A NEURAL NETWORK METHOD FOR SPAMASSASIN RULES GENERATION

Nguyễn Thanh Hà[*], Đặng Đình Quân[+], Trần Quang Anh[#]
[*] Sở Thông tin và Truyền thông Thành phố Hà Nội
[+] Khoa Công nghệ thông tin – Trường Đại học Hà Nội
[#] Học viện Công nghệ Bưu chính Viễn thông

Abstract: SpamAssassin has been widely used for spam filtering on e-mail servers for its recognized real-time performance and its ease of customization. Unfortunately, SpamAssassin does not come with default support for languages other than English. Although its default rule set for English spam detection is frequently updated, users usually have to train their own set of rules to match the signature of their particular e-mail traffic. There have been many proposed methods for the generation of SpamAssassin rules in many languages including but not limited to English [6], [9], [16], Chinese [11], Thai [17] and Vietnamese [12]. The general drawback of these methods is the use of hand-engineered feature selection, which is a time-consuming process because it involves a lot of data observation and analysis. In this paper, we propose a multilayer neural network model for generating SpamAssassin rules which selects good features and optimize rule weights at the same time. The weighted rule set obtained from training this neural network can be applied directly in SpamAssassin. The experiments showed that our network is fast to train and the resulted rule set has comparable detection rates to previous rule generation methods.

Keywords: neural network, rules generation, spam filtering, SpamAssassin.

I. INTRODUCTION

Roughly five decades since its first implementation for ARPANET in 1971, electronic mail (e-mail) has involved into the most important form of online communication. Nowadays, its applications include but not limited to online identity verification, personal and business communications. According to Radicati's report [20], in 2018, there were 281.1 million e-mails being sent daily and the number of e-mail users reached 3.823 billion. Spam (unsolicited bulk e-mail) accounts for 55% of all e-mail messages as reported by Symantec in 2019 [21]. This volume of spam represents a serious problem which is not only annoying but also costly to e-mail users.

The two most popular approaches to spam filtering are rule-based (or signature-based) filtering and machine learning. Although spam filters based on machine learning proved superior efficiency, better detection rates often come with the cost of more computational power. Meanwhile, rule-based filters have been widely used for their low complexity and non-intrusive nature [18]. Among rule-based techniques, SpamAssassin[1] remains the most utilized one on the e-mail server side. Because of its fast detection engine and sophisticated rule formats, SpamAssassin is able to capture a wide range of e-mail features in real-time applications of spam filtering. Since SpamAssassin's capability depends on its rule set, researchers have proposed hybrid methods which make use of machine learning elements to generate rules from data [6], [11], [16].

Rule generation techniques for SpamAssassin follow a similar approach to traditional machine learning methods which consists of two major steps: feature selection/representation and model optimization. Once a presumably good set of features are chosen and vectorized, the model is trained only on that particular feature set. It is agreed [5] that the effectiveness of learning-based methods for spam filtering depends greatly on the feature selection phase. In other words, these rule generation techniques rely heavily on good rule (feature) selection to be effective. Unfortunately, this step is usually done separately and has no connection to the later step of training the rule set on data. The performance of the trained rule set is restricted by the quality of the feature set, which may not be the most effective one. Furthermore, the number of features also affects the filter's performance. Generally, using more features results in better evaluation results in exchange for longer execution time. On the other hand, a spam filter tends to achieve better generalization (cross-corpora performance) with less features [18].

In recent years, neural networks have become easier to train thanks to new optimization methods and new activation functions. Neural networks are generally trained with a gradient-based method such as stochastic gradient descent (SGD) which relies on the calculation of partial derivatives. With the introduction of the back-propagation algorithm [1], it became possible to effectively optimize the weights of connections associated to hidden layers in multi-layered neural networks with linear transfer functions and non-linear activation functions. The detection mechanism

Correspondence: Nguyễn Thanh Hà
Email: thanhha140589@gmail.com
Manuscript communication: received: 10/05/2020, revised: 11/25/2020, accepted: 12/12/2020.

of SpamAssassin is based on weighted keyword rules, which is similar to the perceptron model (a single-layer neural network). What its current rule optimization tool does is actually fitting a perceptron model on e-mail data. The model is built from a SpamAssassin rule set where each node acts as a rule in the set. In other words, each node in the perceptron model carries the rule's weight as its own weight.

In this paper, we propose a novel method that makes use of a multilayer neural network model for SpamAssassin rules generation. In this method, individual features are weighted and good features can be empirically selected. To realize these goals, we apply a customized training process on a neural network in which the former layers play the feature selection role and the last layer mimics the detection mechanism of SpamAssassin.

The rest of this paper is organized as follows:

- Section II reviews published works on rules generation techniques for SpamAssassin.
- Section III discusses the detailed steps of the proposed method.
- Section IV describes our experiments, the dataset and experiment results.
- Section V draws a conclusion of this research's outcome and discusses research direction.

II. RELATED WORKS

SpamAssassin is a popular open-source spam filter which makes use of multiple mechanisms for detecting spam messages. One of its detection mechanisms is based on weighted regular expression rules. These rules match against the header or body of e-mail. When an e-mail is being processed, a certain number of rules in the rule set are triggered by the content of that e-mail. The weights of those triggered rules are summed up as a single score which is the spam score of the e-mail message. If the spam score exceeds a pre-defined threshold value T, the message is then marked as spam. SpamAssassin allows the creation of customized rules and provides its users with a rule learning tool. This tool uses the SGD algorithm to train a *perceptron* model on labeled e-mail training data. The reason for this choice is that SpamAssassin's detection mechanism is similar to a perceptron network where node weights represent rule scores and node activation is equivalent to rule match. One can either set the value of T before learning SpamAssassin rules so as to let the learning algorithm adjust rule scores to suit the threshold T, or generate SpamAssassin rules first and later set the value of T to suit the threshold used by the learning algorithm.

Many methods have been proposed to improve SpamAssassin's spam detection using data. In [6], different spam filtering techniques dated until 2003 were integrated into SpamAssassin and compared. Different feature detectors (e.g. SpamAssassin, Information Gain, clustering) and different machine learning algorithms (e.g. Naïve Bayes and variants, Perceptron by gradient descent, ID3) were used to generate SpamAssassin rules. Experiments were conducted on several datasets: author's e-mails (15,000 e-mails), X Window System developer's Xpert mailing list + Annexia spam archive (15,000 e-mails, 50% spam, 50% ham), Lingspam, SpamAssassin. The paper reported best results from the SpamAssassin combined with clustering feature detector. However, the authors also stated that more tuning work and better corpus were needed to reproduce other papers' results more accurately.

In [9], the author described his method to adjust the scores in a rule set containing all default SpamAssassin keyword rules and a number of Bayes rules. These new rules, which are activated when the Bayesian probability of an e-mail falls within a specific range, were added to the default rule set. For example, "*BAYES_00 matches when bayes spam probability is between 0% and 5% etc*" [9]. In order to obtain the best detection rate, a generic algorithm was used to find the scores for these pseudo-rules and other rules in the set. Rule score training was based on a self-built dataset of 1,176 hams and 1,611 spams. This method was evaluated and compared with 4 other spam detection methods on a testing dataset (also collected by the author) of 109 hams and 1,011 spams. These compared methods are Multi-Response Linear Regression (MLR), Logistic Regression [2], SVM trained by the SMO algorithm [15] and a variation of the C4.5 decision tree algorithm called J48 [3]. Results showed that the proposed method performed significantly better than SMO, which has the most stable performance across different testing scenarios among the 4 compared methods, in terms of *ham error rates*. This method also achieved the highest Total Cost Ratio (TCR) in all experimented methods in [9]. TCR is a measure of how costly the method is compared to the manual remove of spam messages. The higher the value of TCR, the better.

The authors of [8] argued that the rule-based nature of SpamAssassin is not suitable for spam detection since spam e-mails are always changing. In order to verify this argument, the authors compared the default SpamAssassin rule set against a CBDF filter (a statistical method proposed by Kilgarriff et. al. in [4]). With the advantage of fitting to the training dataset, it is not surprising to see a significantly higher performance of CBDF compared to SpamAssassin. In addition, there was also the fact that personal e-mails were used for the experiments. SpamAssassin's rule set was manually engineered and rule scores optimized on a corpus collected by the SpamAssassin Project. The bundled rule set is intended for the use of general English spam detection. It is not supposed to perform well on a personalized context. The 3,834 personal messages (of which 205 are spam e-mails) that were used in [8] are not representative enough to make the experiments convincing. That being said, it can be implied from these experiments that the default rule set of SpamAssassin's is not suitable in personalized settings.

Another effort to improve SpamAssassin's performance was reported in [7]. The authors proposed the use of word stemming – a widely used preprocessing technique in information retrieval – as a way to combat spammers' attempts to fool spam filters by using different word forms that are visually similar to the original word.

Examples of such words are "V*agr@", "V.i-a.g*r.a", etc. The stemming algorithm maps different representations of the same word to a unique hash value. These hashes (also called *stems*) are then used in the operations of rule-based or statistical filters. It means that different forms of the same word are treated as appearances of a single word in a document. As a result, spammer's attempts to modify a message will only result in the same one. The experiment in [10] indicates that the application of the technique has greater effects in improving filtering performance on more recent messages (collected in 2004) than on older ones (collected in 2003).

A framework for generating statistical SpamAssassin rules for Chinese was presented in [11], which employed different feature detection methods. In this method, only spam-related features are utilized for spam detection. The authors of [11] showed the effects of different hyper parameters such as the number of rules and the average pattern size. A previously introduced word segmentation method was used in [11] to control the average size of tokenized patterns. The authors used an SGD method [10] for training SpamAssassin rule scores which are treated as neuron weights in a perceptron network. From experimenting on a large self-built corpus of 194,088 spam and 305,140 ham, the author reported best performance for 500 rules with an average pattern size of 3 characters (6 bytes) and Conditional Probabilities as feature detector.

In 2009, the application of another technique to improve SpamAssassin was proposed [16]. The authors combined active learning (AL) with semi-supervised learning (SSL) in order to not only increase SpamAssassin's detection rates but also greatly reduce the work needed to label training data – making the method more practical to the general users. This method is applicable when there is a large dataset in which only a small portion are labelled. Semi-supervised learning has been used to automatically assign labels to the rest of a dataset provided that a part of it was manually labelled. Generally, a classifier is trained with the labelled data before being used to label a certain number of unlabeled ones. Those newly labelled samples with high confidence are then added to the training set to re-train the classifier. The authors of [16] believe that the samples which return high confidence actually contain very little new knowledge because they are similar to the labelled ones from the training data. Instead, the ones which the classifier is uncertain about have a higher chance of holding beneficial information. Based on this assumption, [16] proposed to leave the labeling of those suspicious unlabeled messages to e-mail users (active learning). However, since the users only agreed to manually label a limited number of messages, clustering was employed so that only the centroid needs to be manually labelled and the label propagates the entire cluster. It is necessary to note that the propagation of label only applies to '*pure*' clusters – those whose messages receive the same label from the classifier. At this point, a number of newly labelled messages are added to the training set and the classifier is re-trained and the process can be repeated. Experiments on the TREC07p dataset, which contains 50,199 spams and 25,220 hams, shows that the method performs significantly better than the built-in auto-learning (SSL) feature of SpamAssassin (the experimented version is 3.2.5). Different setups are also compared to indicate the effects of the number of queries to the user, the number of clusters in the clustering step and the rate of label propagation. Increasing the number of user queries results in better true positive rates and lower false positive rates while changing the number of clusters does not modify the performance significantly. Additionally, higher rates of propagation often reduce performance rather than improve it.

The authors of [17] aimed to modify the statistical SpamAssassin rules approach in [11] for the Thai language. A hybrid word segmentation method for Thai called CUWS was used for input tokenization. The two feature detectors that has the best performance for Chinese – Conditional Probabilities and Bayes' Theorem – were adapted from [11]. The dataset used for evaluation of this model contains only 1,000 spams and 1,000 hams, all of them are in Thai language and are manually selected. The paper concluded that Thai rules increased SpamAssassin's overall detection confidence (with higher, more distinguished scores between spam and ham). It also reported the performance of the generated rule set where spam recalls are from 76.8% to 86.4% and ham errors are from 0% to 5% across 10-fold cross-validation attempts. Whether the performance could be increased by increasing the size of training data was not reported.

Another method to create SpamAssassin rules that targeted the Vietnamese language was reported in [19]. Features are extracted from the subject and body of both spam and ham e-mails in order to reduce the rate of false positives (ham misclassified as spam) which are more severe than false negatives. Moreover, a hybrid evolutionary algorithm (Hybrid Particle Swarm Optimization with Wavelet Mutation [14]) was used to optimize rule scores for its ability to better avoid overfitting than the previously used SGD algorithm. Experiment showed that the portion of ham rules in a rule set should be between 25% and 50% for best performance. While [19] found that the combination of spam and ham features worked best for Vietnamese, [11] and [17] found that only spam-liked patterns achieved higher performance in their languages respectively.

III. GENERATING SPAMASSASSIN RULES BASED ON NEURAL NETWORK

The reviewed methods above tried to improve SpamAssassin by focusing on different aspects in the spam detection process, namely the pre-processing of e-mail content [7], feature selection [11], [19], employing semi-supervised learning on e-mail data [16] and introduction of new rules and assigning rule scores [9]. In this paper, the authors aim to improve SpamAssassin by proposing another method for extraction of useful rules from e-mail data and optimization of those rules' scores. The method is based on training a neural network using a gradient-based algorithm. However, the actual goal is not the neural network itself, but rather a particular selection of weights from it.

A. Data preprocessing & representation

From the training set consisting of spam and ham, we use vnTokenizer – a Vietnamese word segmentation tool [13] – to separate the words from the messages' body and subject. Then, we create a set of distinct words (vocabulary) called $\mathbf{V_s}$ from the subjects. We call the similar set from the message bodies $\mathbf{V_b}$. In the proposed method, removing stop words are not needed because feature selection is done during neural network training and unimportant words are excluded during the process.

Each e-mail message is treated as a bag of words and represented by a one-hot encoding vector to simulate SpamAssassin's detection mechanism. Each element of this vector is a word feature, with value 1 meaning the word is present in the e-mail message and value 0 meaning the opposite. In a one-hot encoding scheme for text, the frequency of a word is not recorded, thus the value of a word feature will be 1 even if the word appears multiple times. The fact that one feature is needed for every word in the dataset (a.k.a. for every word in the vocabulary) makes the size of the input vector equal to the size of the vocabulary. In our method, subject and body features are distinguished, so the encoded vector $\boldsymbol{x}$ of an e-mail message contains two separate segments for subject words and body words. Therefore, its length is: $|\boldsymbol{x}| = |\mathbf{V_s}| + |\mathbf{V_b}|$.

B. The neural network model

The neural network that we use to learn SpamAssassin rules from a dataset consists of two main components. The first component is called the *feature selector* and the other one is called the *predictor*. The network and the training algorithm are designed so that selecting good features and learning the correct weights for those features are done in one process.

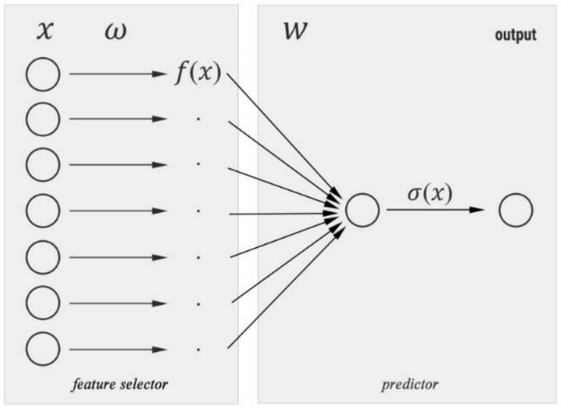

Fig. 1. The neural network structure with feature selector and predictor parts.

The *feature selector* part consists of one layer of neurons which are activated by function f defined in (1). An input e-mail message is fed into the feature selector layer in the form of a binary vector x which was described in a previous section. The input vector x and the weight set ω have the same size, which means each element in the input vector can be associated with one weight from ω. The role of ω is to hold the importance of each word, which in turn decides whether the word is selected as a feature. The approach is to first exclude all features and gradually activate significant features – the ones whose weights increase to a certain threshold after a certain amount of training. To achieve this effect, we introduce a global hyper parameter ε and a weight ω_i associated to each element x_i of the input vector. At each neuron of the *feature selector* part, the product of the input x_i and $f(\omega_i)$ is taken. When the output of f is 0, the feature represented by x_i is excluded from the forward pass but still included in the backward pass of the training process.

$$f(x) = \begin{cases} 1, & x > \varepsilon \\ 0, & x \le \varepsilon \end{cases} \tag{1}$$

In other words, it will have no effect on the output of the network but its weight ω_i will still be updated by the training algorithm and it still has a chance to be selected later. The value of ε should control the number of rules after training since it directly affects rule selection.

The remaining part of the network, the *predictor* part, is a perceptron with a sigmoid activation function, without bias. This predictor layer is also the last layer of the network. It takes the output of the previous *feature selector* layer as its input and outputs a scalar value. Let the output of the *feature selector* layer be vector h and the output of this predictor layer be a scalar k. The output of the network is calculated using the formula (2).

This predictor part simulates the default detection mechanism of SpamAssassin where the weights in the set w act as rule scores. These weights are initialized as random non-negative numbers and will stay non-negative throughout the training process.

$$k = \sigma\left(\sum_{i=1}^{|h|} h_i \cdot w_i\right) \tag{2}$$

Being output by the sigmoid function, k is a real number within the range (0, 1). The prediction result can be obtained by mapping k into into a discrete value of either 0 (*ham*) or 1 (*spam*) respectively. The mapping function depends on the specific problem where the network is applied. In general, we define a threshold value T that divides the range (0, 1). If k is greater or equal to T, a positive prediction is concluded and vice versa. T should be the middle value between the lowest and highest bounds of k (which are not always 0 and 1, see section III. C. for explanation).

C. Training the neural network

We train our neural network using the gradient descent method with backpropagation. The first step is to generate non-negative initial weights for two weight sets ω and w. It is done using random numbers in a folded normal distribution (taking the absolute value of Gaussian random numbers). Each initial weight is normalized by dividing it with its layer's size. The gradient descent method can be summarized as follows. Each sample in the training set is labeled with a target value (desired outcome). For each sample, calculate the output using current weights and get the different from output and target as the output's error. Calculate the partial derivative of the error with respect to each weight – which is the gradient of the weight. With gradients, update the weights in a manner which reduces the error. A *learning_rate* value may be used to control the speed at which the weights change. Each loop through all the samples in the training set is called an epoch. This training process is repeated for many epochs until a desirable total averaged error (or any chosen evaluation measure) is reached.

We have made some changes to the normal gradient descent training procedure to suit our network. Firstly, each weight ω_i in the set ω is updated as long as x_i is 1, regardless of whether the corresponding feature is selected by function f or not. This can be done by assuming that function f always returns 1 when calculating the partial derivative of ω_i. In opposite, the output of function f will decide if a weight w_i in the set w is updated or not. In other words, all weights in ω which is associated with input x_i value of 1 are updated whereas only the weights in w which subjects to $f(\omega_i) = 1$ (a.k.a. selected features) may be updated. Secondly, since sigmoid activation is used and rule weights are non-negative, the target value y_0 for ham samples should be 0.5 instead of 0. This is because the weighted sum of activated features cannot be lower than 0 and the sigmoid function outputs 0.5 for input 0. If the target y_0 is set to 0 for ham samples, the output error could not be reduced pass 0.5. This issue may lead to unnecessary reduction of rule weights when a ham sample is fed to the network during training.

*Table I. **D_0** dataset statistics*

User	No. of messages		Ham	Spam
1	All	3,854	2,998	856
	EN	1,645	1,031	614
	VI	2,209	1,967	242
2	All	4,144	1,858	2,286
	EN	622	25	597
	VI	3,522	1,833	1,689
3	All	5,478	2,191	3,287
	EN	3,845	1,301	2,544
	VI	1,633	890	743
Total		13,476	7,047	6,429

D. Generating a weighted rule set

The *predictor* part of the network structure is the representation of SpamAssassin's rule-based detection. Each neuron of the predictor layer has a weight can be associated with a word. The neurons which are selected by the activation function f of the previous layer are equivalent to SpamAssassin rules. A SpamAssassin rule set can then be generated by extracting information from the neural network model. In our experiments, we use SpamAssassin to test the resulting rule sets.

IV. **EXPERIMENTS**

A. Datasets

Both the proposed method and the method in [19] utilized a word segmentation method [13] which do not work well with content in languages other than Vietnamese. If the target rule set is expected to handle emails in multiple languages, then it is required to have a method to reliably detect content language – which is not within the established scope of this research. English, however, is the language which can be tokenized by splitting by white-spaces and punctuations. Therefore, it is feasible to also perform tests on a labeled English dataset. Since a public spam e-mail corpus in Vietnamese cannot be found, we have collected a Vietnamese e-mail dataset to experiment with the proposed method. The raw dataset, hereafter referred as **D_0**, consists of 17,869 e-mails from 3 users who regularly use e-mail for work. The messages in **D_0** are written in English and Vietnamese. After removing e-mails with empty body or e-mails whose content is mainly composed of images, there are 13,476 e-mails left.

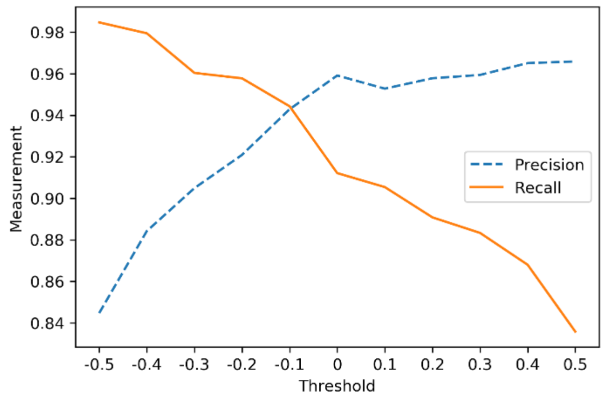

Fig. 2. recall and precision values for different threshold values for the method in [19]

E-mail owners are asked to label their e-mails. For each message, they have to complete two labels: language and spam. The *language* label takes two values: "*en*" and "*vi*". The *spam* label indicates that a message is spam (true) or ham (false). The labelers are asked to label their e-mails with this rule: if both the subject and body of the message has no valuable information, mark it *spam*, otherwise, mark it *ham*.

In our dataset, we also extracted three more features which are: the number of attachments, the number of hyperlinks and the number of <img> tags. We believe that these features can be useful in further research. **Table 1** summarizes the statistics of our **D_0** dataset.

The following experiments only utilize Vietnamese e-mails and textual features which are subject and body in the dataset. We extracted only Vietnamese e-mails from dataset **D_0**. As the result, we obtained a set of 7,364 messages total, of which there are 4,690 ham and 2,674 spam. We call this the **D_1** dataset in our experiments.

*Table II. Cross-validated F_1 score and precision measures of two methods on dataset **D_1***

Attempt #	Method in [19]		Proposed method	
	Precision	F_1	Precision	F_1
1	0.9628495	0.9513718	0.9674080	0.9553200
2	0.9563476	0.9236125	0.9650735	0.9733037
3	0.9592226	0.9501699	0.9630713	0.9739323
4	0.9505882	0.9280245	0.9380793	0.9420964

5	0.9656238	0.9500285	0.9750567	0.9699248
6	0.9551777	0.9395667	0.9252269	0.9390311
7	0.9637827	0.9284745	0.9755455	0.9726182
8	0.9616317	0.9246602	0.9667049	0.9555514
9	0.9510974	0.9372036	0.9685275	0.9676212
10	0.9645881	0.9506829	0.9420821	0.9514994
Average	**0.9590909**	**0.9383795**	**0.9586776**	**0.9600898**

In addition, we also use the TREC07 public spam corpus for evaluating the performance of the proposed method for English e-mails. With this dataset, it is also possible to compare our results with other English-based SpamAssassin rules generation methods. The TREC 2007 corpus [12] includes e-mail messages collected from an e-mail server in a time period of roughly 1 month. It is carefully analyzed and labeled by spam specialists at TREC and it has been widely used for spam filter benchmarking. The corpus contains 75,419 e-mail messages, 50,199 of which are marked as spam and the remaining 25,220 are legitimate messages. Both the message content and headers are provided. More datasets can be obtained from [12]. In our experiments, the TREC07 dataset is hereafter called the $\mathbf{D_2}$ set.

B. *k-fold cross validation*

In our experiments, k-fold cross validation (k = 10) is applied to increase confidence on the results. A dataset is first shuffled before it is divided into 10 equal parts which have roughly the same spam-to-ham ratio. The training and testing were repeated 10 times where each part of the dataset is selected as the test set while the rest are combined as the train set. This ensures that every part of the dataset contributes to both the training and testing results of a particular method. The results reported in this paper are the average values obtained from performing k-fold cross validation.

C. *Experiment 1*

1) *Summary of previous method*

Among previous studies about generating SpamAssassin rules, we have found one study [19] that targeted the Vietnamese language. The method in [19] can be summarized as follows:

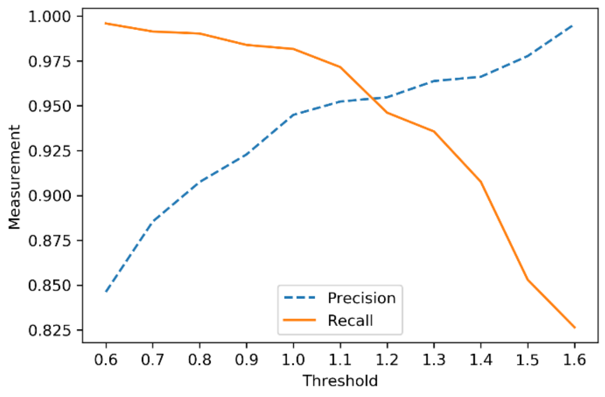

Fig. 3. recall and precision values for different threshold values for the proposed method.

Firstly, words are extracted from e-mail subject and body using the method in [13]. Then, a number of highest-quality words from spam e-mails and from ham e-mails are selected based on Bayes' probability theorem. Each selected word is considered a feature as well as a rule, and a SpamAssassin rule set can be generated from the set of these words/features/rules. This feature selection step to get the set of keyword rules is done separately from the next step of optimizing rule scores. Without any connection from the set of selected rules to the prediction result of the rule set, the quality of selected rules could not be verified. Thus, this feature selection step is a blind process. Next, an evolutionary optimization algorithm called HPSOWM was used to optimize rule scores on a labeled training dataset of Vietnamese e-mail messages.

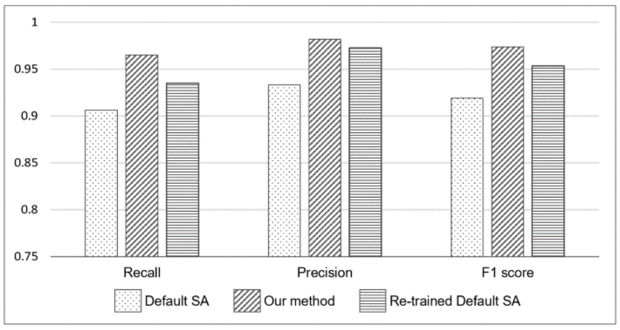

Fig. 4. Average recall, precision and F_1 values of three different method configurations on the English dataset $\mathbf{D_2}$.

In [19], various numbers of selected words as well as various ratios between selected spam words and ham words were tested.

2) *Experiment setup*

In this experiment, we reproduced the result of [19] and compared the performance of our proposed method on dataset $\mathbf{D_1}$. Two methods were used to independently build two separate SpamAssassin rule sets. For our previous method [19], a SpamAssassin detection threshold value $T = 0.0$ was used for both training and testing since the method also makes use of ham (negatively weighted) rules and $\sigma(0) = 0.5$ (the center point between the target values 0.0 and 1.0). Moreover, because the target number of rules has to be set manually, the reported best values of 500 spam rules and 500 ham rules were selected for the experiment. For our proposed method, a threshold value $T = 1.1$ was used since 0.75 is the center point between the target values (0.5 and 1.0) and the network output $\sigma(1.1) \approx 0.75$. We determined the threshold value in advance in order to let the training algorithm fit the rule weights according to the threshold. By doing so, the selected threshold value should be the optimized one. The experiment is run in a k-fold cross validation scheme, explained earlier in this paper.

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{3}$$

To reliably measure the performance of the generated rule sets, we used F_1 score (5) since it is a balanced combination of the two popular measures: recall and precision. F_1 score does not suffer from the problem in classification where unreliable results come from the fact that samples are not distributed evenly between classes.

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} \quad (4)$$

In the spam detection problem, the number of false alarms often receives the most attention because it is costly to discard a legitimate e-mail message. For this reason, we also use precision (3) to report the result of this experiment. precision is calculated from the number of *true positives* (correct prediction of spam, tp) and *false positives* (ham misclassified as spam, fp) while recall (4) is calculated from *true positives* and *false negatives* (spam misclassified as ham, fn). Spam messages are often stored in a separate folder in the user's inbox, and are usually deleted automatically after a while. By the time of this writing, Gmail (*mail.google.com*) automatically deletes spam messages which are older than 30 days. It is against a user's benefits when a ham message is detected as spam and being deleted without the user knowing. A low precision measure indicates that this situation happens more frequently.

$$\text{F}_1 = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \quad (5)$$

Since a statistical classifier is not guaranteed to achieve 100% accuracy (the amount of correct predictions over all predictions), recall is often sacrificed to gain better precision. It can be done by reducing the classifier's *sensitivity*, making it harder for the classifier to generative positive predictions. Reducing sensitivity lowers recall while raising precision and vice versa. In SpamAssassin, the filter's sensitivity is governed by the previously mentioned threshold value T. Sensitivity and threshold are opposite terms: the higher the threshold, the lower the sensitivity.

In practice, e-mail users are often concerned with the trade-off between recall and precision. High recall frees the user's inbox of spam but also leads to more legitimate messages being moved to the junk mail folder. Meanwhile, high precision means less ham are mistakenly marked as spam but also means less spam are detected, leaving more spam messages in the user's inbox. In this experiment, we also report recall and precision at different threshold values to demonstrate the concerned trade-off (see Fig. 2 and Fig. 3).

3) Result

It can be observed from **Table 2** that the new method achieved comparable precision as the one reported in [19]. However, the method in [19] has a significantly lower recall rating, as can be inferred from a lower F_1 score. It can be drawn from these figures that the proposed method can filter much more spam messages while having a similar capacity to prevent legitimate messages from being sent to the junk mail box.

*Table III. Results of three methods on dataset **D₂***

#	Default SA		Re-trained SA		Proposed	
	Prec.	F_1	Prec.	F_1	Prec.	F_1
1	0.91433	0.90893	0.96914	0.94703	0.98190	0.96610
2	0.91548	0.92585	0.97302	0.95304	0.97952	0.97570
3	0.92336	0.90660	0.97623	0.94955	0.98637	0.97605
4	0.93057	0.90933	0.97410	0.96272	0.97916	0.97159
5	0.93701	0.92164	0.97390	0.95862	0.98439	0.97589
6	0.94760	0.92726	0.97558	0.95281	0.98728	0.98044
7	0.92081	0.90485	0.97009	0.95318	0.98466	0.97804
8	0.94088	0.93806	0.97098	0.95175	0.97866	0.96892
9	0.96800	0.92173	0.97469	0.95488	0.97580	0.96973
10	0.93139	0.92673	0.96946	0.95236	0.98055	0.97238
Avg.	**0.93294**	**0.91910**	**0.97272**	**0.95359**	**0.98183**	**0.97348**

k-fold cross-validated precision (Prec.) and F_1 score of our proposed method, default SpamAssassin rule set and re-trained default SpamAssassin rule set on English dataset **D₂**

D. Experiment 2

We carried out this experiment to see how effective this new method is for English spam detection compared to the original method that generated SpamAssassin's default rule sets. For this goal, the dataset **D₂**, which is a public spam corpus in English, was used for this experiment. The default, unmodified rule set that comes with SpamAssassin 3.4.2 is used as a baseline for the comparison. Although this rule set is supposed to effectively detect spam for English e-mail messages in general, it was not originally trained on **D₂**. Therefore, we also re-trained its rule weights on **D₂** and included the adjusted rule set in the comparison. We use the two metrics in the previous experiment which are F_1 score and precision for presenting k-fold cross-validated results.

The default SpamAssassin rule set achieved a relatively high performance despite not being trained on the same dataset. After re-training of rule scores, the results increased significantly, especially in the precision measure. Among the three configurations of this experiment, our proposed method outperforms the other two methods with the highest values in both precision and F_1 metrics. Fig. 4 shows the relation between three performance measures across the experimented rule sets.

V. CONCLUSION

SpamAssassin rules were previously generated using the traditional approach which involves hand-engineered feature selection [6], [11], [17], [19]. In this approach, rule selection and score training are separate processes where the former one decides the outcome of the latter. However, this is a one-way influence in which score training cannot provide any feedback to help improve the quality of rule selection. In other words, feature selection is not optimized because there is not a cost function to optimize on. Contrary to that approach, our proposed method combines the two processes into a single neural network so that rule selection can also be optimized based on the final cost function (the training error). The experiments showed that our presented method is able to achieve superior performance to previous techniques on both English and Vietnamese datasets. With this model as a general framework, modifications can be made to parts of the neural network to achieve desirable effects. For example, the activation function f can be improved to include more mechanisms to improve its ability to measure a feature's quality.

REFERENCES

[1] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. In *Nature*, *323*(6088), 533.

[2] Le Cessie, S., & Van Houwelingen, J. C. (1992). Ridge estimators in logistic regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *41*(1), 191-201.

[3] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[4] Kilgarriff, A., & Salkie, R. (1996). Corpus similarity and homogeneity via word frequency. In *Proceedings of Euralex* (Vol. 96).

[5] Graham, P. (2003). Better bayesian filtering. In *Proceedings of the 2003 spam conference* (Vol. 11, pp. 15-17). Cambridge, MA.

[6] Massey, B., Thomure, M., Budrevich, R., & Long, S. (2003, June). Learning Spam: Simple Techniques For Freely-Available Software. In *USENIX Annual Technical Conference, FREENIX Track* (pp. 63-76).

[7] Ahmed, S., & Mithun, F. (2004). Word Stemming to Enhance Spam Filtering. In *In In Proceedings of Conference on Email and Anti-Spam (CEAS)*.

[8] O'Brien, C., & Vogel, C. (2004, January). Comparing SpamAssassin with CBDF email filtering. In *Proceedings of the 7th Annual CLUK Research Colloquium* (pp. 6-7).

[9] Seewald, A. K. (2004). Combining Bayesian and Rule Score Learning: Automated Tuning for SpamAssassin. *Intelligent Data Analysis. Technical report, TR-2004-11 Austrian Research Institute for Artificial Intelligence, Vienna, Austria*.

[10] Stern, H. (2004). Fast SpamAssassin score learning tool [online]. Available at *https://svn.apache.org/repos/asf/spamassassin/trunk/masses/README.perceptron*

[11] Tran, Q. A., Duan, H., & Li, X. (2006). Real-time statistical rules for spam detection. *IJCSNS International Journal of Computer Science and Network Security*, *6*(2B), 178-184.

[12] Cormack, G. V., & Lynam, T. R. (2007). TREC 2007 Public Corpus. Retrieved 2020, from *https://plg.uwaterloo.ca/~gvcormac/treccorpus07/about.html*.

[13] Huyen, N. T. M., Roussanaly, A., & Vinh, H. T. (2008, March). A hybrid approach to word segmentation of Vietnamese texts. In *International Conference on Language and Automata Theory and Applications* (pp. 240-249). Springer, Berlin, Heidelberg.

[14] Ling, S. H., Iu, H. H., Chan, K. Y., Lam, H. K., Yeung, B. C., & Leung, F. H. (2008). Hybrid particle swarm optimization with wavelet mutation and its industrial applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *38*(3), 743-763.

[15] Zeng, Z. Q., Yu, H. B., Xu, H. R., Xie, Y. Q., & Gao, J. (2008, November). Fast training support vector machines using parallel sequential minimal optimization. In *2008 3rd international conference on intelligent system and knowledge engineering* (Vol. 1, pp. 997-1001). IEEE.

[16] Xu, J. M., Fumera, G., Roli, F., & Zhou, Z. H. (2009, July). Training spamassassin with active semi-supervised learning. In *Proceedings of the 6th Conference on Email and Anti-Spam (CEAS'09)* (pp. 1-8).

[17] Songkhla, C. N., & Piromsopa, K. (2010, January). Statistical rules for thai spam detection. In *2010 Second International Conference on Future Networks* (pp. 238-242). IEEE.

[18] Caruana, G., & Li, M. (2008). A survey of emerging approaches to spam filtering. *ACM Computing Surveys (CSUR)*, *44*(2), 1-27.

[19] Dinh, Q. D., Tran, Q. A., & Jiang, F. (2014, December). Automated generation of ham rules for Vietnamese spam filtering. In *the 2014 Seventh IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)* (pp. 1-5). IEEE.

[20] Team, R. (2018). Email Statistics Report, 2018-2022. The Radicati Group.

[21] Symantec, C. (2019, February). Internet Security Threat Report 2019. Mountain View, CA, USA.

PHƯƠNG PHÁP SINH LUẬT SPAMASSASSIN DỰA TRÊN MẠNG NƠ-RON

Tóm tắt: SpamAssassin đã và đang được ứng dụng rộng rãi để lọc thư rác trên các máy chủ thư điện tử bởi vì bộ công cụ này có khả năng xử lý thời gian thực tốt và có thể dễ dàng tùy chỉnh. Tuy nhiên, SpamAssassin chỉ tích hợp sẵn hỗ trợ dành cho tiếng Anh. Mặc dù tập luật mặc định của SpamAssassin dành cho tiếng Anh được cập nhật thường xuyên, người sử dụng thường phải tự huấn luyện tập luật phù hợp với dữ liệu thư điện tử của họ. Đã có nhiều phương pháp được đề xuất cho bài toán sinh luật SpamAssassin cho nhiều ngôn ngữ như tiếng Anh [6], [9], [16], tiếng Trung [11], tiếng Thái [17] và tiếng Việt [12]. Hạn chế chung của những phương pháp này nằm ở khâu lựa chọn đặc trưng được thực hiện trực tiếp bởi chuyên gia. Công việc này đòi hỏi nhiều thời gian bởi vì dữ liệu cần được quan sát và phân tích tỉ mỉ. Trong bài báo này, các tác giả đề xuất một mô hình mạng nơ-ron nhiều lớp để sinh luật, tối ưu hóa trọng số luật và đồng thời tự động chọn ra tập thuộc tính tốt. Tập luật có trọng số thu được từ kết quả huấn luyện mô hình có thể được áp dụng trực tiếp trên phần mềm lọc thư rác SpamAssassin. Các thí nghiệm cho thấy mô hình mạng nơ-ron tốn ít thời gian để huấn luyện và tập luật được sinh ra có hiệu năng tốt so với những phương pháp sinh luật trước đó.

Từ khóa: mạng nơ-ron, sinh luật, lọc thư rác, SpamAssassin

Nguyễn Thanh Hà, hiện đang công tác tại Sở thông tin và Truyền thông Hà Nội. Đang làm nghiên cứu sinh ngành Hệ thống thông tin tại Học viện Công nghệ Bưu chính viễn thông.

Lĩnh vực nghiên cứu và chuyên môn bao gồm: AntiSpam, Công nghệ phần mềm và Hệ thống thông tin.

Email: thanhha140589@gmail.com

Đặng Đình Quân, là giảng viên tại khoa Công nghệ thông tin – Trường Đại học Hà Nội.

Lĩnh vực nghiên cứu và chuyên môn bao gồm: AntiSpam, Học máy và Giải thuật tiến hóa.

Email: quandd@hanu.edu.vn