

# Application of Evolutionary Algorithms in Detecting SMS Spam at Access Layer

M Zubair Rafique, Nasser Alrayes and Muhammad Khurram Khan  
Center of Excellence in Information Assurance, CoEIA  
King Saud University  
Riyadh 11653, Saudi Arabia  
{zrafique.c,nalrayes,mkhurram}@ksu.edu.sa

## ABSTRACT

In recent years, Short Message Service (SMS) has been widely exploited in arbitrary advertising campaigns and the propagation of scam. In this paper, we first analyze the role of SMS spam as an increasing threat to mobile and smart phone users. Afterward, we present a filtering method for controlling SMS spam on the access layer of mobile devices. We analyze the role of different evolutionary and non-evolutionary classifiers for our spam filter by assimilating the byte-level features of SMS. We evaluated our framework on real-world benign and spam datasets collected from Grumbletext and the users in our social networking community. The results of carefully designed experiments demonstrated that the evolutionary classifiers, like the Structural Learning Algorithm in Vague Environment (SLAVE), could efficiently detect spam messages at the access layer of a mobile device. To the best of our knowledge, the current work is the first SMS spam filter based on evolutionary classifier that works on the access layer of a mobile device. The results of our experiments show that our framework, using evolutionary algorithms, achieves a detection accuracy of more than 93%, with false alarm rate of  $\leq 0.13\%$  in classifying spam SMS. Moreover, the memory requirement for incorporating SMS features is relatively small, and it takes less than one second to classify a message as spam or benign.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*information filtering*; H.3.4 [Information Storage and Retrieval]: Systems and Software—*performance evaluation (efficiency and effectiveness)*.

## General Terms

Algorithms, Experimentation, Security

## Keywords

SMS Spam, Smart Phones, Access Layer Detection

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'11, July 12–16, 2011, Dublin, Ireland.

Copyright 2011 ACM 978-1-4503-0557-0/11/07 ...\$10.00.

## 1. INTRODUCTION

Short Message Service (SMS) is one of the most popular and ubiquitously adopted data communication services on mobile devices. The prime factor behind this is its simplicity for end users and huge revenue generation for service providers. A recent report by Portio Research, an independent UK-based research company, shows that more than 5.5 trillion text messages were sent over carrier networks worldwide in year 2009 [20]. This trend appears to be increasing as a survey projected that 6.6 trillion SMS messages would be exchanged globally during 2010. SMS is widely used in automated *information retrieval systems*, mobile banking, over-the-air (OTA) mobile configuration, social web site alerts (e.g., Facebook), and for user authentication.

With this level of usage, the volume of spam SMS received by mobile users has also dramatically increased. A recent study, by the messaging security firm Cloudmark showed that 66% of UK mobile phone users have received spam text messages [26]. Another survey [15] reported that the number of spam SMS exceeds more than 50% of the total SMS messages received by consumers. Moreover, it was observed that more than 200 million subscribers were hit by SMS spam in a single day [25]. Apart from simple advertisement messages, SMS spam has also been used in modern sophisticated attacks, like the stealing of personal information, phishing, and scam propagation schemes [26]. One important point is that the majority of spam SMS is sent directly by (or through) cellular companies or on behalf of third party providers [3]. Therefore, it is important and relevant to filter the SMS spam on mobile devices and smart phones. Despite the severity of the problem, SMS spam detection has received little attention in the research community.

We, therefore, performed an empirical study to analyze the applicability of different evolutionary and non-evolutionary classifiers in solving the real-world problem of detecting SMS spam on mobile devices. We here propose a real-time SMS spam detection framework that models the byte-level transitions of an SMS message as a potential input attribute for the different classification algorithms. Our framework functions at the access layer of a mobile device; as a result, it silently moves a spam SMS message into a spam folder without disturbing the user through ring tone or vibration alerts. We selected four evolutionary and four non-evolutionary algorithms from various machine learning schemes. These classifiers are listed in Table 2. We evaluated these classifiers on a real-world dataset of SMS. The SMS spam dataset consisted of more than 2000 messages and is collected through Grumbletext [11] and from our

research lab. We also collected more than 5,000 benign SMS messages through our customized mobile terminal interface. These benign SMS messages were obtained from a large number of volunteers in our social network with diverse socio-economic backgrounds, including engineers, students, housewives, professionals, and corporate employees.

The results of our experiments show that our framework achieves a detection rate of more than 93% with a false alarm rate  $\leq 0.13\%$  when distinguishing between benign and spam SMS. It needs little memory to store the features vector and less than one second for classification (when using evolutionary classifiers); as a result, it could easily be deployed and integrated into the base band processor of a mobile phone.

The rest of this paper is organized as follows. We present related work on the mitigation of SMS spam in Section 2. The SMS spam detection framework is proposed in Section 3, followed by an explanation of feature extraction in Section 4. We discuss the classifiers in Section 5 and describe the evaluation strategy and dataset in Section 6. We report the results of our experiments in Section 7. Finally, we conclude the paper with a discussion of our future work.

## 2. RELATED WORK

Many cellular operators have recently devised sophisticated mechanisms, like Open Mobile Alliance (OMA), to protect mobile users from SMS spam [12]. While these administrative measures cut off a massive quantity of spam directed towards consumers, the problem of personalized spam filtering on mobile devices remains unsolved. The existing spam filtering techniques for mobile phones are based on the content of SMS [6][7]. Most of these techniques are straightforward adaptations of email spam detection schemes and usually incorporate features—specific words, character bi-grams, and tri-grams—for the classification of spam messages [10]. Some other works (such as [13]) have proposed schemes based on machine learning algorithms (like SVM, KNN, or Naïve Bayes) for the classification of SMS spam. Recently, the author of [23] used a Hidden Markov Model to detect SMS spam messages at the access layer of mobile devices. Evolutionary algorithms have been applied to solve real-world problems in the domain of email spam [16][17][21]. However, no previous study has incorporated evolutionary algorithms for the detection of SMS spam on mobile devices.

## 3. SMS SPAM DETECTION FRAMEWORK

We now present a SMS spam detection framework that works at the access layer of a mobile phone to detect spam messages. We set the following requirements for our spam detection filter: (1) it should correctly identify spam and benign SMS, (2) the learning process must be robust to word adulteration techniques, and (3) it should be real-time deployable on smart phones, i.e., it should require little processing and storage resources. This framework is the generic form of the spam detection system proposed by the authors of [23], which used Hidden Markov Model to detect the spam SMS at the access layer of mobile device. Our framework is modular in nature, which allows it to operate on the different architectures of mobile devices and smart phones.

### 3.1 Modules

Figure 1 shows the architecture of our SMS spam detection framework inside a mobile phone. Note that our frame-

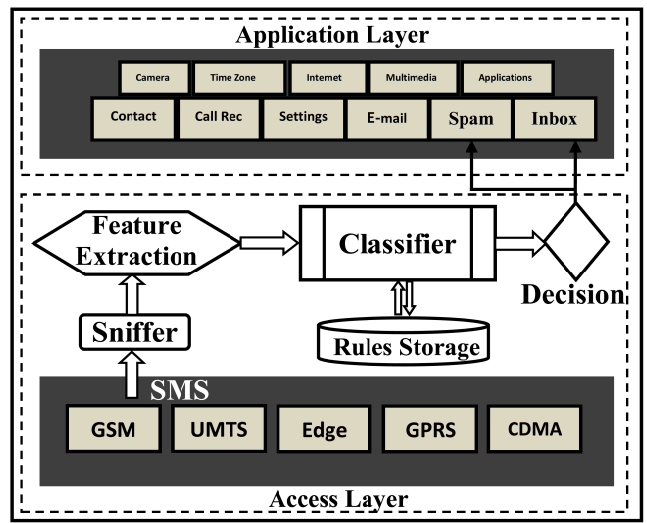


Figure 1: SMS Spam Detection Architecture.

work receives the SMS from the base-band processor<sup>1</sup> of the mobile device at the access layer. This allows us to sniff the SMS before it reaches the application layer.

The SMS sniffer module captures the SMS from the modem in a specific format called as SMS-DELIVER Protocol Description Unit (PDU)[9] with complete SMS payload (UserData) and headers in a hexadecimal representation. Figure 2 depicts the format of a complete SMS captured by the SMS sniffer module at an access layer. The SMS user data (TP-UD) (maximum of 140 bytes) is extracted to analyze the message and determine whether it is benign or spam. This user data can be encoded in different formats and are used for displaying different kinds of messages, e.g., Unicode (UCS2) text message, WAP, Over the Air (OTA) configuration messages, etc. [9]. The Data-Coding-Scheme (TP-DCS) field in the SMS header is used to indicate the particular encoding of the SMS. We designed our framework to analyze all of the encoded SMS formats to detect various kinds of spam messages that can be received by a user.

As the user data are extracted, the feature extraction module computes the byte-level (hexadecimal octet) transitions of SMS TP-UD as potential features. Then, the extracted features from different encoding schemes are given as input to the classifier. The idea of incorporating byte-level features for different encoding schemes to detect SMS spam was proposed in [23]. The advantage of byte-level feature extraction is that the spam detection can also be performed on messages with a modern composition style such as short abbreviated spam SMS (through alphabets or special characters) that are not actually part of any formal language.

The classification module takes the values of the byte-

<sup>1</sup>In current mobile devices, the SMS is received by the GSM modem from a Short Message Service Center (SMSC). GSM modem provides an interface between the cellular network and the application processor of a mobile device. The SMSs received by the modem are delivered to the OS of the application processor through the telephony stack. AT commands are used to control the GSM modem inside the smart phone. The telephony stack decodes the API calls (Application Program Interface) into corresponding AT commands and AT result codes depending on the types of message.

Oct. no.	Bit no	7	6	5	4	3	2	1	0
Address of SMSC max. (12 bytes)	1	Length of SMSC Address Information							Address Length
	1	1	Type of Number			Numbering Plan Identification			Type-of-Address
	1	SMSC Number in Semi Octet Representation							Address Value
	2								
	x								
Address of Sender max. (12 bytes)	1	TP-RP	TP-UDHI	TP-SRI	X	X	TP-MMS	TP-MTI	First-Octet(M)
	1	Length of Sender Address Information							Address Length
	1	1	Type of Number			Numbering Plan Identification			Type-of-Address
	1	Sender Number in Semi Octet Representation							Address Value
	x								
Time Stamp 7 bytes	1	Bits 7-6 TP-PID		Bit 5 TP-PID		Bits 4...0 TP-PID			TP-PID(M)
	1	Bits 7-4 TP-DCS			Bits 3-0 TP-DCS			TP-DCS(M)	
	1	Year							TP-SCTS(M) in Semi-Octet Format
	2	Month							
	-	Day							
-	Hour								
-	Minute								
-	Second								
7	Time Zone								
1	User Data Length							TP-UDL(M)	

Figure 2: SMS-DELIVER Format.

level features as input and solves a binary classification (benign vs. spam) problem with the help of rules (population) stored in the rules database. In this work, we evaluated the performance of the spam detection framework for different classification algorithms. These algorithms are described in Section 5. The classification decision made by the classification module is used to decide between two alternatives: (1) the SMS is silently moved to the spam folder if the classifier determines that it is spam, (2) otherwise, the message is moved to the Inbox and the user is notified.

### 3.2 Methodology

Our spam detection methodology consists of two phases. In the training phase, benign and spam labeled messages from the user’s Inbox are passed to the spam detection system. The classification module learns its rules or evolves its population based on the byte-level transitions of these labeled messages in different encoding schemes. For accurate classification, the spam detection filter can be instantly updated on smart phones to accommodate textual changes in the received spam or benign messages. Once the training phase of the spam detection filter is completed, it starts functioning on incoming SMS to classify spam messages based on the behavior learned during the training phase.

### 4. FEATURE EXTRACTION

Our spam detection framework can be customized to use any set of features extracted from the User Data of an SMS-DELIVER PDU at the access layer of a mobile device. Usually the User Data in an SMS-DELIVER PDU is encoded using three different formats: 7-bit encoded messages (max. 160 characters), 8-bit messages (max. 140 characters), and 16-bit messages (max. 70 characters) [9]. Table 1 shows the encoding of an SMS in different formats. We argue that, for the effective detection of spam SMS, the feature extraction module should be robust to word adulteration techniques. In other words, it should not use the keyword features and it should be real-time, i.e., it should require less memory and processing time on the resource constrained mobile devices.

We therefore present SMS Message  $M$  as a sequence of octets (bytes)  $M = \{b_0, b_1, b_2, \dots, b_l\}$ , where  $l$  is the length of the User Data. With no loss of generality, we can also treat every successive  $n$  octets in  $M$  as our input attributes for the classification module. This is commonly referred to  $n - gram$  feature space, where the frequency of  $n$  bytes in

Table 1: SMS in Different Encoding Schemes.

	$c$	$n$	$g$	$r$	$t$	$s$	$u$	$w$	$i$	$n$		
7-	43	F7	59	4E	9F	83	EA	A0	7B	DA	0D	
8-	43	6E	67	72	74	73	20	75	20	77	69	6E
16-	0043	006E	0067	0072	0074	0073	0020	0075	0020	0077	0060	006E

$M$  acts as a measure of its importance in  $M$ . The right choice of the value of  $n$  plays a critical role in such systems: if  $n$  is too small, the probability of false detection increases. In contrast, if we choose a large value for  $n$ , it significantly increases the processing overhead, making the framework infeasible for resource constrained mobile devices. Furthermore, the joint distribution of the  $n - gram$  feature space may contain redundant information, which may result in an inaccurate analysis for a given problem [1]. For the real-time detection of spam SMS, we need a model that not only extracts the relevant underlying information but is efficient in terms of system throughput.

We therefore model the octet sequence of SMS using a Markov process [8] with a discrete (finite or countable) state-space. The Markov state-space characterizes the conditional distribution of a random process. The conditional distribution has an advantage over the joint distribution ( $n - gram$  feature space) because it reduces the underlying feature space [1]. This is important for our SMS spam detection framework because it corresponds to the removal of redundant information from the joint distribution.

The order of the state-space, induced by the Markov process, depicts the number of past states (octets) needed to determine the current state (next octet). A correlation analysis of different encoding schemes shows a 1<sup>st</sup> order dependence for the byte-level distribution of SMS [23]. For our generic spam detection framework, we modeled the byte-level distribution of SMS message  $M$  in  $k$  states, i.e.,  $\theta_b = \{\theta_{b_0}, \theta_{b_1}, \theta_{b_2}, \dots, \theta_{b_k}\}$ . Transition function  $\psi$  operates on each incoming SMS  $M$  and embeds the spatial information in the form of a transition probability of states. With each byte value  $b_i$ , such that  $b_i \rightarrow \theta_{b_i}$ , we computed the transition probability of byte  $b_{i+1}$  followed by byte  $b_i$  as  $t_{\{\theta_{b_i}, \theta_{b_{i+1}}\}}$ . The transition probability values of  $k$  states are represented by Transition matrix  $T(\theta_b)$ . Mathematically, entrenching function  $\psi$  is represented as:

$$\psi(M) \rightarrow (T(\theta_b))_{\theta_b \in M} \quad (1)$$

and the Transition Matrix as :

$$T(\theta_b) = \begin{bmatrix} t_{\theta_{00}, \theta_{00}} & t_{\theta_{00}, \theta_{01}} & \dots & t_{\theta_{00}, \theta_{FF}} \\ t_{\theta_{01}, \theta_{00}} & t_{\theta_{01}, \theta_{01}} & \dots & t_{\theta_{01}, \theta_{FF}} \\ \vdots & \vdots & \ddots & \vdots \\ t_{\theta_{FF}, \theta_{00}} & t_{\theta_{FF}, \theta_{01}} & \dots & t_{\theta_{FF}, \theta_{FF}} \end{bmatrix} \quad (2)$$

Note that the number of states in the transition matrix ranges from 00 to FF. This corresponds to a feature space of 64KB, making it feasible to deploy on resource constrained mobile devices. The transitions in bytes represent the complete formation of the SMS. Any deviations in the transition values reflects a different composition for the SMS message. This situation indicates anomalous user data—possibly a spam message—received by the user. The transition ma-

**Table 2: Classification Algorithms & Standards.**

Evolutionary	
CLASSIFIER	STANDARD
XCS	Michigan-Style GMBL
SLAVE	Genetic Fuzzy
UCS	Michigan-Style GMBL
cAnt-Miner	Ant Colony Optimization
Non-Evolutionary	
CLASSIFIER	STANDARD
RIPPER	Inductive Rule Learning
Naïve Bayes	Statistical Model
C4.5	Decision Tree Induction
C-SVM	Support Vector Machine

trix values are given as potential features to the classification module, and the instances are stored in the ARFF format.

## 5. CLASSIFICATION MODULE

We analyzed the use of evolutionary classifiers for the detection of spam SMS at the access layer using byte-level features. We also included reputable non-evolutionary algorithms to gain better insight into which classifier to select for our framework. We next evaluate the effectiveness of the classifiers based on their classification accuracy and run-time performance (their training and testing times). The goal is to select a suitable classifier that can effectively detect spam SMS at the access layer of a mobile device (smart phone).

We selected a diverse set of reputable evolutionary and non-evolutionary classification algorithms for our study. The underlying rationale for choosing different classification algorithms in our study was an attempt to cover a broad set of standards in evolutionary computing and machine learning. The classifiers and standards used in this study are summarized in Table 2. We now provide a brief description of each classifier. An interested reader can find a detailed description of these algorithms in the cited references.

### 5.1 Evolutionary Algorithms

#### 5.1.1 *eXtended Classifier System (XCS)*

XCS is a Michigan-style classifier [27] that evolves a set of rules as a population of classifiers and is based on accuracy. The algorithm derives the complete ruleset from the whole population, where each individual represents a single rule. Training data are used to compute the initial rules, which are then evolved into new rules using a niche Genetic Algorithm (GA) that is stored in the population. The fitness of the individual is based on the prediction accuracy of each rule. A prediction array is used to store the expected payoff values for all of the actions. Depending upon the knowledge of the rules, some of them are also deleted from the population. The evolution of accurate generalizations and real value representations make XCS adaptable to solve various real-world problems [28].

#### 5.1.2 *sUpervised Classifier System (UCS)*

UCS [4] is also an accuracy-based, Michigan-style classifier. It works on the same principle as XCS and evolves an initial population of rules from training data. However, it differs from XCS as it is based on a supervised learning

scheme that computes fitness instead of the reinforcement learning employed by XCS. The genetic algorithm is applied to correct the rule-set for updating its population. Furthermore, UCS does not maintain a prediction array. The online learning and evolving characteristic of UCS can be easily exploited to solve real-world problems and has been well studied in the literature.

#### 5.1.3 *Continuous Ant-Miner Algorithm (cAnt-Miner)*

Ant-Miner is a data mining algorithm based on the Ant Colony Optimization (ACO) algorithm [19]. The ant colony algorithm is based on the behavior of ants searching for food and finding an optimal path. Ants randomly begin searching the food and leave “markers” (pheromones) on their paths. There is a certain probability that other ants, when they come across these markers, will follow the same path. A particular path to the food source will be more populated with pheromones if more ants traverse that path. The shorter path, with more concentrated pheromones, is the optimal path to the source of food. The Ant-Miner algorithm translates this behavior into a classification domain. It starts creating a rule one at a time with an empty rule set by accumulating terms for a partial rule in a probabilistic manner. The quality of the rule determines the iterative addition of the pheromone (updating of the rule). Finally, the optimal rules are generated by defining a threshold on a pheromone value. We used the cAnt-Miner algorithm (for real values) [18], an extension of Ant-miner, to classify the SMS spam.

#### 5.1.4 *SLAVE Algorithm*

SLAVE (Structural Learning Algorithm in Vague Environment) is a genetic learning algorithm based on the use of fuzzy logic concepts and the genetic iterative approach. It selects and learns only one fuzzy rule in each iteration as the result of reducing the search space of possible solutions. After that, it fixes a class and selects the best prior for this class. The complete set of rules is acquired through a sequence of repeated iterations, which is eventually used for classification. Completeness and consistency are the parameters used to determine the fitness of the rules.

### 5.2 Non-Evolutionary Algorithms

#### 5.2.1 *Support Vector Machine (C-SVM)*

Support Vector Machines [14] convert multifarious domain knowledge with overlapping inputs in to non-overlapping parametric objects by modeling the instances from the input space to the feature space using kernel functions. The classification is done by constructing a hyper plane between instances of different classes. C-SVM—a variant of SVM—is a commonly used classifier for a two class problem. The commonly used kernels in SVM are Linear, Polynomial, Radial Basis Function (RBF) and Sigmoid kernels.

#### 5.2.2 *C4.5 Algorithm*

C4.5 [22] is based on the concept of information theory and generates a decision tree to solve the classification problem. The algorithm operates by mapping a set of features to a specific class. The decision tree is constructed from training data based on the information gain of features. It recursively operates on the subsets of the features until all of the features are evaluated or no additional information is gained

by splitting the remaining features. The features with the largest information gain are considered for classification.

### 5.2.3 Repeated Incremental Pruning to Produce Error Reduction (RIPPER)

RIPPER [5] is based on associating rules with reduced error pruning (REP). RIPPER iteratively constructs rule-sets through the *greedy* induction of rules from the training data. The reduction in the size of the rule-set and its fitness to the dataset is obtained through postpass optimization. To prevent over fitting, the cross-validation and minimum-description length techniques are used in combination.

### 5.2.4 Naïve Bayes Algorithm (NB)

Naïve Bayes [24] is a probabilistic classifier based on the assumption that the presence/absence of attributes are statistically unrelated. The Naïve Bayes classifier can be effectively trained through supervised learning depending on the nature of the probabilistic model. More precisely, given a set of features  $A = \{a_1, a_2, a_3, \dots, a_n\}$  and Class  $C$ , the presence probability is effectively calculated as  $P(A/C) = \prod_{i=1}^{|A|} P(A_i/C)$ . The algorithm is well known for its performance in many real-world applications.

## 6. EVALUATION STRATEGY

In this section, we present our approach for evaluating and comparing the above-mentioned algorithms. Our main goal was to analyze the detection capability of different algorithms on actual SMS messages received by mobile users. Therefore, we first discuss our approach for collecting a real-world benign and spam dataset. We present our tool to acquire an SMS dataset in the PDU format. Afterwards, we define the performance metrics for the detection of spam SMS to carry out an unbiased analysis of the different classifiers. Finally, we show the results of our experiments.

### 6.1 Dataset Acquisition

We built a collection of more than 5000 real-world benign and 800 spam SMS by organizing a volunteer campaign on the premises of our university and in the local community. The purpose of the campaign was to collect and analyze both spam and benign SMS received by mobile users. The participants in our campaign included teenagers, researchers, students, professionals, teachers and some senior citizens. This gave us a diverse set of SMS messages that were received by users on their mobile devices and smart phones.

To read the messages in the SMS-DELIVER format, we developed a *modem terminal interface* that directly accesses SMS from the memory of the baseband processor of a mobile phone. Our interface interacts serially with the modem of a mobile device through AT commands. It first configures the modem to operate in PDU mode by giving the AT+CMGF=0 command. Once the modem is configured in the PDU mode, using AT+CMGL=ALL, all of the messages in the memory of the base-band processor of the mobile phone are redirected to the terminal. The user data is then extracted from SMS-DELIVER format in different encoding schemes.

Apart from our SMS collection campaign, we also collected more than 1200 spam messages from Grumbletext. Grumbletext is a UK-based consumer complaint web site, where cell phone users post messages online that they consider spam. We manually extracted and identified spam SMS by a thorough examination of numerous web pages.

To support our detection framework, the plain English text was converted into the SMS-DELIVER format using our customized PDU encoder written in C#. The encoder converted the messages in all three encoding schemes (see Section 4).

## 6.2 Performance Metrics

We now define the performance metrics used for an accurate evaluation and comparison of the different algorithms. The main goal was not to bias our approach to a particular classifier or learning standard for the detection of spam SMS. We used the following four performance metrics:

1. Detection Rate: The percentage of spam messages correctly classified as spam,
2. False Alarm Rate: The percentage of benign messages incorrectly classified as spam,
3. Training Time: The time taken (in seconds) by the classifier to train itself on the given dataset,
4. Testing Time: The time taken (in seconds) by the classifier to detect an SMS as benign or spam.

The Detection Rate (DR) shows the spam detection accuracy of a classifier. A higher DR indicates better spam detection. The False Alarm Rate (FAR) indicates the false detection of incoming messages. A classifier with a high FAR will move benign messages into the spam folder without user notification. The training time is important for periodically updating the rule-base of the classifiers on mobile device. The testing time shows the delay introduced in SMS reception (at the OS of the mobile device) by the classifier. These four parameters were very useful for evaluating the accuracy and efficiency of the algorithms on resource constrained mobile devices and smart phones.

### 6.3 Classification

We evaluated the classifiers by using an open source software KEEL [2]. For cANT-Miner we used the implementation of [18]. For an unbiased evaluation, we used standard parameter values for all of the classifiers. The classifiers are separately trained on messages in 7-bit, 8-bit, and 16-bit encoding schemes. We then used a stratified 10-fold cross validation procedure on the dataset of each encoding scheme. In this procedure, we partitioned each dataset into 10 sections, where 9 of them were used for training the classifiers and the remaining section was used for testing. This process was repeated for all of the sections, and the reported results are an average for all the sections.

## 7. EXPERIMENTS AND RESULTS

In this section, we present the results of our experiments. We first show our analysis of the quality of the attributes used for our classification module. We then discuss in detail the performance of the classifiers in terms of the performance metrics (see Section 6.2). Our evaluation results include the *detection rate*, *false alarm rate*, *training time*, and *testing time*. The classification accuracy (DR and FAR) results are summarized in Table 3 for 7-bit, 8-bit, and 16-bit encoded SMS. The efficiency analysis of the different classifiers (training and testing time) is shown in Table 4.

**Table 3: Classification Accuracy in (%) of Selected Algorithms on SMS Dataset.**

	7-bit				8/16-bit			
	Training		Testing		Training		Testing	
	DR	FAR	DR	FAR	DR	FAR	DR	FAR
XCS	64.2	2.7	5.6	0	68.3	2.1	10.2	4.34
UCS	52.05	3.98	52.2	4.7	53.4	4.2	52.2	0.23
cAnt-Miner	86.35	2.34	83.37	7.21	88.5	1.3	81.6	17.67
Slave	93.45	0.01	92.66	0.05	93.6	0.1	93.4	0.21
Naïve Bayes	87	19.22	87.7	25.5	84.5	2	83.8	6.32
RIPPER	83.8	3.24	77	6	83.6	6.3	86.5	3.44
C4.5	86.12	0.2	74	0.5	85.4	0.5	84.5	3.41
C-SVM	100	0.1	85.7	1.4	100	0.1	83.6	2.01

## 7.1 Qualitative Analysis of Byte-Level Features

We now present some general insights into the distinctive nature of the attributes extracted from our datasets. This analysis will be helpful in investigating the effectiveness of the evolutionary and non-evolutionary algorithms used to detect SMS spam at the access layer. The focus of our evaluation is based on the *quality* of the extracted features modeled using the 1<sup>st</sup> order Markov process (see Section 4). Several measures based on information and statistic theory parameters are used by researchers to examine the importance of features in a given dataset. Information gain and gain ratio are two widely used algorithms for the qualitative analysis of features [8] and are often employed for selecting distinctive features.

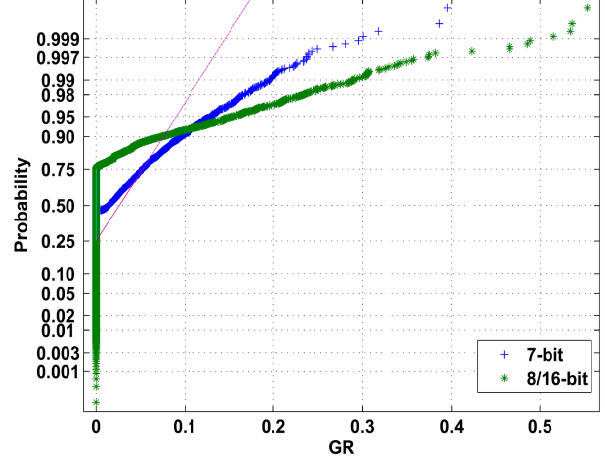
Information gain measures the decline in uncertainty having a prior knowledge of attribute values [8]. With class attribute  $C \in \{Benign, Spam\}$  and feature  $F$ , the uncertainty is given by their respective entropies  $H(C)$  and  $H(F)$ . Mathematically, the information gain of  $F$  with respect to  $C$  is given as:

$$IG(C; F) = H(C) - H(C|F) \quad (3)$$

We believe that evaluating the attributes by using information gain may not give us accurate results for the given problem, as it will bias our evaluation towards features split in defined classes [8]. This draw-back of information gain can be overcome by using Gain Ratio (GR). The gain ratio of feature  $F$  with respect to class attribute  $C \in \{Benign, Spam\}$  is given as  $GR(C; F)$ , where

$$GR(C; F) = \frac{IG(C; F)}{H(F)} = \frac{H(C) - H(C|F)}{H(F)} \quad (4)$$

We performed our analysis of the byte transition values gain ratio (extracted features' gain ratio) distribution for the SMS dataset using 7-bit and 8/16-bit encoding schemes. Figure 3 shows the normal probability plot for the gain ratio of the attributes of the SMS dataset using different encoding schemes. It can be seen that the behavior of gain ratio features' distribution is 'skewed' on all of the encoding schemes. In actuality, the gain ratio value ranges from 0 to 1. The values near 1 depict a higher classification potential of extracted features and vice versa. In our case, it is clear from Figure 3 that the majority of features have very low gain ratios and can proved to be redundant for classification algorithms. In contrast, very few features have gain ratio values greater than 0.3. These features with high gain ratio


**Figure 3: Normal Probability Plot of Gain Ratio.**

values can be considered vital for the accurate classification of spam messages at the access layer of mobile devices.

## 7.2 Classification Accuracy

The DR and FAR results of the classifiers for 7-bit, 8-bit and 16-bit encoding on the real-world SMS spam dataset are discussed below.

### 7.2.1 Training Accuracy

From Table 3, we can see that all of the algorithms show different training accuracies for the 7-bit encoded messages. Most of the algorithms (except SLAVE and C-SVM) achieved low training accuracies with considerably high FAR values. This is an expected result of the byte-level features of SMS messages, as inferred from our qualitative analysis. The genetic fuzzy algorithm SLAVE achieved the highest training accuracy, with a DR of 93.45% and a 0.01% false alarm rate. Naïve Bayes achieved the worst FAR of 19.22% when trained using 7-bit encoded messages. The high FAR of the Naïve Bayes algorithm was the result of the *bit-swap* in the octets of the SMS message in the 7-bit encoded scheme [9]. This *bit-swapping* in the 7-bit message configuration contravened the Naïve Bayes basic theory of the independence of attribute values. C-SVM was the only non-evolutionary algorithm achieved detection rate of 100%, but at the cost of a 0.1% FAR.

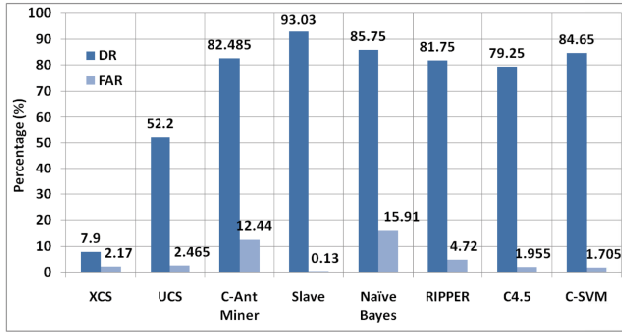


Figure 4: Average Detection & False Alarm Rate.

The 8/16-bit encoded messages are the direct representation of the characters in SMS as compared to 7-bit encoded messages. This direct mapping of characters to the octets representation makes the detection of spam SMS easier (see Section 7.1). Therefore, all of the algorithms showed improvement in the training accuracy results on 8/16-bit encoded messages, as compared to 7-bit encoded messages. Table 3 shows that SLAVE had the lowest FAR of 0.1%, with a detection rate of 93.6%. UCS achieved 4.2% FAR on training but with the lowest DR of 53.4%. C-SVM achieved the highest training accuracy, with the DR of 100% but at the cost of a 0.1% false alarm rate. RIPPER had the highest FAR of 6.3%, with the DR of 83.6%.

### 7.2.2 Testing Accuracy

Table 3 shows the diversity in the testing accuracies of the classifiers on 7-bit encoded messages. The evolutionary classifier SLAVE was able to evolve an effective rule for filtering the SMS spam of 7-bit encoded messages. SLAVE showed the best combined testing accuracy results, with a 92.66% DR and 0.05% FAR. XCS had the lowest FAR of 0%, with the worst DR of 5.6%, and was not an effective algorithm to detect spam messages in 7-bit encoding. cAnt-Miner and UCS achieved DR values of 83.37% and 52.2% with 7.21% and 4.7% FAR respectively. The non-evolutionary algorithms, RIPPER, C4.5, and C-SVM, achieved the DR of 77%, 74% and 85.7% respectively, but at the cost of relatively high FAR values of 6%, 0.5%, and 1.4% respectively. Naïve Bayes had the worst FAR of 25.5%, with a DR of 87.7%. To conclude, SLAVE stood out as the most accurate evolutionary algorithm for classifying 7-bit encoded messages with the lowest false detection of benign messages and high detection of spam messages.

Table 3 shows a noticeable improvement in the FAR values of the non-evolutionary algorithms for the 8/16-bit encoded messages testing accuracy results, as compared to 7-bit encoded messages. The non-evolutionary algorithms, Naïve Bayes, RIPPER, C4.5, and C-SVM, achieved DR values of 83.8%, 86.5%, 84.5% and 83.6% but with 6.32%, 3.44%, 3.41%, and 2.01% FAR values, respectively. The evolutionary algorithm SLAVE again achieved the lowest FAR of 0.21%, with a detection rate of 93.4%. XCS and UCS achieved 4.34% and 0.23% FAR values with DR of 10.2% and 52.2%, respectively. Despite achieving good training

Table 4: Average Training & Testing Times (secs).

	7-bit		8/16-bit	
	Training	Testing	Training	Testing
XCS	415.22	0.021	431.916	0.02
UCS	397.88	0.054	383.97	0.043
cAnt-Miner	28	0.021	27.65	0.016
Slave	180	0.448	177.6	0.367
Naïve Bayes	2.14	0.001	2.25	0.001
RIPPER	56.61	0.001	30	0.001
C4.5	35.86	0.001	35.44	0.001
C-SVM	5.75	0.001	5.15	0.001

accuracy, cAnt-Miner achieved the worst FAR of 17.67% in the detection of 8/16-bit encoded messages. This means that the rules evolved as a result of training the cAnt-Miner algorithm provided good accuracy on the transitions on which it was trained, but poor accuracy when the byte-level transitions of the messages varied significantly.

For the current problem, a high false detection of benign messages is unacceptable for mobile phone users in many real-world scenarios (e.g., Bank notification SMS). Therefore, the optimal classification of spam messages can be obtained with a high detection rate and a false alarm rate of  $\leq 1\%$ . Figure 4 shows the average detection and false alarm rates of the classifiers for 7-bit and 8/16-bit encoded messages. Note that most of the detection rates in the center of the figure are distributed around the 80% mark. Figure 4 also shows that SLAVE is the only algorithm that achieves the optimum accuracy for both 7-bit and 8/16-bit encoded messages with an average FAR of 0.13%. If we restrict ourselves to non-evolutionary algorithms, the C-SVM classifier becomes the optimal choice for the problem at hand but with a cost of 1.75% FAR.

## 7.3 Training and Testing Time Analysis

### 7.3.1 Training Time

The training time is the time required by the classifier to evolve its rule for detection. For the given problem, the training time is of less importance than the testing time. It is not a problem for a mobile device to spend several minutes in training during idle hours (e.g., at night). Table 4 shows that the non-evolutionary algorithms outperformed the evolutionary algorithms (except cAnt-Miner) in the training time. The lowest training time was achieved by Naïve Bayes, which only took 2.14 seconds to build the model. The cAnt-Miner algorithm had the lowest training time among the evolutionary algorithms. This was because cAnt-Miner does not employ any mutation or crossover procedures, which take a considerable amount of time during the evolving of rules in other evolutionary algorithms. The Michigan style LCS (XCS and UCS) consumed more time in training than the fuzzy rule learning classifier SLAVE.

### 7.3.2 Testing Time

Table 4 shows that the C4.5, RIPPER, C-SVM, and Naïve Bayes algorithms had small testing times, similar to their smaller training times. The cAnt-Miner and XCS algorithms had the lowest testing times of evolutionary classifiers. The fuzzy rule learning classifier SLAVE had the highest testing time compared to all of the other classifiers. A higher testing

time for an algorithm might result in the delayed reception of SMS on a mobile device and smart phone.

## 7.4 Implications

It is obvious from our classification accuracy results that the evolutionary algorithm SLAVE had the lowest FAR, with an average DR of 93.03%. While achieving such outstanding results, the SLAVE algorithm was less efficient than the non-evolutionary algorithms in terms of speed. Naïve Bayes turned out to be the fastest classifier but at cost of high FAR. Likewise, some of the other algorithms that took less testing time had high FAR values. We believe that the testing times of the evolutionary classifiers could be notably decreased by using efficient feature selection schemes. The redundant features could be removed by using quality measures such as information gain and gain ratio. The higher ranking features could then be selected in the training of classifiers. This would not only improve the classification accuracy but also decrease the testing time by reducing the complexity of the evolved rules.

## 8. CONCLUSION AND FUTURE WORK

In this paper, we presented a real-time spam detection architecture that models the byte-level transitions of SMS PDU as a potential input attribute for the different classification algorithms. We performed a comprehensive study of four evolutionary and four non-evolutionary classifiers. The other important contribution of this paper is the collection of real-world SMS messages and its classification in different underlying encoding schemes. To the best of our knowledge, this is the first investigative study of evolutionary and non-evolutionary classifiers to develop a real-time detection filter for solving the SMS spam problem at the access layer of smart phones. The focus of our future work will be to study the effect of features selection on the classification accuracy at the access layer. We also plan to extend this work to the mitigation of other security threats in mobile devices.

## 9. REFERENCES

- [1] AHMED, F., HAMEED, H., SHAFIQ, M. Z., AND FAROOQ, M. Using spatio-temporal information in API calls with machine learning algorithms for malware detection. In *Proceedings of the 2nd ACM workshop on security and artificial intelligence* (NY, USA, 2009), ACM, pp. 55–62.
- [2] ALCALÁ-FDEZ, J., SÁNCHEZ, L., GARCÍA, S., DEL JESUS, M., VENTURA, S., GARRELL, J., OTERO, J., ROMERO, C., BACARDIT, J., RIVAS, V., ET AL. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Comput.* 13, 3 (Oct. 2008), Springer-Verlag, 307–318.
- [3] BERGSTÉN, H. Comprehensive study gives insight into mobile spam, June 2005. [http://www.ericsson.com/ericsson/corpinfo/publications/telecomreport/archive/2005/june/mobile\\_spam.shtml](http://www.ericsson.com/ericsson/corpinfo/publications/telecomreport/archive/2005/june/mobile_spam.shtml).
- [4] BERNADÓ-MANSILLA, E., AND GARRELL-GUIU, J. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evolutionary Computation* 11, 3 (2003), MIT Press, 209–238.
- [5] COHEN, W. Fast Effective Rule Induction. In *Proceedings of the 12th International Conference on Machine Learning* (July 1995), Morgan Kaufmann, pp. 115–123.
- [6] CORMACK, G. V., GÓMEZ HIDALGO, J. M., AND SÁNZ, E. P. Feature engineering for mobile (SMS) spam filtering. In *Proceedings of the 30th annual conference on Research and development in information retrieval* (New York, NY, USA, 2007), ACM, pp. 871–872.
- [7] CORMACK, G. V., GÓMEZ HIDALGO, J. M., AND SÁNZ, E. P. Spam filtering for short messages. In *Proceedings of the 16th ACM Conference on information and knowledge management* (NY, USA, 2007), ACM, pp. 313–320.
- [8] COVER, T., THOMAS, J., WILEY, J., ET AL. *Elements of information theory*, vol. 306. Wiley Online Library, 1991.
- [9] ETSI-GSM. 03.40 Technical realization of the SMS. <http://www.3gpp.org/ftp/Specs/html-info/0340.htm>.
- [10] GÓMEZ HIDALGO, J. M., BRINGAS, G. C., SÁNZ, E. P., AND GARCÍA, F. C. Content based SMS spam filtering. In *Proceedings of the 2006 ACM symposium on Document engineering* (NY, USA, 2006), ACM, pp. 107–114.
- [11] GRUMBLETEXT. UK consumer complaints. <http://http://www.grumbletext.co.uk/>.
- [12] GSMWORLD. GSM to Address Spam and Fraudulent Messaging Threats for Consumers, Mar. 2010. <http://gsmworld.com/newsroom/press-releases/2010/4797.htm>.
- [13] HEALY, M., DELANY, S., AND ZAMOLOTSKIKH, A. An assessment of case-based reasoning for short text message classification. In *Proceedings of 16th Irish Conference on Artificial Intelligence and Cognitive Science* (Sept. 2005), AICS '05, pp. 257–266.
- [14] HILL, T., AND LEWICKI, P. *Statistics: methods and applications*. StatSoft, 2006.
- [15] IRONPORT. Case-Study: IronPort Helps a Nationwide Carrier Stop Wireless Threats. [http://www.ironport.com/pdf/ironport\\_case\\_study\\_wireless.pdf](http://www.ironport.com/pdf/ironport_case_study_wireless.pdf).
- [16] ODA, T., AND WHITE, T. Developing an immunity to spam. In *Proceedings of the 2003 international conference on Genetic and evolutionary computation* (Berlin, Heidelberg, 2003), GECCO'03, Springer-Verlag, pp. 231–242.
- [17] ODA, T., AND WHITE, T. Increasing the accuracy of a spam-detecting artificial immune system. In *The 2003 Congress on Evolutionary Computation* (Dec. 2003), vol. 1 of *CEC '03*, IEEE, pp. 390 – 396.
- [18] OTERO, F. E., FREITAS, A. A., AND JOHNSON, C. G. cAnt-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes. In *Proceedings of the 6th international conference on Ant Colony Optimization and Swarm Intelligence* (Berlin, Heidelberg, 2008), ANTS '08, Springer-Verlag, pp. 48–59.
- [19] PARPINELLI, R., LOPES, H., AND FREITAS, A. An Ant Colony Algorithm for Classification Rule Discovery. *Data Mining: A Heuristic Approach 208* (2002), 191–132.
- [20] PORTIO-RESEARCH. Mobile Messaging Future. <http://www.portioresearch.com/mobile-factbook-2009.php>.
- [21] QING, J., MAO, R., BIE, R., AND GAO, X. An AIS-based e-mail classification method. In *Proceedings of 5th international conference on Emerging intelligent computing technology and applications* (Berlin, Heidelberg, September 2009), ICIC'09, Springer-Verlag, pp. 492–499.
- [22] QUINLAN, J. R. Improved use of continuous attributes in C4.5. *Journal of Art. Int. Res.* 4 (March 1996), 77–90.
- [23] RAFIQUE, M. Z., AND FAROOQ, M. 'Be Liberal in What You Recieve' on your mobile phone. In *Proceedings of 20th Virus Bulletin Conference* (Vanc., Canada, 2010), VB '10.
- [24] RISH, I. An empirical study of the naive Bayes classifier. In *Workshop on Empirical Methods in Artificial Intelligence* (Seattle, USA, Aug. 2001), IJCAI 2001, pp. 41–46.
- [25] SOPHOS. 200 million cellphone users hit by SMS spam tidalwave in China, Mar. 2008. [http://www.sophos.com/pressoffice/news/articles/2008/03/china\\_sms.html](http://www.sophos.com/pressoffice/news/articles/2008/03/china_sms.html).
- [26] TEXT4EVER. White Paper: UK spam study, Oct. 2009. <http://www.txt4ever.com/study/spamstudy.pdf>.
- [27] WILSON, S. Generalization in the XCS classifier system. In *Proceedings of the 3rd Annual Genetic Programming Conference* (June 1998), Morgan Kaufmann, pp. 665–674.
- [28] WILSON, S. W. Get Real! XCS with Continuous-Valued Inputs. In *Learning Classifier Systems, From Found. to Apps.* (London, UK, 2000), Springer-Verlag, pp. 209–222.