Information Fusion 48 (2019) 67–83

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

An intelligent system for spam detection and identification of the most relevant features based on evolutionary Random Weight Networks

Hossam Faris[a], Ala' M. Al-Zoubi[a], Ali Asghar Heidari[b], Ibrahim Aljarah[a], Majdi Mafarja[*,c], Mohammad A. Hassonah[a], Hamido Fujita[d]

[a] King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
[b] School of Surveying and Geospatial Engineering, University of Tehran, Tehran, Iran
[c] Department of Computer Science, Birzeit University, Birzeit, Palestine
[d] Faculty of Software and Information Science, Iwate Prefectural University (IPU), Iwate, Japan

ARTICLE INFO

Keywords:
Spam filtering
Email spam detection
Feature analysis
Hybrid machine learning
Evolutionary
Random Weight Network
Feature selection

ABSTRACT

With the incremental use of emails as an essential and popular communication mean over the Internet, there comes a serious threat that impacts the Internet and the society. This problem is known as spam. By receiving spam messages, Internet users are exposed to security issues, and minors are exposed to inappropriate contents. Moreover, spam messages waste resources in terms of storage, bandwidth, and productivity. What makes the problem worse is that spammers keep inventing new techniques to dodge spam filters. On the other side, the massive data flow of hundreds of millions of individuals, and the large number of attributes make the problem more cumbersome and complex. Therefore, proposing evolutionary and adaptable spam detection models becomes a necessity. In this paper, an intelligent detection system that is based on Genetic Algorithm (GA) and Random Weight Network (RWN) is proposed to deal with email spam detection tasks. In addition, an automatic identification capability is also embedded in the proposed system to detect the most relevant features during the detection process. The proposed system is intensively evaluated through a series of extensive experiments based on three email corpora. The experimental results confirm that the proposed system can achieve remarkable results in terms of accuracy, precision, and recall. Furthermore, the proposed detection system can automatically identify the most relevant features of the spam emails.

1. Introduction

The emailing system is an essential communication mean with billions of daily users over the Internet. With this system, there comes a problem which is known as the "spam". This term commonly refers to unwanted, junk messages sent to an Internet user's inbox. Spam is considered as a serious threat that impacts the Internet and the society [1]. By receiving spam messages, Internet users are exposed to security issues, and minors are exposed to illegal and improper contents. Moreover, spam messages waste valuable resources, including storage, bandwidth, and productivity [2]. Therefore, there is a high demand for automatic email spam filtering [3].

Although there is a continuous effort by researchers and practitioners to develop accurate spam detection systems, Internet users still receive tremendous amounts of spam messages, everyday. With aid of spam campaigns, malware and botnets, spammers can send thousands of messages at low or no cost at all [2].

One of the main approaches to handle the spam problem is to use spam filters. These filters identify spam emails based on the analysis of their contents and other additional information [4]. At the beginning, spam filters were developed based on a set of user-defined rules, keyword-filtering, or black-lists of recognized spammers. However, such techniques have to be maintained and updated continuously, which make them inefficient and time-consuming. This approach is known as the knowledge engineering approach.

To overcome the drawbacks of knowledge engineering approaches, a recent trend in the development of spam filters relies on learning-based classification techniques [5]. Machine learning algorithms proved to be more efficient than user-defined rules approaches. They are more flexible, dynamic, and easier to be adapted to new mechanisms followed by spammers in spreading unsolicited messages.

In literature, most of the studies, which are based on learning-based

* Corresponding author.
E-mail addresses: hossam.faris@ju.edu.jo (H. Faris), alaah14@gmail.com (A.M. Al-Zoubi), as_heidari@ut.ac.ir (A.A. Heidari), i.aljarah@ju.edu.jo (I. Aljarah), mmafarja@birzeit.edu (M. Mafarja), mohammad.a.hassonah@gmail.com (M.A. Hassonah), hfujita-799@acm.org (H. Fujita).

https://doi.org/10.1016/j.inffus.2018.08.002
Received 2 June 2018; Received in revised form 1 August 2018; Accepted 4 August 2018
Available online 06 August 2018
1566-2535/

algorithms, have focused on increasing the accuracy of spam detection models, however, few of them studied the relevancy of features and their predictive power. For example, different types of features can be collected and extracted to train a machine learning model, but not all features are relevant and helpful in increasing the accuracy of the model. Moreover, it has been reported in many studies that spammers tend to keep changing their techniques and methods to avoid spam filters. Such behaviour could generate spam emails in unpredictable patterns for trivial spam filters. This forms a strong motivation to develop an efficient and accurate spam detection model that can automatically identify the relevant features and give an insight on their importance in the detection process.

In this work, we propose a hybrid email spam detection system that combines RWN and GA. The proposed system performs automatic Feature Selection (FS) and email classification based on two main stages. The FS stage is designed as a wrapper approach, while the classification stage is handled by RWN. The main advantages of the proposed system include: the system can achieve high classification accuracy, and it can measure the relevant impact of the involved features in the training process. The second advantage will give an insight for the designers of spam detection systems on which features have more significant role in the detection process. This will pave the way for implementing more robust and efficient spam detection systems. Therefore, the main contributions of this work can be summarized in four points:

- A new efficient spam detection model is developed, which is called Auto-GA-RWN.
- The proposed model automatically identifies the most relevant features in the detection process.
- Unlike most of the wrapper-based FS methods proposed in the literature that utilize k-Nearest Neighbor (k-NN), Auto-GA-RWN utilizes RWN as the base classifier; in order to take the advantage of its potentially higher generalization power.
- In addition, the proposed method automatically tunes the number of hidden neurons in RWN to save tremendous efforts which can be spent in tuning this parameter. This is accomplished during the FS process by extending the chromosomes in GA where each chromosome will have two parts: one to represent the selected features and one to represent the number of hidden neurons.

This paper is organized as follows: Section 2 discusses some of the important approaches proposed for spam detection models. In Section 3, the preliminaries of this work are explained. The methodology and proposed approach are described in detail in Section 4. Afterwards, the conducted experiments are discussed in Section 5. Finally, we conclude the main findings of this work and highlight future directions in Section 6.

2. Related works

In literature, various approaches have been proposed for tackling spam detection tasks. After reviewing the most related approaches in literature, we found that the previous research studies can be classified according to the following taxonomy: single standard machine learning-based approaches, hybrid machine learning approaches, and feature engineering approaches.

In first category of the proposed taxonomy, a single machine learning-based method is utilized to investigate the spam detection problem. In this type of methods, machine learning is used to build the classification models based on email's attributes, which represent the features used to identify spam emails [5]. Drucker et al. [6] applied Support Vector Machines (SVM) to the spam detection problem. The experimental results showed that their method can significantly outperform other classifiers in terms of detection rate and training time. Another work established by Amayri and Bouguila [7] also applied SVM classifier to this problem. They studied the impact of SVM' kernels on separation of spam emails from ham emails. They showed that string kernels can achieve superior results compared to distance-based kernels.

Another popular machine learning method used in spam detection is the Bayesian classifier [8]. Metsis et al. [8] evaluated five different forms of Naive Bayes (NB) classifier, which are Multi-variate Bernoulli NB, Multinomial NB, Flexible Bayes, Flexible Bayes, and Multi-variate Gauss NB. Their experimental evaluation based on ROC curves revealed the satisfactory performance of Flexible Bayes and Multinomial NB with boolean attributes. Sahami et al. [9] applied standard Bayesian Network (BN) classifier to junk emails detection tasks. The experimental results on real datasets showed BN can find promising results in terms of detection rate. Other standard machine learning classification methods such as Artificial Neural Networks (ANN) in [10,11], k-NN in [12], and Decision Trees [13] have been also utilized to deal with spam detection problems.

Aski and Sourati [14] proposed a spam filter using three well-known machine learning techniques, namely; Decision Tree classifier (C4.5), Multi-Layer Perceptron (MLP), and NB classifier. Results of this research showed that using MLP with spam filters can lead to more accurate results compared to other techniques. Another study presented in [15] concentrated on collecting the most used features in spam detection systems and investigated the importance of each feature with regard to information gain measure. Idris and Selamat [16] proposed an improved spam detection approach that combined the Negative Selection Algorithm (NSA) with Particle Swarm Optimizer (PSO). The main finding of this work was that the combined approach (NSA-PSO) can outperform the original NSA. Similar approaches that hybridized NSA with Differential Evolution (DE) and PSO were proposed in [17] and [18], respectively.

Following the proposed taxonomy, hybrid machine learning approaches create another line of research. Faris et al. [19] proposed a spam filter approach with two phases; in the first phase, the most informative features were selected based on a wrapper FS approach with PSO, and then, in the second phase, the selected features were used to build a spam filter model using the Random Forest technique. In [20], another spam detection approach was introduced, which works based on a three-layer Back-propagation Neural Network (BPNN) with a Concentration-based Feature Construction (CFC) technique. In addition, Faris et al. [21] and Wu [22] proposed neural network based spam detection approaches.

Zhang et al. [23] presented a wrapper FS approach for spam detection problem. In their approach, a mutation-based PSO was employed as a search strategy and the fitness function aimed at reducing two error types in the confusion matrix; the false positive and the false negative. While Gavrilis et al. [24] proposed a hybrid filter-wrapper FS approach, where entropy was used to reduce the dataset in the first step, and a fine-tuned GA was used in the second step. Mohammad and Zitar [2] also proposed a hybrid approach based on Artificial Immune System (AIS) and GA for spam detection.

The third category of the proposed taxonomy is based on feature engineering. Such works have studied the spam detection task based on proposing a new or modified set of features. For instance, Alqatawna et al. [25] proposed a spam detection approach based on the extracted features from a set of emails. The main challenge for this approach is faced when using an imbalanced dataset, where just 16% of the instances were classified as spam. In [26], the important features were extracted, then, three classifiers were combined using voting schemes to enhance the performance of the classical algorithms in identifying whether an email is a spam or ham.

Overall, it can be understood from the literature that there is a strong tendency towards the application of machine learning methods to spam detection. This is due to their potential learning capabilities. On other hand, it is seen that most of the proposed approaches focused on improving the detection accuracy of spam detection models,

however, only few of them have paid a closer attention to identifying the influence of the features and their types on the accuracy of spam detection.

3. Preliminaries

3.1. An overview of Genetic algorithms

Genetic algorithms (GAs) are well-regarded class of iterative evolutionary optimization techniques introduced by John Holland and inspired by philosophies and evolutions of biological processes [27]. GAs have been extensively utilized in different optimization, FS, and machine learning tasks due to their simplicity, flexibility, ease of implementation, satisfactory efficiency on the majority of problems, and the strong background with regard to the source of inspiration and operations [28–30]. Over the past years, GAs have been broadly investigated in literature and numerous encoding schemes have been established for various practical problems [31,32]. The evolutionary phases in GAs initiate with a random population of candidate solutions (individual genomes, search agents or phenotypes). Each agent has a pool of chromosomes or genotypes which have to be mutated and enriched during the exploration (diversification) and exploitation (intensification) phases of this algorithm [33]. Subsequently, every generation can generate an offspring population based on three core procedures: selection, crossover, and mutation. These mechanisms are inspired from the notion of natural selection in attaining the best candidate gene while maintaining the diversity to evade immature convergence and stagnation to Local Optima (LO) during the GA-based optimization steps [34].

3.1.1. Selection

All through every consecutive generation, a fraction of the current agents should be selected in order to generate (breed) a fresh generation. Selection operators are utilized to select those offspring solutions that can survive to be more evolved during the next generations. There are three types of selection operators widely accepted and applied in previous works [35]: Tournament Selection (TS), Roulette Wheel Selection (RWS) and Random Selection (RS). TS can be considered as the most-employed selection scheme within GAs because of its performance and ease of implementation [36]. In this strategy, a set of agents are selected from the reference pool in a random fashion, and then, the best one among the nominated agents are selected to be further evolved during the next generations of the algorithm. The total number of nominated agents are often called the tournament size *TS* [37]. Fig. 1 shows the outline of TS mechanism [38].

In RWS, agents are selected based on a probability which is calculated using the related fitness values [39]. In this method, a roulette wheel is constructed using a circumference that is equal to the sum of the score of all agents [38]. In RWS, all agents have a section with an area proportional to their fitness scores. The probability rate to choose an agent is found by rotating a roulette wheel, and that section where a pointer stays is used and the matching agent is selected. By this manner, the agents that have a larger fitness rates (i.e. bigger areas) have more chance to be selected than those having lower fitness values (smaller areas). Advantage of this scheme is that it will not disregard any agent

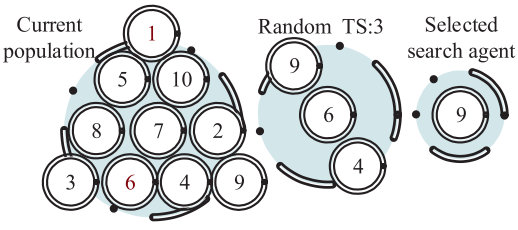


Fig. 1. The mechanism of TS.

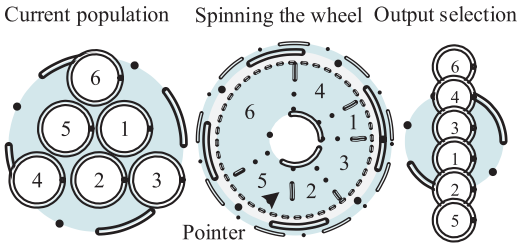


Fig. 2. Selection strategy with roulette wheel mechanism.

in the pool; consequently, it maintains the diversity of agents within feature space. Fig. 2 shows the outline of RWS mechanism.

3.1.2. Crossover

To generate a second set of agents based on those selected in selection stage, two other phases are utilized: crossover (also known as recombination) [40], and mutation [41]. Crossover operators can amalgam two or more parent agents to attain similar, but somewhat dissimilar offspring agents (children). Often, crossover techniques convert the agents into binary form (string) to carry out the processes. There are two widely used classes of crossover methods, one-point crossover, which can cross the binary values at specific crossover points of two parent agents for constructing two new child agents, and two-point crossover, which utilizes two crossover points.

3.1.3. Mutation

Mutation mechanisms are employed in GAs to maintain and improve the genetic diversity of agents to guarantee avoiding premature convergence during next generations [42]. This operation should revise one or more gene values in a chromosome to generate a new agent, which may be better than the previous one. Mutation can happen throughout the evolutionary process based on a predetermined mutation probability, which should be set to low; since a high value can turn the GA into a basic random search. According to different types of genome, several mutation classes can be utilized: bit string, boundary, flip bit, uniform, non-uniform, Gaussian, and Shrink [43].

3.2. Random Weight Networks (RWN)

RWNs were initially designed by Schmidt et al. [44] in 1992 to handle the training process of single-hidden layer feed-forward neural networks (SLFNs) [45]. In unison, Pao et al. [46] developed a comparable model with a similar structure, which was called random vector functional-link networks (RVFLNs). The RWN aims at not only generalizing the SLFN but also multi hidden-layer feed forward networks where a node can be a subnetwork involving extra hidden nodes. Compared to the conventional gradient descent methods that are used to train SLFN, the RWN learning speed is extremely fast and has better generalization performance. Moreover, unlike conventional methods such as backpropagation which needs to manually set the parameters (learning rate, number of epochs, etc.), the RWN does not require intensive human intervention. Furthermore, RWN does not work iteratively to optimize the SLFN parameters. Instead, it randomly initializes the input weights and the hidden biases, then it determines the output weights analytically using Moore–Penrose generalized inverse [47]. To realize the learning and generalization traits of the RVFLNs, reader can refer to Pao et al. [46], and to investigate the approximation power of the RVFLNs, he/she can refer to Igelnik and Pao [48]. In RVFLNs, we have direct links from input layer to output layer, which, in fact, this is the only distinction between RWNs and RVFLNs. With regard to the high learning speed, generalization and approximation capabilities of RWNs and RVFLNs, they have considerably attracted researchers in different fields [49]. The overall structure of RWN is depicted in Fig. 3.

For N arbitrary distinct samples shown by (x_i, t_i), where

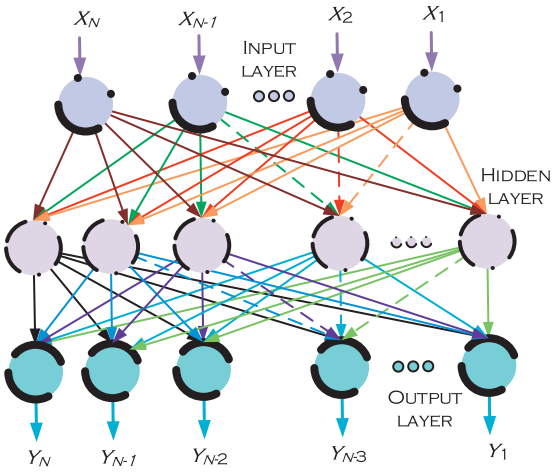


Fig. 3. Overall structure of RWN.

$x_i = [x_{i1}, x_{i2}, x_{iN}]^T \in \mathbb{R}^n$ and $t_i = [t_{i1}, t_{i2}, t_{iN}]^T \in \mathbb{R}^m$, SLFNs with activation function $g(x)$ and $\widetilde{N}$ hidden neurons can be mathematically represented as in Eq. (1) [50]:

$$\sum_{i=1}^{\widetilde{N}} \beta_i g(\omega_i \cdot x_j + b_i) = o_j, j = 1, \cdots, N \tag{1}$$

where b_i denotes ith hidden neuron' threshold, $w_i = [w_{i1}, w_{i2}, \cdots, w_{in}]^T$ is the weight vector to connect the ith hidden neuron to the input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]^T$ is the vector of weights to connect the ith hidden neuron to the output neurons [50].

The conventional SLFNs with $\widetilde{N}$ hidden neurons and activation function $g(x)$ are capable of approximating the initial N samples with zero error means that $\sum_{i=1}^{\widetilde{N}} \|o_j - t_j\| = 0$, i.e., we have w_i, β_i, and b_i such that [50]:

$$\sum_{i=1}^{\widetilde{N}} \beta_i g(\omega_i \cdot x_j + b_i) = t_j, j = 1, \cdots, N \tag{2}$$

The above-mentioned N rules can be rewritten as expressed in Eq. (4):

$$H\beta = T \tag{3}$$

where

$$H(w_1, \cdots, w_N, b_1, \cdots, b_N, x_1, \cdots, x_N) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\widetilde{N}} \cdot x_1 + b_{\widetilde{N}}) \\ \vdots & & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\widetilde{N}} \cdot x_N + b_{\widetilde{N}}) \end{bmatrix}_{N \times \widetilde{N}} \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\widetilde{N}}^T \end{bmatrix}_{\widetilde{N} \times m}, T = \begin{bmatrix} t_1^T \\ \vdots \\ t_{\widetilde{N}}^T \end{bmatrix}_{N \times m} \tag{5}$$

where H is the hidden layer output matrix, the ith column of H is the ith hidden output vector of neuron with regard to $x_1, x_2, \cdots, x_N$ [50].

4. Methodology

This work is conducted based on a methodology of five main stages which are: feature extraction, FS, model development, evaluation and assessment, and feature importance analysis. A high level diagram of this methodology is given in Fig. 4, which shows the sequence of the main stages. A more insightful diagram is depicted in Fig. 5. The main contribution lies in FS and model development stages, where a new model is proposed based on GA and RWN. All stages of the methodology are discussed in detail in the following subsections.

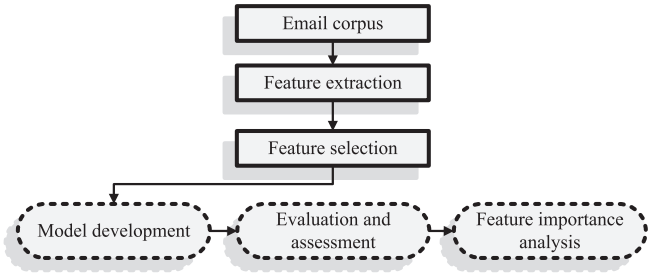


Fig. 4. Abstract description of the methodology.

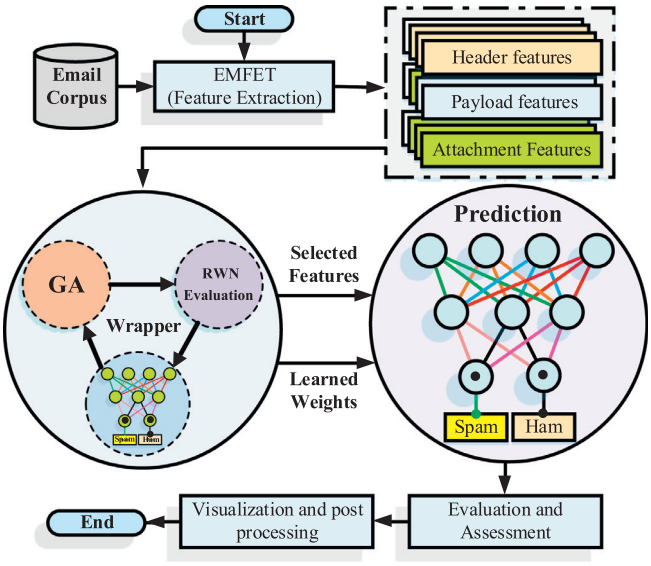


Fig. 5. Detailed description of the methodology.

4.1. Dataset description and feature extraction

In this work, three datasets are constructed based on three public email spam corpora which are SpamAssassin,[1] LingSpam[2] and CSDMC2010 Corpus.[3] Table 1 shows the detailed description of the used spam corpora.

In order to convert the email messages into features that can be processed by the machine learning algorithms, the E-mail Features Extraction Tool (EMFET[4]) is utilized. The EMFET is an open source tool developed by our team and it is publicly available [51]. By using EMFIT, 140 features can be extracted. These features can be categorized as follows: 49 header features, 2 attachment features, and 89 Payload features. More details about each type of extracted features are explained in the following paragraphs:

- Header features: The header of an email is one of the main three components that any email consists of, and it is considered as an essential part that identifies the rest of topic in an email. Features extracted from such component are important in email delivery, which includes email metadata such as the sender, recipient, subject and, the date, along with the time that the email has been sent in.
- Payload features: This type of features is the second component of

[1] The SpamAssassin dataset is available at http://spamassassin.org/publiccorpus/.
[2] The Ling-Spam dataset is available at http://www.aueb.gr/users/ion/data/.
[3] The CSDMC2010 Corpus is available at http://www.aueb.gr/users/ion/data/.
[4] EMFET Tool is available at https://github.com/WadeaHijjawi/EmailFeaturesExtraction.

Table 1
Description of spam corpora.

Dataset	Total messages	Spam rate
SpamAssassin	8150	14.69%
LingSpam	2894	16.63%
CSDMC2010	4327	31.85%

the email structure, and it can be further divided into three subcategories, each one of them has its own characteristic. The first one is the email body features that contain unstructured data such as text, HTML tags, images and other objects. The second part is the Readability features which represent the properties of reading a text in an email body, this text might be a word, a sentence or a paragraph. This sort of features is usually utilized to differentiate between simple and complex words that depend on the sequence of speech sounds. The last category is the Lexical Diversity features, which can be extracted from the vocabulary size and other measures such as occurrences count.

- Attachment features: Attachment in an email usually contains files that can be related or not to the subject of the email message. This component consists of two features, which are the number of attachments the email contains, and the unique count of the attached files' types.

Fig. 6 shows the number of extracted features from each category. For a full list of the features, we refer the interested reader to Appendix A [51,59].

4.2. Proposed approach: automatic GA-RWN wrapper-based feature selection (auto-GA-RWN)

In this work, we apply the FS step based on the training part of the dataset to eliminate irrelevant features, reduce the dimensionality of the data, and to increase the classification accuracy. To perform FS, we adopt a wrapper-based approach. The wrapper consists of three main components: the search algorithm, the induction algorithm (i.e learning algorithm) and feature evaluation.

Wrappers are more preferable than other methods like filters when the accuracy is more important than speed. This is because they search for a subset of features that maximizes the accuracy for a particular classifier. Therefore, the base classifier is usually selected to be the same as the final classification algorithm which will be applied on the testing set. On the other hand, the disadvantage of the wrappers is that they are computationally intensive. For that reason, most of the studies in the literature utilize a fast induction algorithm such as k-NN and decision trees. In our proposed approach, Auto-GA-RWN utilizes RWN as the base classifier; in addition to the fast induction property, RWN has high generalization power.

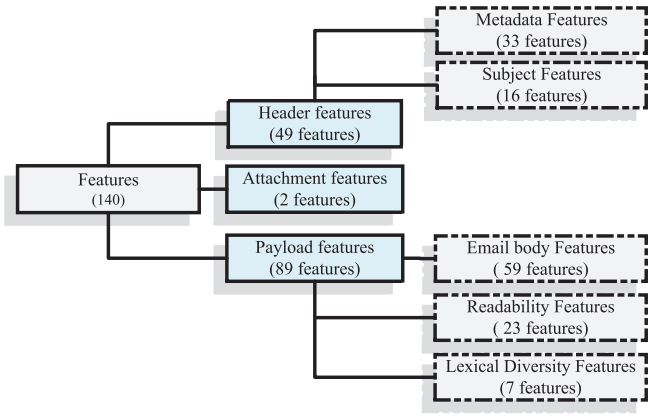


Fig. 6. Number of extracted features from each category.

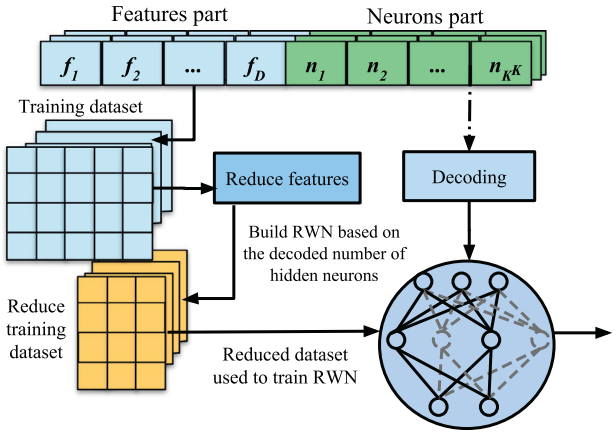


Fig. 7. Solution representation in Auto-GA-RWN.

In our wrapper-based technique, GA is selected to search the combinatorial search space of the possible feature subsets. GA has shown very promising exploratory performance for different combinatorial problems in the literature including FS [52].

4.2.1. Design issues

In order to apply any optimizer for a given task, there are two design concerns to be carefully addressed: first, the design of the solution representation, and second, the selection of the fitness function. For our proposed approach, these points are addressed as follows:

- Solution representation: In our proposed model, the chromosome of GA is designed to represent a candidate solution as a one-dimensional binary vector with two parts as shown in Fig. 7. The first part is a set of binary flags, which indicate if the corresponding features are selected or not, that is, if the value of a given gene i is 0, the feature F_i is not selected, and if the value of gene i is 1, this means that feature F_i is selected. The second part of the chromosome consists of a set of binary flags that determine the number of neurons in the hidden layer of RWN. The length of the chromosome is $D + K$, where D is the number of features in the dataset (i.e. dimensionality of the dataset), and K is the number of reserved bits for representing the maximum number of hidden neurons.
- Fitness function: A fitness function is needed in metaheuristic algorithms to assess the quality of the generated possible solutions. In our model, we propose a fitness function that considers three objectives which are increasing the classification accuracy of the RWN network, decreasing the number of selected features, and to decrease the complexity which is quantified by the number of hidden neurons in the RWN. The fitness function can be calculated as in Eq. (6):

$$Fitness = \alpha Err + \beta\frac{f}{F} + \gamma\frac{n}{N} \tag{6}$$

where Err is the classification error rate of the RWN classier, f denotes the number of selected features, F is the total number of features, n is number of neurons determined by the evaluated GA chromosome, and N is the maximum number of possible neurons in the RWN network. The parameters α, β, and γ are three factors to control the weight of each of the classification rate component, the number of selected features, and the number of neurons in the fitness function. The values of α, β, γ are set within [0,1], usually $\beta = (1 - \alpha)$ as in [53]. The maximum number of neurons is set to 1024, which is represented by 10 elements in the chromosome.

4.2.2. Auto-GA-RWN procedure

The procedure of the proposed Auto-GA-RWN algorithm can be

described as in the following steps:

- Initialization: At first, the algorithm starts by randomly initializing a population of m chromosomes, each of which is represented as binary random vectors.
- Genotype–Phenotype mapping (Feature selection and RWN construction): Before evaluating the generated candidate solutions in GA, it is required to map the chromosomes, the genotypes, to the actual solution, which are known as phenotypes [54]. As described and shown before in Fig. 7, the first part of the chromosome is used to perform FS based on the training part of the dataset. This will result as a reduced training dataset. The second part of the chromosome will determine the number of hidden neurons in the RWN network. The new training dataset and along with the number of hidden neurons will be used to construct a candidate RWN network, which will be evaluated as described in the next step.
- Fitness evaluation: The prediction power of all generated RWN networks from the previous step is evaluated using the fitness function expressed in Eq. (6). The obtained fitness values will influence the selection mechanism in the next step.
- Selection and reproduction: In this step, the reproduction operators (i.e. the selection mechanism, crossover operator, and mutation process) are applied to produce a new generation of RWN networks.
- Termination: The aforementioned three processes are applied over the course of iterations until the maximum number of iterations is met. After the termination, the algorithm will return the best obtained set of features along with the best obtained RWN network represented by the number of hidden neurons and their learning weights.

4.3. Evaluation and assessment

In this stage, the predictive power of the best obtained RWN network will be evaluated. Since the target problem is a binary classification problem, the measurements listed below will be used for assessment. All the measurements are calculated based on the confusion matrix shown in Fig. 8. The positive and negative classes refer to the spam (S) and ham (H) classes, respectively. Note that precision and recall are calculated for each class.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}. \tag{7}$$

$$Precision_S = \frac{TP}{TP + FP}. \tag{8}$$

$$Precision_H = \frac{TN}{TN + FN}. \tag{9}$$

$$Recall_S = \frac{TP}{TP + FN}. \tag{10}$$

$$Recall_H = \frac{TN}{TN + FP}. \tag{11}$$

		Actual class	
		Positive	Negative
Predicted class	Positive	True Positive TP	False Positive FP
	Negative	False Negative FN	True Negative TN

Fig. 8. Confusion matrix.

$$G - Mean = \sqrt{Recall_S \times Recall_H} \tag{12}$$

4.4. Feature importance analysis

More analyses is conducted in this stage to study the relative importance of the input features in the case of email spam classification problem. Identifying the most influencing features could effectively assist us in designing more accurate spam filters. In this work, we adopt an approach that relies on the appearance frequency of the features in the best obtained solutions. We calculate the relative variable relevance of a given feature as follows:

Suppose X is a set of input features $\{x_1, x_2, ..., x_D\}$, where $x_i \in X$. The frequency of feature x_i with respect to a set of B best models is defined as:

$$RefCount(x_i, B) = \sum_{i=1}^{B} f_i \tag{13}$$

where f_i is the corresponding binary flag of the feature x_i

Consequently, the frequency of x_i with respect to a set of B best models can be defined as:

$$rel(x_i, B) = \frac{RefCount(x_i, B)}{B} \tag{14}$$

5. Experimental results and discussion

In this section, a comparative analysis is presented to investigate, in depth, the performance of the proposed Auto-GA-RWN for classification and FS. All observed assessments and comparisons were performed and recorded using a PC with Intel Core(TM) i5-5200U 2.2 GHz CPU and 4.0GB RAM. All algorithms are implemented in MATLAB 2013 software. For training and testing, 10-folds cross validation is applied and repeated three times.

For this, the following experiments and analysis will be conducted:

- Experiment I: In this experiment, we compare the performance of RWN to other well-known classifiers that are commonly employed in the literature as induction classification algorithms in wrapper-based FS methods. The purpose of this experiment is to prove the motivation for using RWN as the base classifier.
- Experiment II: In this experiment, we test the performance of GA-RWN that performs classification and FS, however, the number of neurons is manually set. The goal of this experiment is to show that the performance of RWN can vary according to the number of hidden neurons. Therefore, there is a need for an automatic way to tune this number.
- Experiment III: In this part, we compare the performance of the proposed Auto-GA-RWN model to the manually-tuned GA-RWN from the previous experiment. In addition, the Auto-GA-RWN is compared to other common classification approaches from the literature.
- Importance analysis: In the last stage of this section, we conduct an analysis to study the relative importance of the input features.

5.1. Experiment I: comparing RWN to other induction algorithms

In this experiment, we evaluate the performance of RWN for spam classification without performing any FS based on the three spam datasets. In addition, we compare its performance to other algorithms that are commonly used as induction algorithms in wrapper-based FS methods, which are k-nearest neighbor (k-NN) with $k = 1$, 3, and 5 (i.e. 1-NN, 2-NN, and 3-NN), Naïve Bayes (NB), and SVM. Note that the hyperparameters of SVM have optimized using the grid search, where the range of the cost (C) is [0.01,1.0], and the gamma (γ) from 0.001 to 0.5. Number of hidden neurons in RWN is empirically tuned in the

Table 2
Basic classifier results for SpamAssassin dataset.

Classifiers	Accuracy		$Recall_S$		$Recall_H$		$Precision_S$		$Precision_H$		G-Mean	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
NB	0.899	0.009	0.575	0.059	**0.955**	0.005	0.688	0.031	0.929	0.009	0.740	0.038
1-NN	0.919	0.000	0.971	0.000	0.936	0.000	0.627	0.000	0.791	0.000	0.780	0.000
3-NN	0.904	0.000	0.973	0.000	0.918	0.000	0.515	0.000	0.775	0.000	0.708	0.000
5-NN	0.896	0.000	0.975	0.000	0.909	0.000	0.455	0.000	0.763	0.000	0.666	0.000
Grid-SVM	0.854	0.024	0.005	0.002	**1.000**	0.000	**1.000**	0.000	0.854	0.000	0.068	0.013
RWN	**0.922**	0.004	**0.976**	0.004	0.919	0.004	0.793	0.012	**0.933**	0.011	**0.880**	0.007

Table 3
Basic classifier results for Lingspam dataset.

Classifiers	Accuracy		$Recall_S$		$Recall_H$		$Precision_S$		$Precision_H$		G-Mean	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
NB	0.874	0.015	0.759	0.083	0.897	0.028	0.601	0.046	**0.950**	0.016	**0.823**	0.016
1-NN	0.808	0.000	0.889	0.000	0.882	0.000	0.404	0.000	0.420	0.000	0.599	0.000
3-NN	0.826	0.000	**0.919**	0.000	0.878	0.000	0.358	0.000	0.467	0.000	0.574	0.000
5-NN	0.809	0.000	0.916	0.000	0.863	0.000	0.271	0.000	0.392	0.000	0.498	0.000
Grid-SVM	0.843	0.164	0.057	0.009	**1.000**	0.000	**1.000**	0.000	0.842	0.001	0.238	0.019
RWN	**0.895**	0.016	0.479	0.096	0.811	0.047	0.978	0.006	0.904	0.016	0.681	0.070

interval [20,100].

Table 2 compares the results of different basic classifiers in tackling SpamAssassin dataset. As per results in Table 2, we can observe that the RWN-based classifier has attained the maximum classification accuracy, which is followed by the 1-NN, 3-NN, NB, 5-NN, and Grid-SVM, respectively.

Table 3 reflects the comparison results for basic classifiers in dealing with the Lingspam dataset. As per results in Table 3, we can see that the RWN classifier has also obtained the maximum classification accuracy. The NB, Grid-SVM, 3-NN, 5-NN, and 1-NN has attained the next rates, respectively.

Table 4 reveals the basic classifier results for the evaluated CSDMC2010 dataset. It can be observed in Table 4 that the RWN classifier has again attained the best average accuracy compared to the other competitors. The 1-NN, 5-NN, Grid-SVM, 3-NN, and NB have attained the next places, respectively.

Overall, it can be noticed that RWN classifier has a better performance than 1-NN, 3-NN, 5-NN, Grid-SVM and NB in dealing with all datasets.

5.2. Experiment II: evaluation of GA-RWN with manual tuning of hidden neurons

Number of neurons in the hidden layer can have a high impact on the resulting performance of the RWN network. Moreover, it is well known that the RWN requires much more neurons than the classical MLP network. In order to determine the best number of neurons in GA-RWN, different numbers of neurons were experimented on different datasets, starting from 20 neurons to 1000. The results of these experiments are reported in Table 5 and depicted in Fig. 9.

As per results for SpamAssassin dataset in Table 5, it is observed that the evolutionary-based RWN with 600 neurons can reach to the highest accuracy rates. As it can be seen in Fig. 9(a), there is a gap between the classification rates of evolved RWN versus the standard version. Based on accuracy results for LingSpam dataset, it is seen that the GA-RWN_{Roul} has reached to the highest classification rates using 50 neurons, while GA-RWN_{Rand} has touched the best classification rates using 40 neurons. The GA-RWN_{Tour} only needs 30 neurons to classify with highest accuracies. As it can be seen in Fig. 9(b), the variation of rates is higher than other datasets, but the GA-based techniques have enhanced the classification rates compared to the classic RWN. From the results for CSDMC2010 dataset, it is detected that all evolutionary-based RWN approaches have reached the best results using 400 neurons, while the $RWN_{Classic}$ needs 300 neurons. As it can be seen in Fig. 9(c), the classification rates of the classic RWN show a decreasing trend when the number of neurons is greater than 500.

This experiment showed that a tremendous effort is needed for determining the best number of hidden neurons in RWN. Moreover, the results showed that this number highly depends on the dataset, and it can be varied from one dataset to another. Therefore, there is strong need for automatic approach for tuning the hidden neurons in RWN.

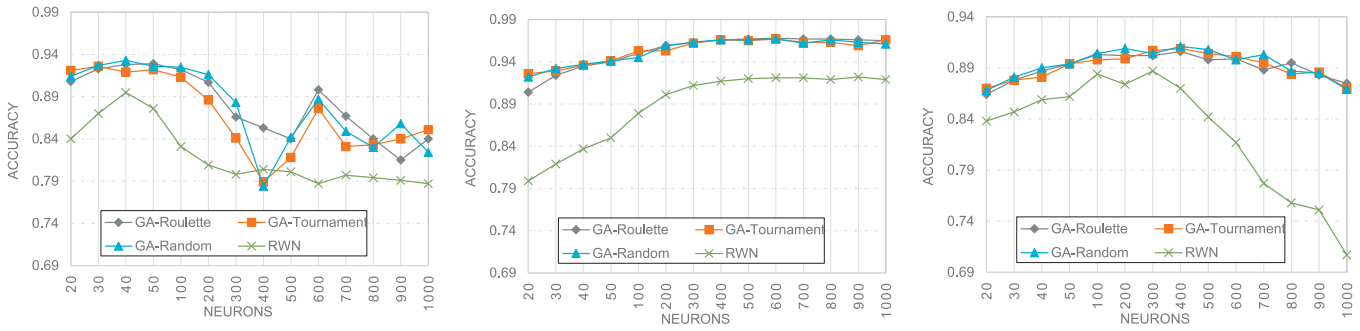
Table 6 reveals the comparative classification results for SpamAssassin dataset when 1-NN and RWN are utilized as the base classifiers along with different GA-based algorithms. Inspecting the results in Table 6, we see that the 1-NN joint with GA using TS has reached to the finest accuracy level compared to other selection techniques inside

Table 4
Basic classifier results for CSDMC2010 dataset.

Classifiers	Accuracy		$Recall_S$		$Recall_H$		$Precision_S$		$Precision_H$		G-Mean	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
NB	0.822	0.009	0.531	0.02	**0.958**	0.01	0.856	0.029	0.814	0.007	0.713	0.014
1-NN	0.872	0.000	0.926	0.000	0.889	0.000	0.758	0.000	0.829	0.000	0.838	0.000
3-NN	0.863	0.000	0.943	0.000	0.866	0.000	0.694	0.000	0.854	0.000	0.809	0.000
5-NN	0.867	0.000	**0.958**	0.000	0.861	0.000	0.677	0.000	0.886	0.000	0.805	0.000
Grid-SVM	0.867	0.131	0.935	0.015	0.720	0.017	0.877	0.007	0.839	0.032	0.821	0.013
RWN	**0.887**	0.017	0.846	0.062	0.824	0.053	**0.908**	0.039	**0.924**	0.026	**0.875**	0.023

Table 5
Comparison of the classification results for different RWN-based networks with different number of neurons on all datasets.

Neurons	Algorithms	20	30	40	50	100	200	300	400	500	600	700	800	900	1000
SpamAssassin	GA-RWN_*Roul*	0.904	0.924	0.935	0.940	0.950	0.959	0.963	0.966	0.967	**0.968**	0.967	0.967	0.966	0.965
	GA-RWN_*Tour*	0.926	0.929	0.936	0.941	0.953	0.953	0.962	0.966	0.965	**0.967**	0.963	0.963	0.959	0.966
	GA-RWN_*Rand*	0.922	0.932	0.937	0.941	0.945	0.959	0.963	0.966	0.966	**0.967**	0.962	0.966	0.963	0.961
	RWN_*Classic*	0.799	0.819	0.837	0.850	0.879	0.901	0.912	0.917	0.920	0.921	0.921	0.919	**0.922**	0.919
LingSpam	GA-RWN_*Roul*	0.908	0.923	0.928	**0.929**	0.922	0.907	0.866	0.853	0.840	0.898	0.867	0.840	0.815	0.840
	GA-RWN_*Tour*	0.921	**0.926**	0.919	0.922	0.913	0.886	0.841	0.789	0.818	0.876	0.831	0.833	0.840	0.851
	GA-RWN_*Rand*	0.914	0.927	**0.933**	0.926	0.925	0.916	0.883	0.784	0.842	0.887	0.849	0.830	0.858	0.824
	RWN_*Classic*	0.840	0.870	**0.895**	0.876	0.831	0.809	0.798	0.804	0.801	0.787	0.797	0.794	0.791	0.787
CSDMC2010	GA-RWN_*Roul*	0.864	0.878	0.887	0.894	0.903	0.902	0.902	**0.906**	0.898	0.899	0.888	0.895	0.883	0.875
	GA-RWN_*Tour*	0.870	0.878	0.881	0.894	0.898	0.899	0.907	**0.909**	0.904	0.901	0.895	0.884	0.886	0.870
	GA-RWN_*Rand*	0.868	0.881	0.890	0.894	0.904	0.909	0.904	**0.911**	0.908	0.898	0.903	0.887	0.885	0.869
	RWN_*Classic*	0.838	0.847	0.859	0.862	0.884	0.874	**0.887**	0.870	0.842	0.817	0.777	0.758	0.751	0.707



(a) SpamAssassin dataset (b) LingSpam dataset (c) CSDMC2010 dataset

Fig. 9. Impact of the number of hidden neurons on the performance of the RWN networks for different datasets.

Table 6
Comparison of the classification results for GA-1-NN and GA-RWN for SpamAssassin dataset.

Algorithm	Accuracy		$Recall_S$		$Recall_H$		$Precision_S$		$Precision_H$		G-Mean	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
GA-kNN_{Roul}	0.887	0.021	0.949	0.013	**0.921**	0.013	**0.542**	0.077	0.653	0.083	**0.715**	0.055
GA-kNN_{Tour}	**0.889**	0.019	**0.952**	0.009	0.920	0.014	0.535	0.088	**0.664**	0.073	0.712	0.062
GA-kNN_{Rand}	0.880	0.025	0.945	0.013	0.917	0.017	0.516	0.101	0.624	0.097	0.695	0.075
GA-RWN_{Roul}	**0.968**	0.003	**0.990**	0.002	**0.973**	0.002	**0.840**	0.013	**0.934**	0.010	**0.912**	0.007
GA-RWN_{Tour}	0.967	0.002	**0.990**	0.002	0.972	0.002	0.834	0.011	0.933	0.011	0.908	0.006
GA-RWN_{Rand}	0.967	0.002	**0.990**	0.002	0.972	0.002	0.834	0.014	0.933	0.010	0.908	0.008

Table 7
Comparison of the classification results for evolutionary 3-NN and RWN-based strategies for Lingspam dataset.

Algorithm	Accuracy		$Recall_S$		$Recall_H$		$Precision_S$		$Precision_H$		G-Mean	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
GA-kNN_{Roul}	0.870	0.023	0.934	0.012	0.913	0.017	0.550	0.092	0.621	0.077	0.714	0.063
GA-kNN_{Tour}	0.869	0.027	0.935	0.013	0.910	0.020	0.536	0.107	0.617	0.086	0.705	0.073
GA-kNN_{Rand}	**0.876**	0.027	**0.938**	0.015	**0.916**	0.019	**0.564**	0.099	**0.642**	0.090	**0.725**	0.068
GA-RWN_{Roul}	0.929	0.015	**0.730**	0.095	0.832	0.073	0.968	0.021	**0.948**	0.017	0.839	0.052
GA-RWN_{Tour}	0.926	0.016	0.718	0.094	0.833	0.082	0.968	0.027	0.946	0.016	0.831	0.048
GA-RWN_{Rand}	**0.933**	0.010	0.728	0.090	**0.855**	0.060	**0.974**	0.015	**0.948**	0.016	**0.840**	0.051

GA. Based on the RWN-based results, we can see that all accuracies are boosted compared to the results of 1-NN classifier, and the GA with RWS theme has reached to the maximum classification accuracy level. By using RWN and GA with RWS, the accuracy level has boosted up by 8.1% compared to the matched case using 1-NN. However, the results of different selection schemes are almost similar and all rates are above 96.7%.

Table 7 shows the comparative classification results for Lingspam dataset when 3-NN and RWN-based approaches are employed as the base classifiers along with developed GA-based algorithms. With regard to the results of 3-NN in Table 7, we observe that the 3-NN joint with GA using random selection has reached the best accuracy rate compared to other selection techniques inside GA. According to the RWN-based results, we can see that all accuracies are higher than the outcomes of 3-NN classifier and the GA with random selection has obtained the best classification rate. Based on RWN with GA and random selection instead of using 3-NN base classifier, the accuracy rate has enhanced up to 5.7%. It is seen that the results with different selection

Table 8
Comparison of the classification results for evolutionary 1-NN and RWN-based strategies for CSDMC2010 dataset.

Algorithm	Accuracy		$Recall_S$		$Recall_H$		$Precision_S$		$Precision_H$		G-Mean	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
GA-kNN$_{Roul}$	0.817	0.051	0.875	0.041	0.857	0.036	0.694	0.076	0.727	0.085	0.779	0.060
GA-kNN$_{Tour}$	**0.827**	0.037	**0.886**	0.029	**0.862**	0.028	**0.701**	0.061	**0.747**	0.062	**0.788**	0.045
GA-kNN$_{Rand}$	0.795	0.041	0.858	0.032	0.842	0.030	0.663	0.066	0.690	0.066	0.753	0.050
GA-RWN$_{Roul}$	0.906	0.025	0.801	0.086	0.905	0.056	0.957	0.032	0.911	0.032	0.873	0.045
GA-RWN$_{Tour}$	0.909	0.020	**0.829**	0.050	0.888	0.060	0.947	0.035	**0.921**	0.020	**0.885**	0.023
GA-RWN$_{Rand}$	**0.911**	0.017	0.803	0.054	**0.913**	0.038	**0.962**	0.019	0.912	0.020	0.878	0.028

Table 9
Accuracy results after automatic selection of neurons using Auto-GA-RWN with different selection mechanisms for all datasets.

Datasets	α	0.999		0.995		0.99		0.95	
	β	0.001		0.005		0.01		0.05	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std
SpamAssassin	Auto-GA-RWN$_{Roul}$	0.966	0.003	0.966	0.003	0.965	0.002	0.959	0.004
	Auto-GA-RWN$_{Tour}$	0.965	0.003	0.965	0.002	0.964	0.002	0.958	0.004
	Auto-GA-RWN$_{Rnd}$	**0.967**	0.002	0.966	0.002	0.966	0.003	0.960	0.003
Lingspam	Auto-GA-RWN$_{Roul}$	**0.930**	0.014	0.921	0.017	0.924	0.023	0.920	0.024
	Auto-GA-RWN$_{Tour}$	0.922	0.022	0.912	0.033	0.910	0.058	0.916	0.033
	Auto-GA-RWN$_{Rnd}$	0.924	0.028	0.920	0.020	0.914	0.053	0.911	0.084
CSDMC2010	Auto-GA-RWN$_{Roul}$	0.906	0.030	0.905	0.018	0.902	0.021	0.900	0.004
	Auto-GA-RWN$_{Tour}$	**0.908**	0.018	0.898	0.023	0.897	0.019	0.904	0.004
	Auto-GA-RWN$_{Rnd}$	0.905	0.018	0.903	0.024	0.905	0.017	0.903	0.003

Table 10
Comparison of obtained average number of features using Auto-GA-RWN with different selection mechanisms for all datasets.

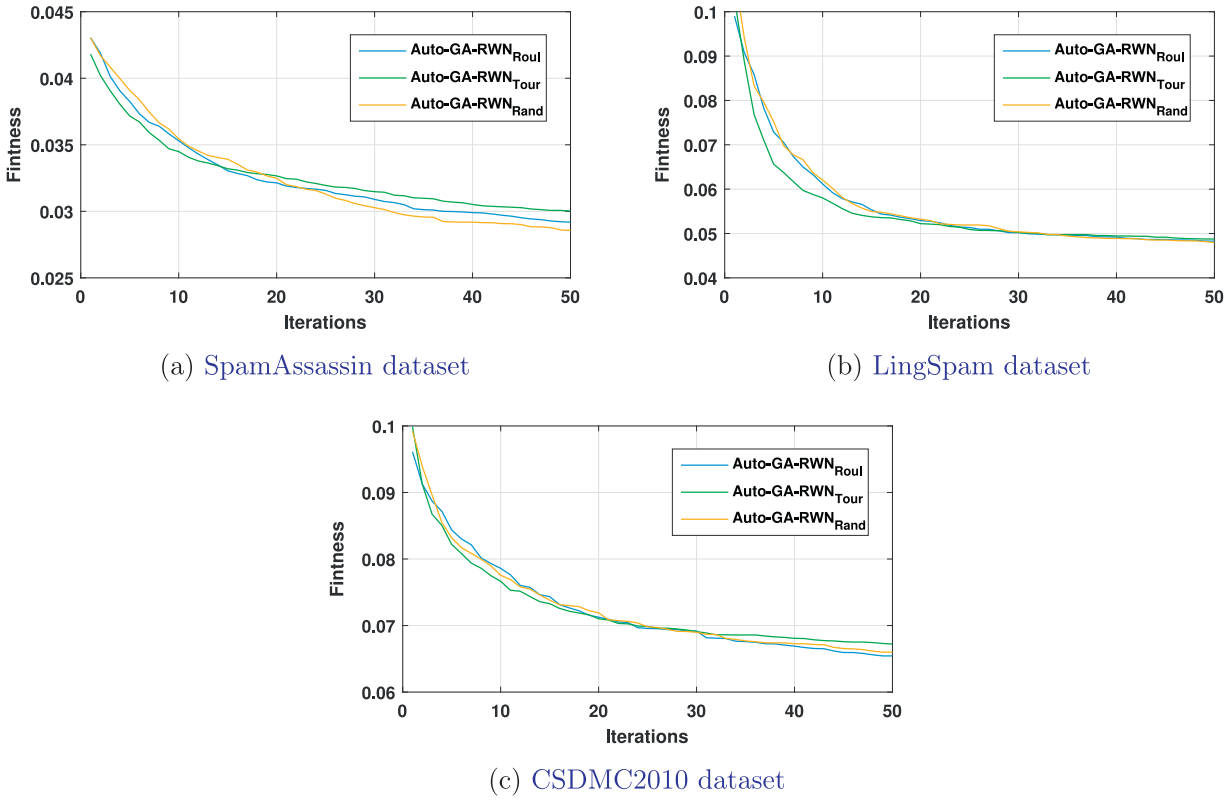
Datasets	α	0.999		0.995		0.99		0.95	
	β	0.001		0.005		0.01		0.05	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std
SpamAssassin	Auto-GA-RWN$_{Roul}$	72.400	5.817	70.867	6.447	68.800	6.272	46.167	7.047
	Auto-GA-RWN$_{Tour}$	74.567	5.263	72.633	5.933	69.900	7.160	50.900	5.996
	Auto-GA-RWN$_{Rnd}$	73.733	5.717	72.700	5.754	69.133	5.198	**45.333**	5.135
Lingspam	Auto-GA-RWN$_{Roul}$	69.400	4.658	69.067	6.741	70.267	5.783	65.033	5.846
	Auto-GA-RWN$_{Tour}$	70.267	5.595	71.433	6.021	69.933	6.777	68.633	7.690
	Auto-GA-RWN$_{Rnd}$	69.000	7.216	70.433	4.869	69.933	7.114	**60.700**	7.530
CSDMC2010	Auto-GA-RWN$_{Roul}$	75.200	5.950	74.367	5.048	73.267	5.681	64.967	7.047
	Auto-GA-RWN$_{Tour}$	73.800	6.354	70.867	4.842	71.867	6.367	66.133	5.996
	Auto-GA-RWN$_{Rnd}$	74.733	5.747	72.633	6.049	72.933	6.680	**63.833**	5.135

Table 11
Average number of neurons using Auto-GA-RWN with different selection mechanisms for all datasets.

Datasets	α	0.999		0.995		0.99		0.95	
	β	0.001		0.005		0.01		0.05	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std
SpamAssassin	Auto-GA-RWN$_{Roul}$	608.167	97.251	551.667	70.678	469.800	72.574	**204.833**	39.064
	Auto-GA-RWN$_{Tour}$	638.433	100.192	551.867	103.920	473.833	78.402	224.633	47.335
	Auto-GA-RWN$_{Rnd}$	621.933	102.355	559.533	83.594	474.967	84.851	212.367	46.585
Lingspam	Auto-GA-RWN$_{Roul}$	63.367	28.731	69.467	27.936	57.367	21.216	**48.800**	27.597
	Auto-GA-RWN$_{Tour}$	60.300	22.058	66.167	29.782	60.500	25.955	52.933	24.029
	Auto-GA-RWN$_{Rnd}$	63.300	24.778	59.800	17.841	65.467	28.551	50.600	16.569
CSDMC2010	Auto-GA-RWN$_{Roul}$	275.400	82.048	276.833	72.161	247.033	74.571	150.367	39.064
	Auto-GA-RWN$_{Tour}$	255.267	64.541	236.767	70.320	227.133	74.298	159.000	47.335
	Auto-GA-RWN$_{Rnd}$	259.533	68.273	245.533	63.479	263.200	97.176	**146.733**	46.585

techniques are almost similar and all accuracy results are higher than 92.6%.

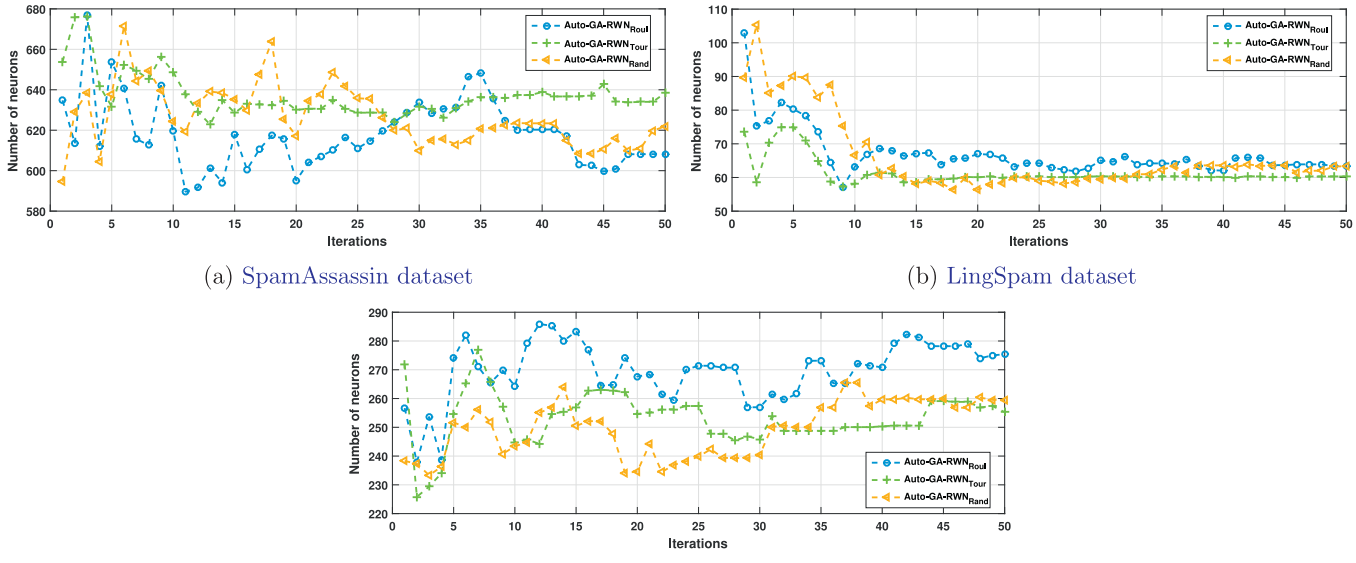
Table 8 reflects the comparative classification results for CSDMC2010 dataset when 1-NN and RWN-based techniques are used as the base classifiers along with the GA-based algorithms. Regarding the RWN-based results, we can see that all accuracies are enhanced compared to those of 1-NN classifier. Additionally, the GA with random selection has found the features with the best classification rate. By using RWN and GA with random selection, the accuracy rate has enhanced up to 11.6% compared to the similar case using 1-NN. It is seen that all accuracy results for RWN-based method are higher than 90.6%, while the k-NN-based classification results cannot surpass 82.7%.



(a) SpamAssassin dataset (b) LingSpam dataset

(c) CSDMC2010 dataset

Fig. 10. Average convergence curve of Auto-GA-RWN with different selection mechanisms based on different datasets.



(a) SpamAssassin dataset (b) LingSpam dataset

(c) CSDMC2010 dataset

Fig. 11. Average number of neurons over the course of iterations in Auto-GA-RWN based on different datasets.

5.3. Experiment III: evaluation results of Auto-GA-RWN

In this section, the proposed Auto-GA-RWN model with different selection mechanisms is applied and experimented for FS, classification and tuning the number of neurons, simultaneously. At first, we study the impact of α, β and γ of the fitness function presented in Eq. (6). Many researchers in the literature set α and β to 0.99 and 0.01, respectively [38,40,55,56]. In this work, we experiment small variations of these values as shown in Table 9. Note that each of these parameters controls the weight of its term. For our application, the most important term is the accuracy of the detection model, while the complexity and reduction of the number of features come in the second priority. Therefore, we experiment α with higher values, whereas β and γ are set to smaller equal values (i.e. $\beta = \gamma$).

Table 9 reports the accuracy results for Auto-GA-RWN with different selection mechanisms at different values for α, β, and γ.

As per results in Table 9, when α is equal to 0.999, it can be observed that the impact of different selection schemes on the resulted accuracy rates for each dataset is almost similar, which means that all Auto-GA-RWN versions has shown an almost similar performance, but

Table 12
Comparison between Auto-GA-RWN and GA-*k*NN.

Datasets	Algorithm	Accuracy		Recall 1		Recall 2		Precision 1		Precision 2		G-Mean	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
SpamAssassin	GA-*k*NN$_{Tour}$	0.889	0.019	0.952	0.009	0.920	0.014	0.535	0.088	0.664	0.073	0.712	0.062
SpamAssassin	Auto-GA-RWN$_{Rnd}$	**0.967**	0.002	**0.990**	0.001	**0.972**	0.002	**0.832**	0.013	**0.933**	0.008	**0.908**	0.007
LingSpam	GA-*k*NN$_{Rand}$	0.876	0.027	**0.938**	0.015	**0.916**	0.019	0.564	0.099	0.642	0.090	0.725	0.068
LingSpam	Auto-GA-RWN$_{Roul}$	**0.930**	0.014	0.710	0.116	0.851	0.058	**0.973**	0.015	**0.945**	0.020	**0.828**	0.07
CSDMC2010	GA-*k*NN$_{Tour}$	0.827	0.037	**0.886**	0.029	0.862	0.028	0.701	0.061	0.747	0.062	0.788	0.045
CSDMC2010	Auto-GA-RWN$_{Tour}$	**0.908**	0.018	0.825	0.037	**0.887**	0.053	**0.948**	0.029	**0.919**	0.015	**0.884**	0.019

Table 13
p-values of the Wilcoxon test based on G-mean values of Auto-GA-RWN versus GA-*k*NN for all datasets ($p \geq 0.05$ are underlined).

SpamAssassin	LingSpam	CSDMC2010
3.02E−11	1.38E−06	8.15E−11

relatively, the algorithm with RS has attained the top average rate. For Lingspam dataset, Auto-GA-RWN with RWS has reached to best accuracy results, and Auto-GA-RWN with TS has attained the best rate for the third dataset. By adjusting the values of α and β, no significant gap has happened between the average results for each dataset.

Table 10 shows the calculated number of features when we utilized automatic selection of neurons. From the results in Table 10, we can see that different GA-based algorithms have shown a similar efficacy in discovering the optimal number of features for each dataset. As β increases, number of features decreases for the first dataset. A similar pattern can also be seen for GA with RWS and TS algorithms in dealing with the third dataset.

Table 11 reveals the attained number of neurons using different GA-based algorithms.

The convergence curves of the Auto-GA-RWN with different selection mechanisms are shown in Fig. 10 for SpamAssassin, Lignspam, and CSDMC2010 datasets, respectively.

As per curves in Fig. 10(a), it is observed that the different selection schemes show a similar convergence pattern on SpamAssassin dataset. However, the Auto-GA-RWN with random selection strategy is relatively preferable in terms of convergence rate to some extent. Based on behaviors monitored in Fig. 10(b) and (c), it is detected that all variants show a similar monotone decreasing pattern in dealing with Lignspam and CSDMC2010 datasets.

Fig. 11 shows the time-varying patterns in the average number of neurons over the course of iterations for different variants of Auto-GA-RWN on studied datasets. It is observed that at the beginning of the iterations there is more fluctuation in the number of hidden neurons, while at the end, the search becomes more stable around the number that maximizes the accuracy rate. Tuning the number of hidden neurons is one of the most important characteristics of the proposed Auto-GA-RWN algorithm.

Comparing the results of Auto-GA-RWN to GA-*k*NN, which is one of the most commonly applied wrapper-based methods in the literature, Table 12 lists the results of both approaches based on SpamAssassin, Lingspam and CSDMC2010 datasets. It can be seen that the new Auto-GA-RWN approach outperforms GA-*k*NN in most of evaluation measures. Moreover, Auto-GA-RWN shows higher stability in terms of lower standard deviation values. To support these findings, the non-parametric Wilcoxon ranksum statistical test is performed at 5% significance level for Auto-GA-RWN against GA-*k*NN. The test is calculated based on the G-mean values of each algorithm. Table 13 shows the obtained p-values of the test. The results of the test show that there is a significant difference between the results of Auto-GA-RWN and GA-*k*NN approaches.

5.4. Analysis of the most relevant features

In this section, we describe in detail the top 25 selected features by Auto-GA-RWN for every dataset, and then, we compare them with the

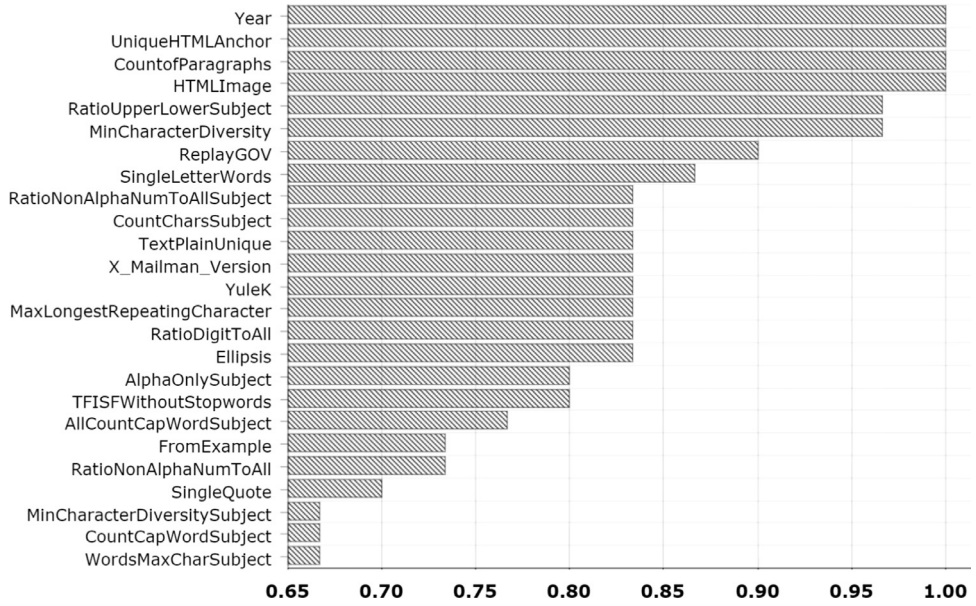


Fig. 12. Top 25 features (SpamAssassin dataset).

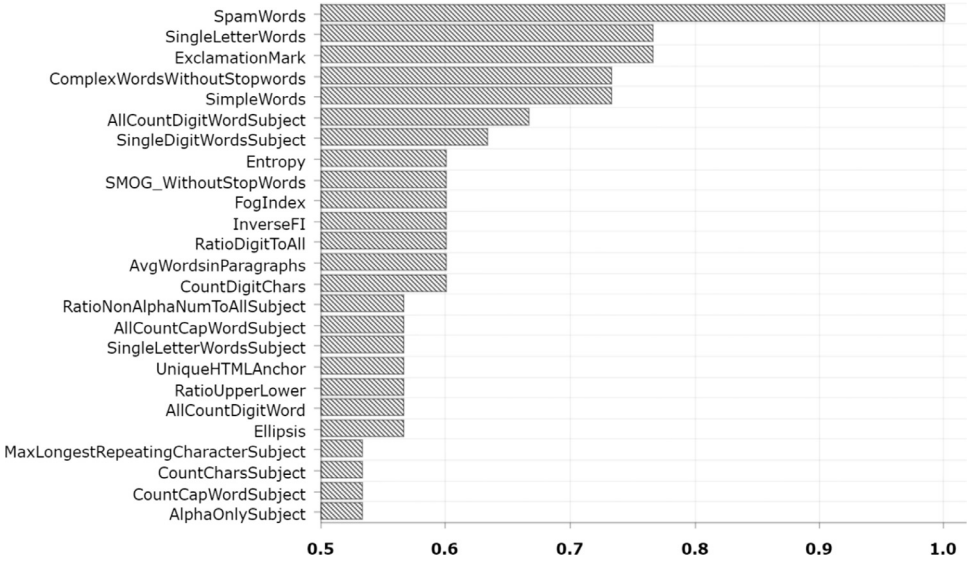


Fig. 13. Top 25 features (Lingspam dataset).

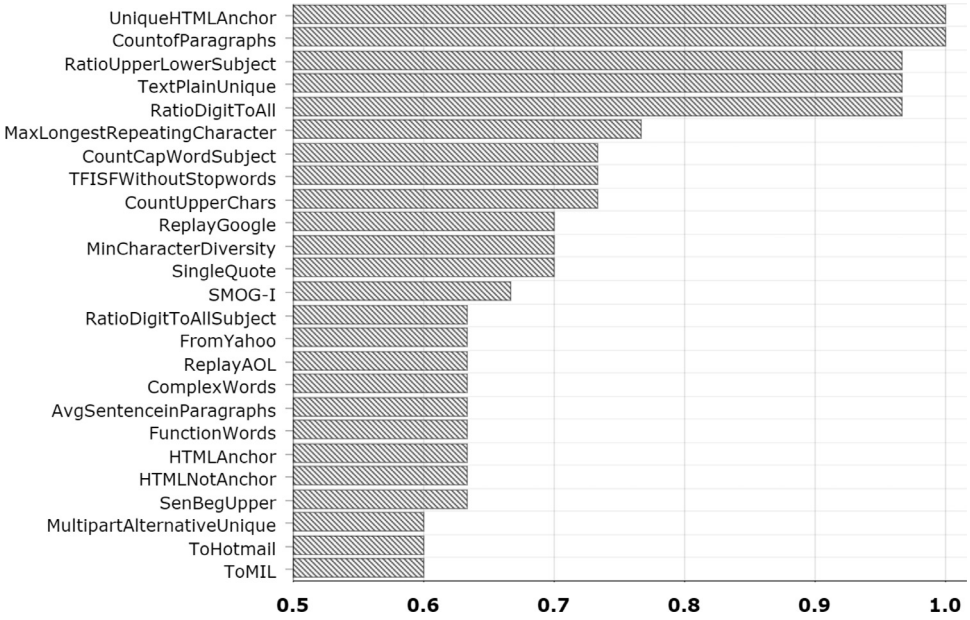


Fig. 14. Top 25 features (CSDMC2010 dataset).

top 25 features selected by Information Gain and ReliefF filters. Figs. 12–14 depict the average weights of these features for all iterations, from most to least recurring, for SpamAssassin, Lingspam and CSDMC2010 datasets, respectively.

For Auto-GA-RWN, as illustrated in Fig. 12, the features selected in all iterations for SpamAssassin dataset mostly belong to payload-body features, which are UniqueHTMLAnchor, CountOfParagraphs and HTMLImage, with only one header feature, i.e., Year. It is also noticed from the results that most of the other top features belong to the header part where, for example, RatioUpperLowerSubject, MinCharacterDiversity, ReplayGOV and SingleLetterWords are all selected in more than 85% of the iterations, keeping in mind that SingleLetterWords is a mutual feature for both header and body categories. Moreover, most of the features selected between 80% and 85% of the time are header features as well, while few of them belong to body subcategory, and only one lexical diversity feature, Yulek, which is a payload feature.

For Lingspam results in Fig. 13, it is seen that nearly a different set of top features is selected, given that only one feature, SpamWords, which falls under body subcategory, is selected in all iterations. Furthermore, there is a noticeable difference in the recurrences between SpamWords and the rest of the features, starting for the other features from around 75% of the iterations down to almost 55%. All of these features belong to subject and body subcategories as the original corpus of this dataset does not has the other parts of the email.

Close number of features for both header and payload categories is found as well for the CSDMC2010 dataset in Fig. 14; as one of the top 5 selected features repeated in over 95% of the iterations is mutual between header and payload categories, and 2 belong to the body-payload subcategory, which are UniqueHTMLAnchor and CountOfParagraphs. While the other 2 features, RatioUpperLowerSubject and TextPlainUnique, are under the header category. On the other hand, the rest of the features, falling between 50% and 80% of recurrences, tend to have slightly higher number towards the payload category; as there are 10 body features, 2 readability features which are SMOGE-I and ComplexWords, and 10 header features.

Table 14
Top 25 features obtained by the proposed algorithm versus filter-based feature selection on SpamAssassin dataset.

	Auto-GA-RWN	Information gain	ReliefF
1	**Year**	Month	X_Mailman_Version
2	**UniqueHTMLAnchor**	UniqueHTMLAnchor	RatioNonAlphaNumToAllSubject
3	CountofParagraphs	HTMLAnchor	Month
4	HTMLImage	HTMLTags	RatioUpperToAllSubject
5	RatioUpperLowerSubject	HTMLNotAnchor	Day
6	**MinCharacterDiversity**	RatioNonAlphaNumToAll	Minute
7	ReplayGOV	LongestCapital	FromExample
8	SingleLetterWords	ExclamationMark	Hour
9	**RatioNonAlphaNumToAllSubject**	X_Mailman_Version	RatioDigitToAll
10	**CountCharsSubject**	MinCharacterDiversity	MinCharacterDiversity
11	**TextPlainUnique**	MaxLongestRepeatingCharacter	ToExample
12	**X_Mailman_Version**	CountUpperChars	RatioNonAlphaNumToAll
13	YuleK	CountDigitChars	TextPlainUnique
14	**MaxLongestRepeatingCharacter**	AvgSentenceinParagraphs	RatioDigitToAllSubject
15	**RatioDigitToAll**	AvgCharinParagraphs	CountCapWordSubject
16	Ellipsis	AvgWordsinParagraphs	CountCharsSubject
17	AlphaOnlySubject	TextPlainUnique	Second
18	TFISFWithoutStopwords	SpamWords	MaxLongestRepeatingCharacter
19	AllCountCapWordSubject	RatioUpperLower	VocabularyRichness
20	**FromExample**	SpecialChars	Year
21	**RatioNonAlphaNumToAll**	AlphaNumeric	RatioUpperToAll
22	SingleQuote	FISimpleWithoutStopwords	MaxLongestRepeatingCharacterSubject
23	MinCharacterDiversitySubject	CountChars	AlphaOnlySubject
24	**CountCapWordSubject**	SingleDigitWords	WordsMaxCharSubject
25	**WordsMaxCharSubject**	AllCountDigitWord	LongestCapital

*Features in **bold** were identified by either IG or ReliefF.

Table 15
Top 25 features obtained by the proposed algorithm versus filter-based feature selection on Lingspam dataset.

	Auto-GA-RWN	Information gain	ReliefF
1	**SpamWords**	ExclamationMark	RatioDigitToAll
2	**SingleLetterWords**	FRESWithoutStopwords	SpamWords
3	**ExclamationMark**	FKRI_WithoutStopwords	FRESWithoutStopwords
4	**ComplexWordsWithoutStopwords**	InverseFI_WithoutStopwords	FRES
5	SimpleWords	FogIndexWithoutStopWords	ARI
6	AllCountDigitWordSubject	InverseFI	FogIndexWithoutStopWords
7	SingleDigitWordsSubject	FRES	FKRI_WithoutStopwords
8	**Entropy**	FogIndex	RatioNonAlphaNumToAll
9	**SMOG_WithoutStopWords**	WordLengthWithoutStopwords	WordsAVGChar
10	**FogIndex**	WordLength	FogIndex
11	**InverseFI**	SpamWords	SMOG
12	**RatioDigitToAll**	FKRI	SMOG-I
13	AvgWordsinParagraphs	SMOG_WithoutStopWords	SMOG_WithoutStopWords
14	CountDigitChars	SMOG-I	WordLength
15	RatioNonAlphaNumToAllSubject	SMOG	WordLengthWithoutStopwords
16	AllCountCapWordSubject	WordsAVGChar	FKRI
17	SingleLetterWordsSubject	FISimpleWithoutStopwords	ComplexWordsWithoutStopwords
18	UniqueHTMLAnchor	Dollar	ComplexWords
19	RatioUpperLower	ARI	FISimple
20	**AllCountDigitWord**	FISimple	VocabularyRichness
21	Ellipsis	WordsMaxChar	CountWordLonger6Char
22	MaxLongestRepeatingCharacterSubject	VocabularyRichness	FISimpleWithoutStopwords
23	CountCharsSubject	MaxLongestRepeatingCharacter	Entropy
24	CountCapWordSubject	RatioNonAlphaNumToAll	AllCountDigitWord
25	AlphaOnlySubject	LongestCapital	ExclamationMark

*Features in **bold** were identified by either IG or ReliefF.

From a general point of view, we can clearly observe that most of important features are categorized into payload-body and header features, while the few rest of them belong to payload-readability and payload-lexical diversity subcategories. It is also generally observed that the features are less likely to be repeated for all 3 datasets. Both observations combined would lead to understand the behavior of spam emails; as spammers tend to mostly target the users using both header and body parts of the email either by sending misleading subjects and false sender addresses, or by sending deceptive and malicious email contents. However, spammers may differ in their methods and techniques in how to exploit both parts.

Now, to contrast the goodness of the feature relevance discovered by Auto-GA-RWN, the top 25 features identified by Auto-GA-RWN are compared with those identified by two of the most commonly known filter-based feature selection methods, which are Information Gain (IG) and ReliefF. The features identified by each method are listed for the three datasets in Table 14, 15 and 16. As it can be seen in the tables, Auto-GA-RWN has at least 10 features in common with IG and ReliefF in each case. This reflects the potential of Auto-GA-RWN in identifying the relevant features in comparison with well-regarded methods like IG and ReliefF.

Table 16
Top 25 features obtained by the proposed RWN-based method versus filter-based feature selection on CSDMC2010 dataset.

	Auto-GA-RWN	Information gain	ReliefF
1	**UniqueHTMLAnchor**	HTMLTags	TextPlainUnique
2	**CountofParagraphs**	HTMLAnchor	Month
3	**RatioUpperLowerSubject**	RatioNonAlphaNumToAll	RatioUpperToAllSubject
4	**TextPlainUnique**	UniqueHTMLAnchor	RatioNonAlphaNumToAll
5	**RatioDigitToAll**	HTMLNotAnchor	MinCharacterDiversity
6	**MaxLongestRepeatingCharacter**	LongestCapital	ToExample
7	**CountCapWordSubject**	AvgCharinParagraphs	RatioNonAlphaNumToAllSubject
8	TFISFWithoutStopwords	HTMLImage	RatioUpperLowerSubject
9	CountUpperChars	AvgWordsinParagraphs	RatioDigitToAllSubject
10	ReplayGoogle	MinCharacterDiversity	FromExample
11	**MinCharacterDiversity**	TextPlainUnique	RatioDigitToAll
12	SingleQuote	FKRI	Year
13	SMOG-I	FISimpleWithoutStopwords	MaxLongestRepeatingCharacter
14	**RatioDigitToAllSubject**	MaxLongestRepeatingCharacter	VocabularyRichness
15	FromYahoo	AvgSentenceinParagraphs	LongestCapital
16	ReplayAOL	FKRI_WithoutStopwords	UniqueContentTypesAttachment
17	ComplexWords	FRES	CountCapWordSubject
18	**AvgSentenceinParagraphs**	FogIndexWithoutStopWords	RatioUpperToAll
19	FunctionWords	ExclamationMark	Day
20	**HTMLAnchor**	FISimple	Hour
21	**HTMLNotAnchor**	InverseFI_WithoutStopwords	CountCharsSubject
22	SenBegUpper	CountofParagraphs	Sichel
23	MultipartAlternativeUnique	FRESWithoutStopwords	Minute
24	ToHotmail	FogIndex	CountOfAttachments
25	ToMIL	InverseFI	SingleCharWordsSubject

*Features in **bold** were identified by either IG or ReliefF.

6. Conclusion and future directions

In this paper, an intelligent spam detection system (called Auto-GA-RWN), was proposed based on hybridizing Genetic Algorithm and Random Weight neural networks. This system performs three simultaneous tasks which are feature selection, automatic tuning of the hidden neurons in the network, and evolving the spam detection model. The proposed Auto-GA-RWN was evaluated based on three publicly available corpora. The obtained evaluation results reveal that Auto-GA-RWN can hit very promising detection rates while, at the same time, it identifies the most relevant features and optimizes the configuration of its core classifier, which is the Randomized Weight Network. After further analysis, considering the relevant features, the results reveal that most of these features are categorized into payload-body and header features, while the few of them belong to payload-readability and payload-lexical diversity subcategories. It is also noticed that the relevant features change from a dataset to another. Overall, it can be concluded that spammers tend to mostly target the users using both header and body parts of the email, either by sending misleading subjects and false sender addresses or by sending deceptive and malicious email contents. However, spammers keep coming up with the methods and techniques that urge for an evolvable and adaptable spam detection system. For future works, other solutions for the imbalanced classification tasks can be studied (i.e. cost-sensitive learning and/or preprocessing). In addition, the impact of these solutions on the relevance of input features can be investigated.

Appendix A

Table 17
The header features.

ID	Feature details	Type	Ref.	ID	Feature details	Type	Ref.
1	Year	Metadata	[25]	26	Replay to MIL?	Metadata	[25]
2	Month	Metadata	[25]	27	Replay to Yahoo?	Metadata	[25]
3	Day	Metadata	[25,57]	28	Replay to AOL?	Metadata	[25]
4	Hour	Metadata	[25,57]	29	Replay to Gov?	Metadata	[25]
5	Minute	Metadata	[25,57]	30	X-Mailman-Version	Metadata	[25]
6	Second	Metadata	[25,57]	31	Exist Text/Plain?	Metadata	[25]
7	From Google?	Metadata	[25]	32	Exist Multipart/Mixed?	Metadata	[25]
8	From AOL?	Metadata	[25]	33	Exist Multipart/Alternative?	Metadata	[25]
9	From Gov?	Metadata	[25]	34	No. of characters.	Subject	[57]
10	From HTML?	Metadata	[25]	35	No. of capitalised words.	Subject	[57]
11	From MIL?	Metadata	[25]	36	No. of words in all uppercase.	Subject	[57]
12	From Yahoo?	Metadata	[25]	37	No. of words that are digits.	Subject	[57]
13	From Example?	Metadata	[25]	38	No. of words containing only letters.	Subject	[57]
14	To Hotmail?	Metadata	[25]	39	No. of words containing letters and numbers.	Subject	[57]
15	To Yahoo?	Metadata	[25]	40	No. of words that are single letters.	Subject	[57]
16	To Example?	Metadata	[25]	41	No. of words that are single digits.	Subject	[57]
17	To MSN?	Metadata	[25]	42	No. of words that are single characters.	Subject	[57]
18	To Localhost?	Metadata	[25]	43	Max ratio of uppercase to lowercase letters of each word	Subject	[57]
19	To Google?	Metadata	[25]	44	Min character diversity of each word.	Subject	[57]

(*continued on next page*)

Table 17 (*continued*)

ID	Feature details	Type	Ref.	ID	Feature details	Type	Ref.
20	To AOL?	Metadata	[25]	45	Max ratio of uppercase letters to all characters of each word.	Subject	[57]
21	To Gov?	Metadata	[25]	46	Max ratio of digit characters to all characters of each word.	Subject	[57]
22	To MIL?	Metadata	[25]	47	Max ratio of non-alphanumerics to all characters of each word	Subject	[57]
23	Count of "To" Email	Metadata	[57]	48	Max of the longest repeated character.	Subject	[57]
24	Replay to Google?	Metadata	[25]	49	Max of the character lengths of words.	Subject	[57]
25	Replay to Hotmail?	Metadata	[25]	–	–	–	–

Table 18
Email body features.

ID	Feature details	Ref.	ID	Feature details	Studies
1	Count of Spam Words	[25,26,58]	31	Number of question marks	[25,26]
2	Count of Function Words	[26,58]	32	No. of multiple question marks	[25]
3	Count of HTML Anchor	[25,26,57,58]	33	No. of exclamation marks	[25,26]
4	Count of Unique HTML Anchor	[25,57]	34	No. of multiple exclamation marks	[25]
5	Count of HTML Not Anchor	[58]	35	No. of colons	[25,26]
6	Count of HTML Image	[26]	36	No. of ellipsis	[25,26]
7	Count of HTML All Tags	[26,58]	37	Total No. of sentences	[25,26,58]
8	Count of Alpha-numeric Words	[26,57,58]	38	Total No. of paragraphs	[25]
9	TF-ISF	[58]	39	Average No. of sentences per paragraph	[25]
10	TF•ISF without stopwords	[58]	40	Average number of words pre paragraph	[25]
11	Count of duplicate words.	[26]	41	Average No. of character per paragraph	[25]
12	Minimum word length	[26]	42	Average No. of word per sentences	[25]
13	Count of lowercase letters	[26]	43	No. of sentence begin with upper case	[25]
14	Longest sequence of adjacent capital letters	[25,26]	44	No. of sentence begin with lower case	[25]
15	Count of lines	[25,26]	45	Character frequency "$"	[25,26]
16	Total No. of digit character	[25]	46	No. of capitalized words.	[57]
17	Total No. of white space	[25]	47	No. of words in all uppercase.	[57]
18	Total No. of upper case character	[25,26]	48	Number of words that are digits.	[57]
19	Total No. of characters	[25,57]	49	No. of words containing only letters.	[57]
20	Total No. of tabs	[25]	50	No. of words that are single letters.	[57]
21	Total No. of special characters	[25]	51	No. of words that are single digits.	[57]
22	Total number of alpha characters	[25]	52	Number of words that are single characters.	[57]
23	Total No. of words	[25,26]	53	Max ratio of uppercase letters to lowercase letters of each word.	[57]
24	Average word length	[25,26]	54	Min of character diversity of each word.	[57]
25	Words longer than 6 characters	[25]	55	Max ratio of uppercase letters to all characters of each word.	[57]
26	Total No. of words (1–3 Characters)	[25]	56	Max ratio of digit characters to all characters of each word.	[57]
27	No. of single quotes	[25,26]	57	Max ratio of non-alphanumerics to all characters of each word.	[57]
28	No. of commas	[25,26]	58	Max of the longest repeating character.	[57]
29	No. of periods	[25,26]	59	Max of the character lengths of words.	[26,57]
30	No. of semi-colons	[25,26]	–	–	–

Table 19
Readability features.

ID	Feature details
1	Number of simple words features (with and without stopwords)
2	Number of complex words features (with and without stopwords)
3	Word length features (with and without stopwords)
4	Fog Index (FI) features (with and without stopwords)
5	Flesch Reading Ease Score (FRES) features (with and without stopwords)
6	SMOG index features (with and without stopwords)
7	Flesch-Kincaid Readability Index (FKRI) features (with and without stopwords)
8	FORCAST index features (with and without stopwords)
9	Simple Word FI features (with and without stopwords)
10	Inverse FI features (with and without stopwords)
11	SMOG-I feature
12	Automated Readability Index (ARI)
13	Coleman-Liau Index (CLI)

Table 20
Lexical diversity features and attachment features.

ID	Feature details
	Lexical diversity features
1	Vocabulary Richness
2	Hapax legomena or V(1,N)
3	Hapax dislegomena or V(2,N)
4	Entropy measure
5	YuleK
6	SichelS
7	Honore
	Attachment Features
1	Number of all attachment files in an email
2	Number of unique content types of attachment files in an email

References

[1] A.M. Al-Zoubi, H. Faris, M.A. Hassonah, et al., Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts, Knowl. Based Syst. 153 (2018) 91–104.
[2] A.H. Mohammad, R.A. Zitar, Application of genetic optimized artificial immune system and neural networks in spam detection, Appl. Soft Comput. 11 (4) (2011) 3827–3845.
[3] S. Günal, S. Ergin, M.B. Gülmezoğlu, Ö.N. Gerek, On feature extraction for spam e-mail detection, International Workshop on Multimedia Content Representation, Classification and Security, Springer, 2006, pp. 635–642.
[4] T.S. Guzella, W.M. Caminhas, A review of machine learning approaches to spam filtering, Expert Syst. Appl. 36 (7) (2009) 10206–10222.
[5] E. Blanzieri, A. Bryl, A survey of learning-based techniques of email spam filtering, Artif. Intell. Rev. 29 (1) (2008) 63–92.
[6] H. Drucker, D. Wu, V.N. Vapnik, Support vector machines for spam categorization, IEEE Trans. Neural Netw. 10 (5) (1999) 1048–1054.
[7] O. Amayri, N. Bouguila, A study of spam filtering using support vector machines, Artif. Intell. Rev. 34 (1) (2010) 73–108.
[8] V. Metsis, I. Androutsopoulos, G. Paliouras, Spam filtering with naive bayes – which naive bayes? CEAS, 17 (2006), pp. 28–69.
[9] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, A bayesian approach to filtering junk e-mail, Learning for Text Categorization: Papers from the 1998 workshop, 62 (1998), pp. 98–105.
[10] D. Puniškis, R. Laurutis, R. Dirmeikis, An artificial neural nets for spam e-mail recognition, Elektronika ir Elektrotechnika 69 (5) (2006) 73–76.
[11] Z. Chuan, L. Xianliang, H. Mengshu, Z. Xu, A lvq-based neural network anti-spam email approach, ACM SIGOPS Oper. Syst. Rev. 39 (1) (2005) 34–39.
[12] D. Trudgian, Spam classification using nearest neighbour techniques, Intelligent Data Engineering and Automated Learning–IDEAL 2004, (2004), pp. 578–585.
[13] S. Rajput, A. Arora, Designing spam model-classification analysis using decision trees, Int. J. Comput. Appl. 75 (10) (2013).
[14] A.S. Aski, N.K. Sourati, Proposed efficient algorithm to filter spam using machine learning techniques, Pac. Sci. Rev. A: Nat. Sci. Eng. 18 (2) (2016) 145–149.
[15] F. Toolan, J. Carthy, Feature selection for spam and phishing detection, eCrime Researchers Summit (eCrime), 2010, IEEE, 2010, pp. 1–12.
[16] I. Idris, A. Selamat, Improved email spam detection model with negative selection algorithm and particle swarm optimization, Appl. Soft Comput. 22 (2014) 11–27.
[17] I. Idris, A. Selamat, S. Omatu, Hybrid email spam detection model with negative selection algorithm and differential evolution, Eng. Appl. Artif. Intell. 28 (2014) 97–110.
[18] I. Idris, A. Selamat, N.T. Nguyen, S. Omatu, O. Krejcar, K. Kuca, M. Penhaker, A combined negative selection algorithm–particle swarm optimization for an email spam detection system, Eng. Appl. Artif. Intell. 39 (2015) 33–44.
[19] H. Faris, I. Aljarah, B. Al-Shboul, A hybrid approach based on particle swarm optimization and random forests for e-mail spam filtering, International Conference on Computational Collective Intelligence, Springer, 2016, pp. 498–508.
[20] G. Ruan, Y. Tan, A three-layer back-propagation neural network for spam detection using artificial immune concentration, Soft. Comput. 14 (2) (2010) 139–150.
[21] H. Faris, I. Aljarah, J. Alqatawna, Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection, Applied Electrical Engineering and Computing Technologies (AEECT), 2015 IEEE Jordan Conference on , vol., no., pp.1–5, IEEE, 2015, pp. 1–5.
[22] C.-H. Wu, Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks, Expert Syst. Appl. 36 (3) (2009) 4321–4330.
[23] Y. Zhang, S. Wang, P. Phillips, G. Ji, Binary pso with mutation operator for feature selection using decision tree applied to spam detection, Knowl. Based Syst. 64 (2014) 22–31.
[24] D. Gavrilis, I. Tsoulos, E. Dermatas, Neural recognition and genetic features selection for robust detection of e-mail spam, Adv. Artif. Intell. (2006) 498–501.
[25] J. Alqatawna, H. Faris, K. Jaradat, M. Al-Zewairi, A. Omar, Improving knowledge based spam detection methods: the effect of malicious related features in imbalance data distribution, Int. J. Commun. Netw. Syst. Sci. 8 (5) (2015) 118–129.
[26] B.A. Al-Shboul, H. Hakh, H. Faris, I. Aljarah, H. Alsawalqah, Voting-based classification for e-mail spam detection, J. ICT Res. Appl. 10 (1) (2016) 29–42.
[27] R.L. Kadri, F.F. Boctor, An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: the single mode case, Eur. J. Oper. Res. 265 (2) (2018) 454–462.
[28] A.A. Heidari, R.A. Abbaspour, A.R. Jordehi, An efficient chaotic water cycle algorithm for optimization tasks, Neural Comput. Appl. 28 (1) (2017) 57–85.
[29] A.A. Heidari, P. Pahlavani, An efficient modified grey wolf optimizer with lévy flight for optimization tasks, Appl. Soft Comput. 60 (2017) 115–134.
[30] M.M. Mafarja, I. Aljarah, A.A. Heidari, H. Faris, P. Fournier-Viger, X. Li, S. Mirjalili, Binary dragonfly optimization for feature selection using time-varying transfer functions, Knowl. Based Syst. (2018).
[31] A.E. Eiben, J. Smith, From evolutionary computation to the evolution of things, Nature 521 (7553) (2015) 476.
[32] H. Hallawi, J. Mehnen, H. He, Multi-capacity combinatorial ordering ga in application to cloud resources allocation and efficient virtual machines consolidation, Future Gener. Comput. Syst. 69 (2017) 1–10.
[33] D.S. Weile, E. Michielssen, Genetic algorithm optimization applied to electromagnetics: a review, IEEE Trans. Antennas Propag. 45 (3) (1997) 343–353.
[34] H. Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm i. continuous parameter optimization, Evol. Comput. 1 (1) (1993) 25–49.
[35] D.E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms, Foundations of Genetic Algorithms, 1 Elsevier, 1991, pp. 69–93.
[36] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inf. 16 (3) (2002) 193–203.
[37] M.M. Mafarja, S. Mirjalili, Hybrid whale optimization algorithm with simulated annealing for feature selection, Neurocomputing 260 (2017) 302–312.
[38] M. Mafarja, I. Aljarah, A.A. Heidari, A.I. Hammouri, H. Faris, A. Al-Zoubi, S. Mirjalili, Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems, Knowl. Based Syst. 145 (2018) 25–45, https://doi.org/10.1016/j.knosys.2017.12.037.
[39] H. Li, D. Yuan, X. Ma, D. Cui, L. Cao, Genetic algorithm for the optimization of features and neural networks in ecg signals classification, Sci. Rep. 7 (2017) 41011.
[40] H. Faris, M.M. Mafarja, A.A. Heidari, I. Aljarah, A.M. Al-Zoubi, S. Mirjalili, H. Fujita, An efficient binary salp swarm algorithm with crossover scheme for feature selection problems, Knowl. Based Syst. 154 (2018) 43–67, https://doi.org/10.1016/j.knosys.2018.05.009.
[41] A. Colombo, D.E. Galli, L. De Caro, F. Scattarella, E. Carlino, Facing the phase problem in coherent diffractive imaging via memetic algorithms, Sci. Rep. 7 (2017) 42236.
[42] M. Nazari-Heris, B. Mohammadi-Ivatloo, A. Haghrah, Optimal short-term generation scheduling of hydrothermal systems by implementation of real-coded genetic algorithm based on improved mühlenbein mutation, Energy 128 (2017) 77–85.
[43] C. Reeves, Genetic algorithms, Handbook of Metaheuristics, Springer, 2003, pp. 55–82.
[44] W.F. Schmidt, M.A. Kraaijveld, R.P. Duin, Feedforward neural networks with random weights, Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on, IEEE, 1992, pp. 1–4.
[45] W. Cao, X. Wang, Z. Ming, J. Gao, A review on neural networks with random weights, Neurocomputing 275 (2018) 278–287.
[46] Y.-H. Pao, G.-H. Park, D.J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing 6 (2) (1994) 163–180.
[47] X. Zhao, D. Li, B. Yang, S. Liu, Z. Pan, H. Chen, An efficient and effective automatic recognition system for online recognition of foreign fibers in cotton, IEEE Access 4 (2016) 8465–8475.
[48] B. Igelnik, Y.-H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Trans. Neural Netw. 6 (6) (1995) 1320–1329.
[49] J. Zhai, X. Wang, X. Pang, Voting-based instance selection from large data sets with mapreduce and random weight networks, Inf. Sci. 367 (2016) 1066–1077.
[50] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on, 2 IEEE, 2004, pp. 985–990.
[51] W. Hijawi, H. Faris, J. Alqatawna, I. Aljarah, A. Al-Zoubi, M. Habib, EMFET: E-mail

features extraction tool, arXiv preprint arXiv:1711.08521(2017).
[52] C.-F. Tsai, W. Eberle, C.-Y. Chu, Genetic algorithms in feature and instance selection, Knowl. Based Syst. 39 (2013) 240–247.
[53] E. Emary, H.M. Zawbaa, A.E. Hassanien, Binary ant lion approaches for feature selection, Neurocomputing 213 (2016) 54–65.
[54] O. Kramer, Genetic Algorithms, Springer International Publishing, Cham, pp. 11–19.
[55] E. Emary, H.M. Zawbaa, A.E. Hassanien, Binary ant lion approaches for feature selection, Neurocomputing 213 (2016) 54–65.
[56] E. Emary, H.M. Zawbaa, C. Grosan, Experienced gray wolf optimization through reinforcement learning and neural networks, IEEE Trans. Neural Netw. Learn Syst. 29 (3) (2018) 681–694.
[57] K.-N. Tran, M. Alazab, R. Broadhurst, et al., Towards a feature rich model for predicting spam emails containing malicious attachments and urls, Eleventh Australasian Data Mining Conference Canberra, ACT, 146 (2013).
[58] R. Shams, R.E. Mercer, Supervised classification of spam emails with natural language stylometry, Neural Comput. Appl. 27 (8) (2016) 2315–2331.
[59] H. Faris, A.-Z. Ala'M, I. Aljarah, et al., Improving email spam detection using content based feature engineering approach, 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), IEEE, 2017, pp. 1–6.