Using Evolutionary Learning Classifiers To Do Mobile Spam (SMS) Filtering

M. Bilal Junaid, Muddassar Farooq
Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer & Emerging Sciences (NUCES)
Islamabad, Pakistan
bilal.junaid@nexginrc.org
muddassar.farooq@nu.edu.pk

ABSTRACT

In recent years, we have witnessed the dramatic increase in the volume of mobile SMS (Short Messaging Service) spam. The reason is that operators – owing to fierce market competition – have introduced packages that allow their customers to send unlimited SMS in less than $1 a month. It not only degrades the service of cellular operators but also compromises security and privacy of users. In this paper, we analyze SMS spam to identify novel features that distinguishes it from benign SMS (ham). The novelty of our approach is that we intercept the SMS at the access layer of a mobile phone – in hexadecimal format – and extract two features: (1) octet bigrams, and (2) frequency distribution of octets. Later, we provide these features to a number of evolutionary and non-evolutionary classifiers to identify the best classifier for our mobile spam filtering system. We evaluate the detection rate and false alarm rate of our system – using different classifiers – on a real world dataset. The results of our experiments show that sUpervised Classifier System (UCS), by operating on the the above-mentioned features' set, achieves more than 89% detection rate and 0% false alarm rate.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—*Security and protection*

General Terms

Experimentation, Security

Keywords

Network Security, System exploits, Spam SMS, Short Messaging Service, Evolutionary Classifiers

1. INTRODUCTION

In recent years, SMS is becoming the most popular service of wireless networks, especially in the developing countries. The most important reason behind its popularity is its ubiquitous availability at significantly low rates compared with the voice service of GSM network. In developing countries, operators – owing to fierce market competition – have introduced packages that allow its customers to send unlimited SMS in less than $1 a month. As a result, the service is being misused by imposters for sending SMS spam. In [1], it is shown that the number of spam SMS exceeds more than 50% of the total SMS messages received by the users. The other survey reports that more than 200 million cell phone users were hit by SMS spam in a single day [2]. The disturbing development is that the majority of spam SMS is sent directly by operators about their latest packages [5].

SMS spam not only intrudes the privacy of the users but also adds to their frustration and annoyance because in most mobile phones its arrival is indicated through an alert tone. In most of these low cost mobile sets – lacking usability interfaces – a user has to open a SMS before deleting it. Therefore, it is relevant to develop a filtering system that silently intercepts a SMS, analyzes it and moves it to a spam folder without any involvement of the user.

SMS spam is different from the email spam in three ways: (1) it does not contain a mailing list of recipients, (2) the message is less than 160 bytes only, and (3) the mobile spam filtering system has to be deployed on resource constrained mobile phones. In order to meet above-mentioned challenges, we have to design a mobile spam filtering system that meets four requirements:

- It operates at the access layer of a mobile phone; as a result, a spam SMS is silently moved to the spam folder without any intervention of the user;
- It is trained with relatively small number of SMS messages (in the order a couple of hundreds);
- It has a high detection rate with approximately a zero false alarm rate;
- Its resource requirements – memory and processing power – are significantly small to make it suitable for deployment on resource constrained mobile phones.

1.1 Our Major Contributions

The major contribution of our paper is a mobile spam filtering system that works at the access layer of a mobile

Table 1: Machine Learning Paradigms and Selected Algorithms (Ev=Evolutionary, NE=Non-Ev)

	Classification Paradigm	Algorithms
Ev	Statistical Learning + GBML	Fuzzy AdaBoost
Ev	Pittsburgh-Style GBML	GAssist-ADI
Ev	Michigan-Style GBML	XCS
Ev	Michigan-Style GBML	UCS
NE	Decision Tree Induction	C4.5
NE	Support Vector Machines	C-SVM
NE	Instance Based Learning	IBk
NE	Statistical Modeling	Naïve Bayes
NE	Rule Based learning	JRip

phone. It analyzes a SMS in hexadecimal notation and extracts two features from this format:(1) octet[1] bigrams, and (2) frequency distribution of bytes. We evaluate the feasibility of a number of evolutionary and non-evolutionary classifiers (see Table 1 for the list) – operating on the above-mentioned features' set – for our filtering system. To the best of our knowledge, it is the first comparative study to analyze the suitability of evolutionary and classical machine learning classifiers for filtering real world SMS spam dataset.

By doing empirical investigations, we are able to identify that UCS is able to meet our four requirements outlined in the previous subsection. Our UCS based spam filtering system has met three requirements: (1) it provides 89% detection rate with 0% false alarm rate, (2) it achieves this performance by training only on 500 messages, and (3) it takes less than 21 KB of memory to store the features' set and approximately 1 second to analyze a message at the access layer.

The rest of the paper is organized as follows. In Section 2, we will provide an overview of the related work by emphasizing different direction of our work. Section 3 describes the process used for collecting real world spam and ham datasets. In Section 4, we discuss the proposed spam filtering system. The results of empirical investigations are reported in Section 6. Finally, we conclude the paper with an outlook to our future work.

2. RELATED WORK

To the best of our knowledge, SMS spam has received little attention by researchers. (May be they do not want to take a proactive approach to nip the spam in the the mobile bud now.) Two types of approaches exist in the literature: (1) adapting content-based email techniques for analyzing the content of a SMS [7] and then classifying it, and (2) analyzing the hexadecimal representation of a SMS [14] and using bytes transition matrix for classification. The technique of [7] utilizes a relatively large features' vector and it is evaded easily by generating a local language SMS in roman English characters. Similarly, the technique of [14] requires 256 Kbytes of memory for features' space and uses a complex Hidden Markov Model (HMM) classifier that makes it infeasible for mobile phones. Recently, LOHIT [10] is proposed, which creates features' vectors and arranges them into clusters to classify them as spam or ham. But the major drawback of this method is higher dimensionality of the features' space. To conclude, none of the above-mentioned techniques appear to be suitable for resource constrained mobile phones.

[1]In the paper, we use the terms "octet" and "byte" interchangeably.

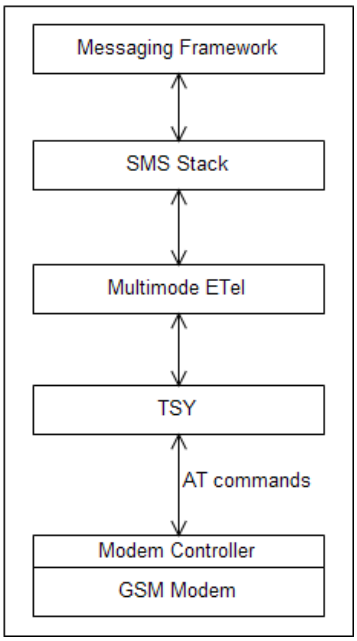

Figure 1: Logical SMS Architectures in phone

The idea of using two lists – black and white – to block spam SMS has a drawback that a user has to manually update the black list to block the spam sending imposters; as a result, it is unable to filter a spam SMS that has arrived from a new number or a spam that has arrived from a number in the contact's list (white list). The idea of declaring digitally singed SMS as ham and others as spam [18] is simply impractical for SMS because of two problems: (1) processing overhead, and (2) the signature needs to be transmitted with the SMS.

In comparison, our technique is novel because it tries to look at the contents of a SMS in hexadecimal notation at the access layer; therefore, it is not easy to evade it by doing word adulteration of symptomatic words (like `WON`, `CONGRATULATIONS!!!`, etc.) or massive use of punctuation marks [17]. Moreover, it uses a combination of octet bigrams and frequency distribution of bytes to reduce the dimensionality of the features' space that makes it suitable for resource constrained mobile phones.

3. DATASET COLLECTION

The baseband processor of a mobile phone (in case of 2.5G networks it is (`GSM`) modem) handles the SMS sent to it by `SMSC` (Short Message Service Center). The standard AT commands are used to control the `GSM` modem. As depicted in Figure 1, the baseband processor delivers the received SMS to the Telephony System Plug-in (TSY). TSY sends it to the (Etel) that in turn delivers it to the application processor using the SMS stack.

We have developed a *modem terminal interface* that can read SMS from the memory of a mobile phone in SMS-DELIVER format [6]. Our interface interacts serially with the modem through *AT commands* and intercepts a SMS be-

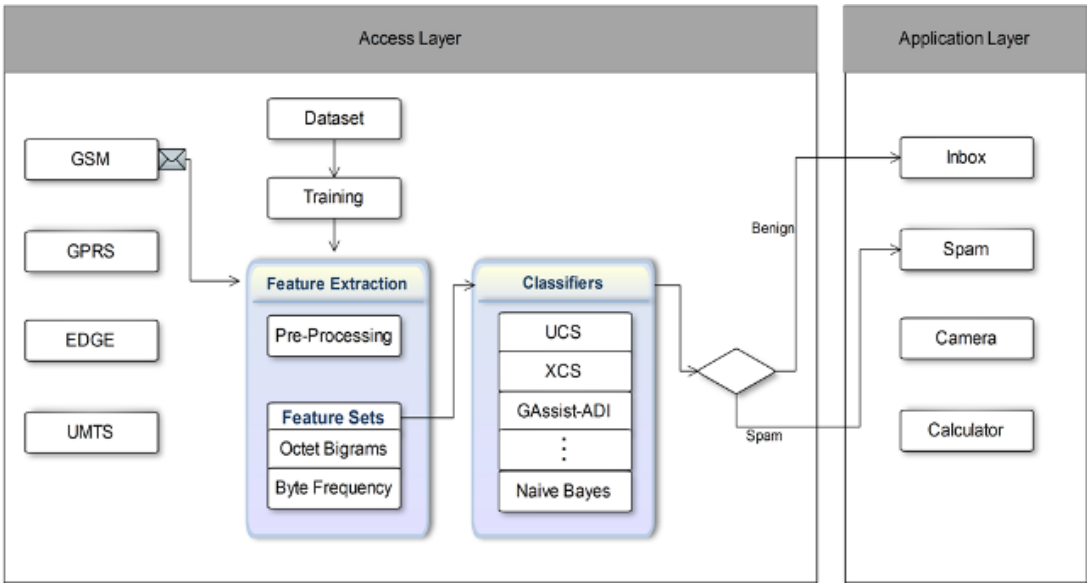

Figure 2: Spam Detection Framework

fore the interface. It first configures the modem to operate in *PDU* mode [2] by giving *AT+CMGF=0* command. Once the modem is configured in the *PDU* mode, using command *AT+CMGL=ALL*, all messages in the memory of a mobile phone are redirected to the terminal.

We have provided two mobile numbers to the users – belonging to different socioeconomic backgrounds: engineers, students, housewives, professionals and corporate employees – in our close social network to forward spam and ham at the two numbers respectively. (We have singed a non-disclosure agreement with our friends that neither we will read the contents of their benign SMS nor distribute it without their consent; however, the condition does not apply to the spam SMS.) We use our *modem terminal interface* to collect both types of messages in SMS-DELIVER format from the two SMS collecting mobile phone sets.

Moreover, we have also used the SMS messages posted at Grumbletext website (in UK) [3] that the users considered as spam. In order to use these messages for system, we have converted these plain English text into SMS-DELIVER using standard SMS encoders. As a result, a real world dataset of SMS that consists of more than 6000 benign SMS and more than 2000 spam SMS is generated for experiments.

We now discuss the architecture of our spam detection framework.

4. SMS SPAM DETECTION FRAMEWORK

The architecture of our SMS filtering system is shown in Figure 2. As mentioned before, it works at the access layer of a mobile phone; as a result, it silently moves the unsolicited SMS to a spam folder without providing any notification to the user (like its email counterpart systems). We now discuss the feature extraction module of our system.

[2]PDU (Protocol Description unit) mode is the one in which a modem is configured to read SMS in SMS-DELIVER format.
[3]`http://www.grumbletext.co.uk`

4.1 Feature Extraction

Recall from the previous section that our system intercepts a message before the TSY interface in SMS-DELIVER. Since, we aim at developing language independent filtering system, we extract the octet values in the hexadecimal format from the payload of SMS (user data). We design our features' set on the basis of hexadecimal octet values. This novel approach provides many benefits over searching for certain keywords or content in a message: (1) it is not sensitive to abbreviations and acronyms – depending on the culture and location of a social community – used by most mobile phone users, (2) it is also not sensitive to world adulteration by putting punctuation or exclamation marks, and (3) it is also not sensitive to local language SMS in roman English characters. We now explain two features of our features' set.

4.1.1 Octet bigrams

The classical techniques use *character bigrams*. For example the octet bigrams of "an egg" are: '0x414E', '0x4E20 ', '0x2045', '0x4547', and '4747'. Instead of creating bigrams of characters, we create octet bigrams at the access layer. To reduce the memory requirements, we map the octets of upper characters to the lower case characters also; as a result, the number of unique octets are fixed at 39. If all octet bigrams are equally likely to occur in the message, we need a vector of $39 \times 39 = 1521$ elements to hash all possible bigrams. The process of hashing is depicted in Figure 3.

4.1.2 Frequency distribution of bytes

According to GSM standards, `SMS` uses 7-bit and 8-bit encoding schemes. In 7 bit encoding scheme, each character is represented by 7 bits; as a result, only 128 characters could be encoded. In comparison, the 8-bit encoding scheme increases the character count to 256 characters. (Remember, we extract the byte values of a text message at the access layer.) In the next step a feature vector of 256 elements is created to cater for all possible byte values in a SMS. As a

consequence, it is possible to calculate the frequency (repetition) distribution of different octets in a given message. The process is shown in Figure 3.

4.2 Spam Detection Process

The process of Spam detection consists of two phases. In the training phase, the system is provided with a dataset that consists of benign and spam SMS. (We have tried datasets of different sizes and empirically determined that a dataset of 500 messages provides good training to the classifiers.) We take a number of different classifiers (see Table 1) and provide them with three different features' set: (1) octet bigrams only, (2) frequency distribution of bytes only, and (3) a merged features' set that contains both octet bigrams and their frequency distributions. The objective is to identify the best features' set.

Each classifier extracts the knowledge and uses its learning phase to build a model of spam and ham. Once the training is finished, the system is used to classify incoming messages in realtime. Finally, the performance of classifiers are analyzed and the classifier with the best performance is selected.

5. CLASSIFICATION ALGORITHMS

In this paper, our aim is to identify the best classification algorithm from a wide range of evolutionary and machine learning classifiers for the purpose of spam detection. We have selected four well known evolutionary and five non-evolutionary algorithms for our study. The selection criterion is to cover a wide spectrum of learning paradigms. We now provide a brief description of each classifier to make the paper self contained. Interested readers are referred to relevant references for further details.

5.1 Non-Evolutionary Classifiers

5.1.1 Naive Bayes Algorithm (NB)

Naïve Bayes algorithm creates a probabilistic model for classification. Even though all features contribute towards the overall probability of classification, Naïve Bayes algorithm assumes that the features are statistically independent of one another. Although this assumption may not hold true for all cases, Naïve Bayes algorithm has shown promising results compared with other well-known classification algorithms in real world applications [15]. An interested reader can find more details about Naïve Bayes algorithm in [15].

5.1.2 K Nearest Neighbor (IBk)

K-Nearest Neighbors algorithm is an example of instance based learning and classification paradigm. It is mostly used in pattern matching problems. In KNN, the training phase involves only storing feature values and corresponding classes. The classification is done through the voting of K training instances which are least distant from the instance under test. The least distant instances ('nearest neighbors') are identified by calculating distance of the instance under test from all training instances. The most commonly used distance measure is the *euclidean distance*. The class of the majority of the nearest neighbors is assigned to the instance under test. An interested reader can find more details about KNN in [8].

5.1.3 C4.5

Decision trees are usually used to map observations about an item to draw conclusions about the item's target value using some predictive model. C4.5 [13] builds a decision tree to classify elements. The criterion of selecting a decision node is the value of its information gain. The attribute which effectively splits the data into smaller subsets is selected as a decision node. The classifier repeats the same process on the subsets until all of them are evaluated. In the end, decision tree is pruned to optimize the classification.

5.1.4 JRip

Jrip develops a set of rules based upon RIPPER algorithm [16]. RIPPER, performs quite efficiently on large noisy datasets with hundreds of thousands of examples. It caters for missing attributes, numerical variables and multiple classes. The algorithm works by initially making a detection model composed of rules which are improved iteratively using different heuristic techniques. The constructed rule set is used to classify the test cases.

5.1.5 Support Vector Machine (C-SVM)

SVM [11] is a boundary based classifier that constructs hyper planes which splits the classes. These hyper planes define boundaries which are used for classification of entities. It uses iterative training algorithm to construct an optimum hyper plane to minimize the error function. The error function classifies SVM into four distinct groups. Classification SVM type 1 (C-SVM) is the most popular among all.

Now we focus our attention to evolutionary classifiers.

5.2 Evolutionary Classifiers

5.2.1 Fuzzy AdaBoost (Fuzzy-AB)

A fuzzy rule based classifier is defined by a fuzzy relationship that assigns each instance a degree of membership to each class; therefore, an instance may belong to multiple classes with different degrees of compatibility. The fuzzy classifiers are well suited for classification problems having imprecise ('fuzzy') boundaries. In the Fuzzy AdaBoost algorithm [9], boosting is used to combine multiple weak fuzzy hypotheses through a genetic iterative learning process to evolve a hybrid classifier that performs significantly better than any of the component hypothesis [9].

5.2.2 Genetic clASSIfier sySTem (GAssist-ADI)

Genetic clASSIfier sySTem (GAssist) [4] is an evolutionary classification algorithm belonging to the Pittsburgh style Genetic-Based Machine Learning paradigm. In Pittsburgh approach, each individual in the population represents a *'complete solution to the classification problem'* [4]. GAssist uses genetic algorithm to evolve the individuals – which represent rulesets – of the population. GAssist uses a fitness function based on the Minimum Description Length (MDL) principle to make optimal trade-off between complexity and accuracy of the rulesets. Generalization of the rulesets is improved by using the incremental learning with alternating strata (ILAS) windowing scheme. We have used the adaptive discretization intervals (ADI) rule representation because our dataset contains only real values. An interested reader can find the detailed description of GAssist algorithm in [4].

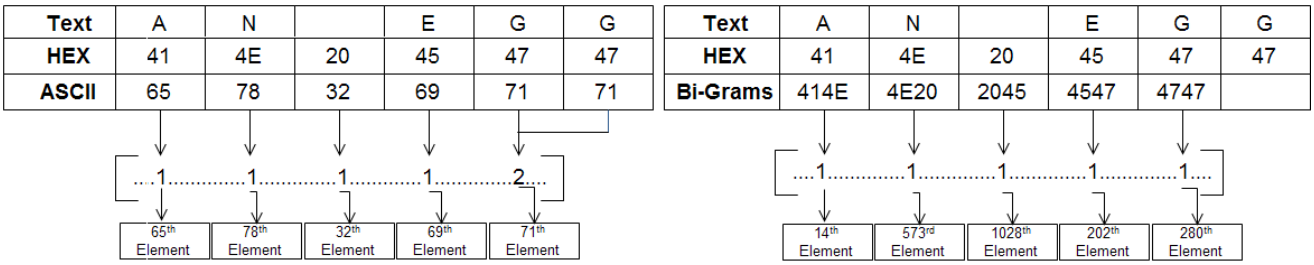

Figure 3: (a) Mapping of Byte frequency and (b) Octet Bigrams

5.2.3 *eXtended Classifier System (XCS)*

XCS [19, 20] is a Michigan style, rule-based, learning classifier system. In the Michigan approach, each individual represents a single rule and the whole population defines the complete ruleset. The initial rules are generated using a subset of the training data. New rules are periodically evolved using a niche genetic algorithm. The prediction accuracy of each rule (or individual) defines the fitness of that individual. The expected payoff values for all possible actions are stored in a prediction array. XCS algorithm has the ability to solve real world problems through evolution of accurate generalizations and real value representations [21].

5.2.4 *sUpervised Classifier System (UCS)*

UCS [12] is also a Michigan style, rule based, learning classifier system. It has been derived from XCS. The basic operation of UCS algorithm is similar to the XCS algorithm. However, there are some differences which separate UCS from XCS. The major difference between the two algorithms is their learning schemes (reinforcement scheme in XCS and supervised scheme in UCS). UCS does not maintain a prediction array. Since the fitness is defined in terms of the classification accuracy, UCS guarantees evolution of accurate classifiers. UCS can learn and evolve rules online, hence it is well suited for realtime online systems.

6. EXPERIMENTS AND RESULTS

We now discuss the results of our empirical evaluations. The basic philosophy followed during the evaluation is not to bias our studies towards a particular learning paradigm (or a classifier). We want that the classifiers build spam and ham models using as small as possible number of messages. We define the performance of a classifier using three metrics: (1) detection rate, (2) false alarm rate, and (3) testing time.

We define detection rate DR as:

$$DR = \frac{TP}{TP + FN} \tag{1}$$

Similarly, false alarm rate (FAR) is defined as:

$$FAR = \frac{FP}{FP + TN} \tag{2}$$

Since we are dealing with a two class problem; therefore, in both definitions the symbols used are defined as:

True positive (TP) is the number of spam messages detected as spam;

False positive (FP) is the number of benign messages detected as spam;

True Negative (TN) is the number of benign messages detected as benign;

False Negative (FN) is the number of spam messages detected as benign.

We define *Testing time* as the time taken at the access layer to recognize a message as benign or spam.

We have used KEEL [3] software for classification of messages. The parameters of all classifiers are tweaked to ensure that the classifier is used with the best configuration to make it relevant to compare their performance metrics. We train the classifier on the above-mentioned three types of features' set. The testing phase is done in two phases as well: (1) testing on seen messages, and (2) testing on unseen messages by gradually increasing their number. We now discuss the results of our empirical evaluations.

6.1 Results

We have tabulated the performance of different classifiers in three tables for each type of features' set. Table 2 is for octet bigrams only, Table 3 is for distribution of octets only and Table 4 is for the combined features' set.

6.1.1 *Training accuracy*

As expected, once we present the classifiers with the same messages that are used in training, almost every classifier – with the exception of Naive Bayes – provides 100% detection rate with 0% false alarm rate. (Compare the first two columns of all tables.)

6.1.2 *Testing Accuracy*

We now present different classifiers with unseen messages. We analyze their scalability as the number of unseen messages are gradually increased from 500 to 3000.

The first insight that we want to build is the effect on performance metrics with an increase in the number of unseen SMS (to be tested), once we are using octet bigrams. If we look at Table 2, we see that (generally speaking) the detection rate of all classifiers is significantly degraded. With the exception of UCS, GAssist-ADI, and Naive Bayes, the detection rates of other classifiers degrade by 35% to 50%. This is logical because it is not possible to learn in 500 messages, all octet bigrams of spam and ham. Moreover, a number of octet bigrams would be present in both types of messages that would further confuse a classifier. Therefore, decision tree based classifiers – C4.5 and JRip – do not achieve good performance (detection rate is degraded to 50%) because of

Table 2: Detection Rate and False Alarm Rate (in %) of selected algorithms for octet bigrams

	Training		Testing									
Amount of Messages	**500 msgs**		**500 msgs**		**1000 msgs**		**1500 msgs**		**3000 msgs**		**Average**	
	DR	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**
Fuzzy-AdaBoost	100	0	90	0	72	0	68	0.04	50	0.08	70	0.03
GAssist-ADI	100	0	92	0	85	0	80	0.001	70	0.01	81	0.003
UCS	100	0	92	0	89	0	82	0	76	0	84	0
XCS	100	0	91	0	76	0	70	0.02	60	0.04	74.25	0.015
C4.5	100	0	94	0	78	0	65	0	50	0.05	71.75	0.013
JRip	100	0	92	0	73	0	60	0.1	52	0.2	69.25	0.075
IBk	100	0	92	0	80	0	74	0.08	60	0.08	76.5	0.04
C-SVM	100	0	91	0	79	0	70	0	65	0.04	76.25	0.01
Naïve Bayes	99	0	95	0	85	0	80	0	75	0	84	0

Table 3: Detection Rate and False Alarm Rate (in %) of selected algorithms for frequency distribution of bytes

	Training		Testing									
Amount of Messages	**500 msgs**		**500 msgs**		**1000 msgs**		**1500 msgs**		**3000 msgs**		**Average**	
	DR	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**
Fuzzy-AdaBoost	100	0	94	0	84	0	70	0	50	0	74.5	0
GAssist-ADI	100	0	96	0	86	0	80	0	75	0	84.25	0
UCS	100	0	95	0	88	0	82	0	78	0	85.75	0
XCS	100	0	95	0	75	0	68	0	60	0	74.5	0
C4.5	100	0	96	0	89	0	65	0	60	0	77.5	0
JRip	100	0	93	0	80	0	60	0	50	0	70.75	0
IBk	100	0	99	0	85	0	80	0	72	0	84	0
C-SVM	100	0	96	0	84	0	70	0	62	0	78	0
Naïve Bayes	100	0	95	0	83	0	80	0	78	0	84	0

Table 4: Detection Rate and False Alarm Rate (in %) of selected algorithms for the merged features' set

	Training		Testing									
Amount of Messages	**500 msgs**		**500 msgs**		**1000 msgs**		**1500 msgs**		**3000 msgs**		**Average**	
	DR	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**	**DR**	**FAR**
Fuzzy-AdaBoost	100	0	96	0	82	0	78	0	60	0.04	79	0.01
GAssist-ADI	100	0	97	0	90	0	82	0	76	0	86.25	0
UCS	100	0	97	0	94	0	92	0	89	0	93	0
XCS	100	0	97	0	87	0	76	0	67	0.01	81.75	0.003
C4.5	100	0	97	0	82	0	76	0	60	0.05	78.75	0.013
JRip	100	0	94	0	80	0	70	0	60	0.08	76	0.02
IBk	100	0	96	0	85	0	82	0	70	0.08	83.25	0.02
C-SVM	100	0	96	0	86	0	72	0	64	0.04	79.5	0.01
Naïve Bayes	99.8	0	96	0	90	0	82	0	70	0	84.5	0

their inability to learn accurate rules from a small number of messages. In comparison, UCS has the ability to evolve rules online; therefore, it is able to create accurate and generalized rules.

It is obvious from Table 3 that (generally speaking) the detection rate of classifier is 4-5 % better using the frequency distribution of bytes in messages as a feature. But, we also see the same trend: the detection rate degrades once we increase the number of (to be tested) unseen messages. Again, UCS and Naive bayes achieve the best detection rate of 78% followed by XCS and IBk at 75% and 72% respectively. It is interesting to note Jrip achieves the worst detection rate of 50% and hence it is unable to generate accurate rules compared with other classifiers even using distribution of bytes.

Now the next logical design option is to utilize both features in training and testing and study the performance of classifiers. The intuition is that once both features are utilized, the classifiers could extract more knowledge; and as a consequence, the performance would improve. It is obvious from Table 4 that using both features has resulted in improving the detection rate of all classifiers. The detection rate of UCS (89%) has significantly improved by 13 percentage points when only octet bigrams are used and by 11 percentage points when only frequency distribution of bytes are used. This shows that UCS has the ability to transform additional knowledge into creating accurate rules. The performance gap has also widened because the next best classifier is GAssist-ADI that is at 76%. It is also interesting to note that the accuracy of Naive Bayes has degraded compared with the previous cases because the assumption that both features are totally independent of one another does not hold in this case.

To conclude, we have short listed UCS to be used as the detection engine in our mobile spam filtering system.

6.1.3 Testing time.

As expected, most of machine learning classifiers (except IBk) take only a fraction of a second to classify an incoming message. In comparison, most of evolutionary learning classifiers take 3-4 seconds per message to classify a SMS. In comparison, UCS only takes 1.2 seconds to classify an incoming message. We believe that a delay of 1 second in displaying a SMS in the inbox at the application layer would not significantly degrade the user experience.

Table 5: Average Testing Time (in seconds)

	Octet Bigram	*Byte Frequency*	*Merged Features*
Algorithms	**Testing**	**Testing**	**Testing**
Fuzzy-AdaBoost	2	1	4
GAssist-ADI	6	2	3.5
UCS	1.2	<1	1.2
XCS	3	1.5	4
C4.5, C-SVM, NB, JRip	<1	<1	<1
IBk	2	<1	2.5

We ignore the training time because it is assumed that training could happen offline and hence its time is not relevant.

6.1.4 Memory requirements.

Recall that mobile phones have limited memory and processing power; therefore, it is important to know the memory needed to store the features' set. We need to store three vectors each for both features: one for benign SMS dataset, one for spam SMS dataset, and one for (to be) tested SMS. In case of octet bigrams, one feature vector consists of $39 \times 39 = 1521$ elements. Each element is of 4 bytes; therefore, one feature vector needs $6,084$ bytes and three vectors together need $18,252$ bytes. Similarly, the frequency distribution vector consists of 256 elements each of 4 bytes. As a result, one vector consists of 1024 bytes and three vectors need 3072 bytes. To conclude, the total memory needed is $18252+3072 = 21324$ bytes. This is acceptable because even the low cost mobile phones have more than 1 Mbyte of RAM.

7. CONCLUSION

In this paper, we have proposed a SMS spam detection system that silently filters spam SMS at the access layer of a mobile phone. We have collected a real world SMS dataset that consists of more than 2000 spam and 4000 ham messages. We utilize two octet based features: (1) octet bigrams, and (2) frequency distribution of bytes; as a result, our filtering system does not depend on semantics of language because it works with the hexadecimal notation. Later, we do a comparative study to compare the performance of evolutionary and non-evolutionary classifiers on our `SMS` dataset. The results of our experiment suggest to use UCS that provides a detection rate of 93% with 0% false alarm rate. Moreover, it classifies a message in approximately one second. This time is acceptable for its deployment on real mobile phones. In future, we want to implement our system on a Symbian based Nokia mobile phone (N97) to validate its performance in real world scenarios. This will be the subject of forthcoming publications.

8. REFERENCES

[1] Case-study: Ironport helps a nationwide carrier stop wireless threats. `http://www.ironport.com/pdf/ironport_case_study_wireless.pdf`. Accessed April 2010.

[2] Sophos: 200 million cellphone users hit by SMS spam tidalwave in china. `http://www.sophos.com/pressoffice/news/articles/2008/03/china_sms.html`, March 2008. Accessed March 2011.

[3] J. Alcala-Fdez et al. KEEL: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing*, 13(3), 2008.

[4] J. Bacardit. Pittsburgh Genetics-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time. *PhD disertation*, 2004.

[5] H. Bergstt'en. Comprehensive study gives insight into mobile spam, ericsson. `http://www.ericsson.com/ericsson/corpinfo/publications/telecomreport/archive/2005/june/mobile_spam.shtml`, June 2005. Accessed May 2010.

[6] J. Brown, B. Shipman, and R. Vetter. SMS: The Short Message Service. *IEEE,Computers*, 40(12):106–110, 2007.

[7] G. V. Cormack, J. M. G. Hidalgo, and E. P. Sánz. Feature engineering for mobile (SMS) spam filtering. In *ACM SIGIR conference on Research and development in information retrieval*, pages 871–871, 2007.

[8] B. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press, 1991.

[9] M. del Jesus et al. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. *Fuzzy Systems, IEEE Trans. on*, 12(3):296–308, 2004.

[10] S. Dixit, S. Gupta, and C. Ravishankar. LOHIT: An online detection and control system for cellular SMS spam. In *IASTED International Conference Communication, Network, and Information Security*, November 2005.

[11] P. Lewicki et al. *Statistics: Methods and Applications*. StatSoft, Inc., 2006.

[12] E. Mansilla et al. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evo. Comp.*, 11(3):209–238, 2003.

[13] J. Quinlan. Improved Use of Continuous Attributes in C4.5. *JAIR*, 4:77–90, 1996.

[14] Z. Rafique and M. Farooq. SMS spam detection by operating on byte-level distributions using hidden markov models (HMMs). In *Virus Bulletin International Conference, VB*, September 2010.

[15] I. Rish. An empirical study of the naive Bayes classifier. In *Proc. IJCAI-01 Workshop on Empirical Methods in AI*, volume 335, 2001.

[16] M. Sasaki and K. Kita. Rule-based text categorization using hierarchical categories. In *Systems, Man, and Cybernetics IEEE International Conference*, volume 3, pages 2827–2830, October 1998.

[17] F. Sebastiani. Machine learning in automated text categorization. In *ACM Computing Surveys (CSUR)*, pages 1–47, March 2002.

[18] T. Tompkins and D. Handley. Giving e-mail back to the users: Using digital signatures to solve the spam problem. *First Monday*, 8(9), 2003.

[19] S. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[20] S. Wilson. Generalization in the XCS classifier system. In *Proc. Genetic Programming*, pages 665–674. Morgan Kaufmann, 1998.

[21] S. Wilson. Get Real! XCS with Continuous-Valued Inputs. *LNCS*, pages 209–222, 2000.