Step 1 – Dataset preparation: According to data labels, separate the training set into two parts. The first part, called D1, contains spam messages and the order part – D2 – contains ham messages.

Step 2 – Extracting words: The Vietnamese tokenization tool tokenizer is used to extract words from the email subjects in D1 into the set WS1. Similarly, words from the content of all emails in D1 are extracted into the set WB1.

Step 3 – Selecting keywords: The most frequent words from WS1 and WB1 are kept and put into two new sets, WS2 and WB2.

$WS2 = \forall w \in WS1, freqWS1(w) > \alpha$
$WB2 = \forall w \in WB1, freqWB1(w) > \beta$

The function freqWS1 (w) returns the times which the word w appears in WS1. The two parameters, α and β, should be adjusted according to the size of the dataset. In our experiments (which are described later in Sect. 4), we use the value 2 and 6 for α and β respectively.

Step 4 – Building ruleset:

A ruleset R1 is build. Subject rules are generated from the keywords in WS2, and body rules are generated from keywords in WB2. Figure 2 shows the structure generated rules. In Fig. 2, is replaced by the actual keyword from the two keyword sets.

```
header    ReplySubj_i   Subject ~= /\b<word>\b/i
describe  ReplySubj_i   Subject contains "word"
score     ReplySubj_i   0.1
```

Step 5 – Rule selection:

SpamAssassin's MassChecks tool is executed to see how the rules in R1 are matched against emails in D1 and D2. Bad rules – rules with low hit rate or rules which match both spam and ham – are removed to create a new ruleset called R2.

Step 6 – Weight (score) optimization:

First, the MassCheck tool is executed again for the new ruleset (R2). In R2, each rule is initialized with a score value 0.1. A perceptron with a linear transfer function and a logic activation function is built. Its weights are mapped to rule scores. It is then trained using the Stochastic Gradient Descent method 164 H.-N. Thanh et al. to achieve highest spam recall and lowest ham error rate. When training completes, the ruleset R3 is created from R2's rules and trained scores.

3.3 Rule Based Algorithm

```
foreach line (lines in rendered message)
{
        if (line contains /pattern_1/)
{
        got_hit("RULE1"); last;
        }
}
foreach line (lines in rendered message)
{
        if (line contains /pattern_2/)
{
got_hit("RULE2"); last;
        }
 ... }
```

IV. RESULTS ANALYSIS

A score is given for each attribute SpamAssassin checks. These show the likelihood that the email is indeed spam. Individual attribute scores are added to give you your overall SpamAssassin score.

When mail server administrators or email providers are setting up SpamAssassin, they can pick their own SpamAssassin score threshold. The default score is 5, though the SpamAssassin team encourages people to set a score they feel comfortable with and lower it accordingly. Since administrators and email providers can put whatever value, they see fit, it's good to not settle for a score of 5 and call it a day. Some email services or administrators might set their threshold to 3, which means your email will be rejected and will not arrive in their inbox.

For this reason, it's important to aim for the lowest score possible. Aim for score below 4. Anything above 2 and you should be looking to fix whatever issues are causing tests to fail.

At first, seeing a negative value as a SpamAssassin test result might be perplexing.

Figure 2: Email Content Page

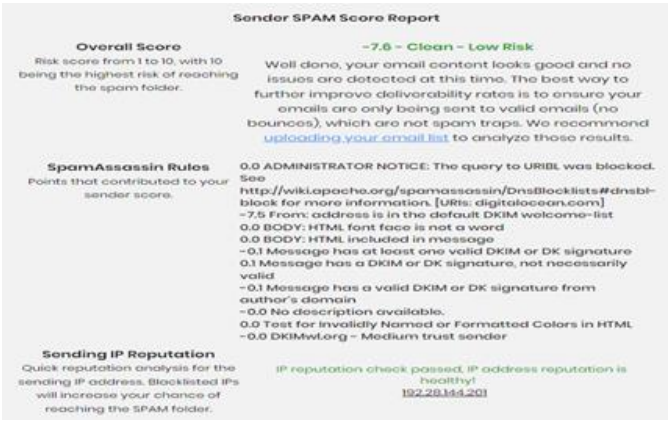

Figure 3: Spam Score Report

AREA TESTED	TEST DESCRIPTION	TEST NAME	loca
header	From: address is in the user's white-list	USER_IN_WHITELIST	-100
header	User is listed in 'all_spam_to'	USER_IN_ALL_SPAM_TO	-100
header	From: address is in the user's DKIM whitelist	USER_IN_DKIM_WHITELIST	-100
header	From: address is in the user's SPF whitelist	USER_IN_SPF_WHITELIST	-100
header	Subject: contains string in the user's white-list	SUBJECT_IN_WHITELIST	-100
header	User is listed in 'more_spam_to'	USER_IN_MORE_SPAM_TO	-20
header	From: address is in the default white-list	USER_IN_DEF_WHITELIST	-15
header	From: address is in the default DKIM white-list	USER_IN_DEF_DKIM_WL	-7.5
header	From: address is in the default SPF white-list	USER_IN_DEF_SPF_WL	-7.5
header	User is listed in 'whitelist_to'	USER_IN_WHITELIST_TO	-6
header	Contains valid Hashcash token (>25 bits)	HASHCASH_HIGH	-5
header	Contains valid Hashcash token (25 bits)	HASHCASH_25	-4
header	Contains valid Hashcash token (24 bits)	HASHCASH_24	-3
header	Contains valid Hashcash token (23 bits)	HASHCASH_23	-2
header	Passed through trusted hosts only via SMTP	ALL_TRUSTED	-1
header	Contains valid Hashcash token (22 bits)	HASHCASH_22	-1
header	Contains valid Hashcash token (21 bits)	HASHCASH_21	-0.7
header	Contains valid Hashcash token (20 bits)	HASHCASH_20	-0.5
full	Message has at least one valid DKIM or DK signature	DKIM_VALID	-0.1
full	Message has a valid DKIM or DK signature from author's domain	DKIM_VALID_AU	-0.1
header	Informational: message was not relayed via SMTP	NO_RELAYS	-0
header	SPF: sender matches SPF record	SPF_PASS	-0
header	SPF: HELO matches SPF record	SPF_HELO_PASS	-0

Figure 4: Test performed on different parts of Email Body

V. CONCLUSION AND FUTURE SCOPE

The dynamism that spam structuring offers and the counter strategy of spammers to combat have made spam filtering an active research area, leaving a broader scope for exploration and evolution of new spam filters encompassing the present-day cyber security requirements. Moreover, the companies think that by using mass mailing and sending advertisements of their brilliant products and services will result in a huge revenue and attraction. Obviously, there will be attraction, but of a bit different than expected as nobody likes to receive spam. In case of making spam economically viable, the problem might disappear at all.

The spam filter teaching almost never ends, because spammers create something new all the time. In addition, the spam trends do change over time, for example, the mortgage topic was of an interest in spammers before the financial crisis in 2007-2008. Now mortgage topics are not risen very often. Besides the teaching and adapting the spam filter towards any needs, the analysis of words, word combinations and sentences should be considered. The deep knowledge of the language can help in developing the rules further. Moreover, the number of rules for SpamAssassin, that is considered the best and most widely accepted solution, could be indefinite. However, it should not be forgotten that the higher verification level could result in a higher chance of false positives.

Cite this Article

Prof. Nagaraj Telkar, Miss. Annapurna Belamakar, "Apache Spam Assassin", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8 Issue 3, pp. 378-385, May-June 2022.

Journal URL : https://ijsrcseit.com/CSEIT12283108