The Curse of 140 Characters: Evaluating the Efficacy of SMS Spam Detection on Android

Akshay Narayan
School of Computing
National University of Singapore
anarayan@comp.nus.edu.sg

Prateek Saxena
School of Computing
National University of Singapore
prateeks@comp.nus.edu.sg

ABSTRACT

Many applications are available on Android market place for SMS spam filtering. In this paper, we conduct a detailed study of the methods used in spam filtering in these applications by reverse engineering them. Our study has three parts. First, we perform empirical tests to valuate accuracy and precision of these apps. Second, we test if we can use email spam classifiers on short text messages effectively. Empirical test results show that these email spam classifiers do not yield optimal accuracy (like they do on emails) when used with SMS data. Finally, in this work we develop a two-level stacked classifier for short text messages and demonstrate the improvement in accuracy over traditional Bayesian email spam filters. Our experimental results show that spam filtering precision and accuracy of nearly 98% (which is comparable with those of email classifiers) can be obtained using the stacked classifier we develop.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Spam; I.2.7 [**Natural Language Processing**]: Text analysis

General Terms

Security; Spam Filtering; Android App Analysis; Bayesian Classifier; Support Vector Machines

Keywords

Spam Filtering; Message Tagging; Naive Bayes; SVM; Android App Analysis

1. INTRODUCTION

Short message service (SMS) is a text messaging service provided by telecommunication operators as a part of the implementation of SS7 protocol stack [20]. These text messages, sent using standard cellphones are limited to 140 octets (70-160 characters long based on the encoding used) owing to the limitation imposed by constraints of the signaling protocol (Mobile Application Part of SS7 protocol). We call these as *short text messages* in this paper. Short text messages are a convenient medium of asynchronous communication for the masses. On the downside, cellphones are becoming the largest target of spammers using short text message spam via SMS. SMS spam has been used to infect Android smart phones and induct them to a spam botnet via malicious links sent from a compromised device [28].

One of the reasons SMS spam has become widespread is that SMS text messaging is becoming cost effective to spammers. The popularity of SMS has led to messaging charges dropping below US$0.001 in markets like China, and even free of charge in others [4]. The cost per SMS and profit trend with respect to SMS advertising is noted in the GSMA Spam Reporting Service [9]. The low-cost and high-profit of SMS is one of the reasons for spammers resorting to SMS spam. Additionally, various web-based solutions for bulk messaging make spammers' job trivial [21]. The success of SMS spam is directly related to the large user base of SMS and similar *pseudo SMS apps* which are text messaging applications like Whatsapp, Viber and so on. The report by GSMA Spam Reporting Service provides a perspective about the consequences and varieties of SMS spam messages [9].

Why SMS spam filtering using apps? SMS spam filtering can be achieved at two points in a cellular network: (i) at the cellular network operator end, and (ii) at end systems (which are users' mobile devices). There are several reasons why SMS spam filtering is difficult on cellular network operator end. First, though there are guidelines for spam filtering by FCC, cellular network operators cannot enforce many of these guidelines as users may have opted to receive bulk messages from various sources or the origin of a spam message is not a mobile device [5, 7]. In addition, SMS spam filtering at cellular network operator end is difficult as it is expensive to obtain SMS content at that point [33]. Secondly, presence of various pseudo SMS applications is another hurdle for spam filtering at cellular network operator end. Users treat these applications as SMS applications. These applications use packet data connection and hence can bypass the SMS spam filters, if any, deployed at the cellular network operator end and such messages reach end users. These applications are equally vulnerable to short text spam. In order to cover the entire spectrum of SMS-type applications, it becomes necessary to deploy app-based spam filters on end user devices.

SPSM'13, November 8, 2013, Berlin, Germany.
Copyright 2013 ACM 978-1-4503-2491-5/13/11 ...$15.00.
http://dx.doi.org/10.1145/2516760.2516772 .

Our Goals: There are multiple applications available on app markets for smart phones which offer SMS spam filtering. Presently there are no benchmarks for users/researchers to evaluate the efficacy of these applications. Motivated by this need, we perform a detailed evaluation of the SMS spam filtering capability of the top twelve popular SMS Spam filtering Android applications (based on the number of downloads and installs) found in the Google play store [1]. We reverse engineer these applications to determine the method used for spam filtering. We find that all of these apps perform SMS spam classification based on rule based filtering, filtering blacklisted numbers, filtering message from an unknown number, keyword/url based filtering and so on [11]. Further, our evaluation of these apps on our published dataset shows that most of the existing SMS spam filtering apps available for Android platform have accuracy less than 50%. We present the details of this finding in Section 2.

Our study raises a natural question, "Could we reuse" techniques from email spam detection on short messages? We study this question from a conceptual standpoint and validate the same with empirical evaluation. Presently, there are many useful and effective spam filtering mechanisms available for emails. However, the same is not true for SMS. Among the machine learning methods used commercially in email spam detection, there is little systematic evaluation of the practical efficacy of one over the other. It is necessary to evaluate the effectiveness of email spam filtering techniques for SMS text messages so that they can be widely used by handset manufacturers to provide a reliable spam filtering service to end users. In this work we detail the conceptual challenges involved in spam filtering for short text messages which makes it difficult to reuse classical spam detection techniques from email settings to SMS setting. We measure the real-world performance of these email spam filters on the same dataset.

Finally, we explore the possibility of improving spam filtering accuracy for short messages using multiple levels of classification. In this direction, we propose an effective two-level stacked classifier which out performs Bayesian spam filters on SMS messages. Using this two-level classifier we could achieve spam classification precision of about 98% which is comparable to commercial email spam classifiers available today.

It is known from previous works that that probabilistic graphical models based algorithms (like Bayesian filters) can be effectively (in terms of power and memory consumption) deployed on resource constrained mobile devices [27]. Based on this premise, we develop a stacked classifier and evaluate its efficacy in filtering SMS spam messages. While building the spam filter application, we can embed the training set withing the classifier binary or train the classifier offline to be used with the spam classifier.

Scope. There are two broad categories of spam filtering techniques: based on (i) SMS message content, and (ii) non-content features. In this work, we focus on content-based SMS filtering on mobile devices using app-based SMS spam filters. Non-content based filtering techniques are largely employed by cellular network operators. Several works have studied non-content based spam filtering which employ identifying gray phone space (associated with data devices such as data cards) message size, number of messages sent daily / weekly, number of unique recipients, clustering co-efficients of target numbers and IP-traffic for the purpose of spam filtering [24, 26, 33]. Finally, previous works point out that non-content features are generally less effective than content based SMS spam filtering [29], but we don't evaluate this comparative strength in this work.

Contributions. In this work, we claim the following contributions.

- We benchmark the efficacy of the top twelve SMS spam filtering apps available on Google Play Store. We reverse engineer these applications and perform code analysis to understand the method of spam filtering used.
- We provide a comparative study of using email spam filters on SMS spam, develop a comparison of machine learning methods for short text message spam detection using SMS corpus.
- Propose a two-level classifier for short text message spam filtering and evaluate its advantage over the conventional classifiers which can be used for SMS spam filtering.
- Evaluate the challenges involved in using machine learning based email spam filtering mechanism to train the SMS spam classifier.

The rest of the paper is organized as follows. We present a detailed conceptual and empirical analysis of various Android applications for spam filtering in Section 2. We present the challenges of reusing email spam detection methods on SMS data and the empirical performance of traditional Bayesian classifiers on SMS data in Section 3. We describe the implementation of our two level classifier for improved spam classification on SMS data along with its empirical performance in Section 4. Prior works related to our contributions are detailed in Section 5. We present our conclusions in Section 6.

2. ANALYSIS OF ANDROID APPS

In this section we benchmark the efficacy of top twelve popular SMS spam filtering applications available in Google Play Store and describe our findings. We select the applications based on number of downloads as reported by Google Play Store. Eleven of the applications have over 50k downloads, with three of them having over 1000k downloads (as of 31 July 2013). We omit applications having less than 10k downloads from our evaluation. The applications we tested are free versions, but we find that paid/premium versions of these applications have even lesser number of downloads. We reverse engineer these applications to the best of our ability to obtain a conceptual understanding of the method used for spam filtering. For those applications which could not be reverse engineered, we perform detailed black-box testing to identify the spam filtering method. Further, we perform empirical analysis of these applications using a subset of the dataset (see Section 3.2) to measure their spam classification accuracy.

2.1 Conceptual Analysis

Methodology. We gathered the SMS spam filtering apps from Google Play Store and perform two types of analysis. First, in white-box analysis, we reverse engineered these applications in order to determine the content based filtering

	Number Based Filtering			Content Based Filtering			
Application	Spam*	Ham**	User Specified Blacklist	Rule Based Classification	Regex/ Strcmp	Machine Learning	# downloads
AVG	Yes	Yes	Yes	Yes	–	–	> 50000k
SmsBlocker ‡	Yes	Yes	Yes	–	–	–	1000k-5000k
Quickheal ‡	Yes	Yes	Yes	–	–	–	1000k-5000k
AntiSpamSMS	Yes	Yes	Yes	Yes	Yes	No	100k-500k
Numbercop	No	No †	Yes	No	No	No	100k-500k
Private Box	Yes	Yes	Yes	No	No	No	100k-500k
SMS Filter	Yes	Yes	Yes	No	No	No	100k-500k
The Call	No	Yes	Yes	No	Yes	No	100k-500k
Postman	Yes	Yes	Yes	No	Yes	No	50k-100k
smsBlocker	Yes	No †	Yes	Yes	No	No	50k-100k
SMS Spam Blocker	Yes	Yes	Yes	Yes	No	No	50k-100k
SpamBlocker	Yes	Yes	Yes	Yes	Yes	No	10k-50k

*Any message marked as spam when sender is not in contact list

**Any message marked as ham when sender is in contact list

Table 1: Observed Behavior of SMS Spam Filtering Applications Obtained by Reverse Engineering

method used. We use a third party application (*dex2jar* [10]) to decompile `dex` files and a `jar` decompiler (*Java Decompiler* [3]) to retrieve source code from the `jar` archive. Three of the applications are obfuscated and we make use of *dex-tools* which is a component of *dex2jar* to retrieve the source code. We performed manual inspection of the so obtained source code to identify the method of spam filtering employed. Based on code inspection, we conclude that eight out of the twelve apps perform number based spam filtering, i.e., all messages sent from unknown contacts are marked as spam and all messages sent from known contacts are marked as ham. Two applications exhibit different behavior and we elaborate on their behavior in Section 2.2. Two applications could not be reverse engineered (indicated by ‡ in Table 1). To identify the spam filtering mechanism in these four apps, we perform a second black-box analysis of the applications to enable various options provided for SMS spam filtering (Section 2.2). We use Android emulator to run these tests. Using black-box analysis we tabulate the features provided to end users for spam control. We use a set of fifty randomly drawn inputs from our published dataset to observe the behavior of these applications and infer the spam filtering method used. Specifically, we observe whether the input was classified as spam or ham.

White-box analysis results. Based on our analysis, we find that apps use two broad categories of techniques for spam filtering: (i) Number based filtering, and (ii) Content based filtering (refer Table 1). Eleven of the twelve applications provide number based filtering as an option that can be toggled in the application settings UI. There is no such UI option for turning on/off content based filtering.

All the applications we analyze provide spam filtering based on number blacklisting. Some of the applications like *AntiSpamSMS* also provide spam filtering based on blacklisted URLs. *AntiSpamSMS* retrieves a list of known spammers from its centralized server on the Internet and filters messages based on it. This option is provided to users via the application settings. Ten of the twelve applications filter out SMS from unknown senders as spam while allowing all messages from known senders (i.e. senders present in contact list) to reach the message inbox. However, two applications (*SMS Blocker* and *Numbercop*) filtered few spam messages from known senders as spam.

Nearly 50% of the applications provide some form of rule based classification. Five of the applications use rules pertaining to presence of certain words, blacklisting people in the contact list and so on. Five applications (≈40%) perform some form of regular expression matching or string comparison to filter spam messages. To the extent of our analysis of decompiled applications, none of these use any form of machine learning but rather use rule-based content matching. Therefore, we expect the accuracy of classification to be low. Table 1 lists out the spam filtering features provided by various apps we tested. We present our findings on accuracy and precision in the next subsection.

Findings: Based on our analysis we summarize our findings below.

- All applications provide number blacklisting facility.
- About 85% applications block any message from an unknown sender.
- Less than 25% applications use reg-ex matching or string matching.
- Some applications filter spam messages correctly from known senders, however this behavior is not consistent across all messages in the spam dataset we used.
- To the extent of our manual audit by reverse engineering, none of the applications use machine learning techniques for spam classification.

2.2 Empirical Results

In addition to reverse engineering, we perform a black-box empirical test of the twelve applications. Black-box analysis allows us to determine performance of candidate applications in terms of precision and accuracy. To evaluate the applications, we select a subset of the dataset described in Section 3. We measure the precision and accuracy of the applications using this input set [1].

[1] Since the Android emulator was slow and timed out after about ten messages sent from the telnet interface, we limit the input to randomly selected fifty messages per application.

Application	TP	FP	FN	TN	Precision	Accuracy
SmsBlocker	0	0	25	25	0.00	0.50
SpamBlocker	6	12	19	13	0.33	0.38
AVG Antivirus	0	0	25	25	0.00	0.50
Postman[1]	0	0	25	25	0.00	0.50
SmsBlocker[1]	0	0	25	25	0.00	0.50
SMS Spam Blocker	0	0	25	25	0.00	0.50
Quickheal[2]	25	0	0	25	1.00	1.00
Quickheal[3]	0	25	25	0	0.00	0.00
AntiSpam SMS[4]	7	4	18	21	0.64	0.56

TP: True Positive, FP: False Positive, FN: False Negative, TN: True Negative

[1] Sender in contacts list [2] With blacklisting turned on
[3] With blacklisting turned off [4] Sender in contacts, blacklist on

Table 2: Spam Filtering Performance of Apps

Experimental results. Here we present the empirical results of spam filtering performance of various applications that we tested. We installed these applications on Android emulator and tested them with the same test set we used to compare the performance of various classifiers listed in the next section. We selected a few applications to enable various options available in them. For some applications such as *Quickheal*, performance dropped when used with default options (black listing turned off). *AntiSpamSMS* for instance provided us with a precision of 64% when its blacklist look up was turned on and when we enabled the option to treat messages sent by senders in contact as non-spam. Applications like *SpamBlocker* and *AntiSpamSMS* performed some spam filtering owing to the reg-ex matching/string comparison present in their spam filtering logic. When we turned on the blacklist look up option of *Quickheal* we obtained 100% accuracy. It would be interesting to learn the spam filtering method used in this case (we like to point out that Quickheal could not be reverse engineered). Five of the applications listed have accuracy of 50%. However, first, we like to point out from our observation that this 50% accuracy was obtained as either all of spam SMS sent from unknown contacts were marked as spam or all messages sent from known contacts were marked as ham which is logically incorrect. Second, there is evidence of no machine learning even in the obfuscated applications since the true positives are near zero. These results are captured in Table 2.

3. REUSING EMAIL SPAM DETECTION METHODS

Given that existing SMS spam detectors on Android are ineffective, we ask “is it possible to reuse spam filtering techniques from email spam detection?”. In this section we describe (i) challenges involved in working with short text messages, (ii) efficacy of using email spam detection algorithms on short text messages, and (iii) present our empirical results of using email spam classifiers on short text messages.

3.1 Conceptual Analysis

Before getting into the details of using email classifiers on SMS data, it is worthwhile to make a note of the challenges related to processing short messages in general and spam classification of short messages in particular. We make the following observations.

- Running classifiers on email datasets give sufficiently accurate results owing to the fact that there are more words to help the classifier. However in case of SMS, the character limit of 140 octets make it less words to work with. Hence such classifiers are weaker. Character limit does not strictly hold for the pseudo SMS applications mentioned. However, users treat these applications similar to SMS. As a result, we expect the content pattern in the data exchanged via these applications to be similar to real SMS text.

- Due to the character limit in SMS, acronyms and abbreviations find extensive use. The language variability in SMS text acronyms and abbreviations make the representation sparse for a classifier. This further weakens the classifier.

- Email spam classifiers generally remove all n-grams (for some small n) as they do not contribute to useful features for classification. However owing to the length restrictions, bi- and tri-grams are used extensively in short messages and these cannot be eliminated directly. Such elimination may lead to having a null vector input to the classifier. For example, this message `ok! bye! tc!` will result in a null vector when bi-grams and tri-grams are eliminated. Hence a special treatment of bi-grams and tri-grams is needed in case of short messages.

- We find that in SMS spam filtering, using the entire text message as is for classification leads to a large number of false positives. Considering the above discussions, we need carefully select which words to use as features and which words to exclude. There is no published guideline on selecting the feature set. We touch upon the effects of using bi-grams versus not using bi-grams on classification in Table 3.

- Treating same words in different letter cases as different features yield different results in classification. We cover this aspect in brief in Section 3.2.

- Many email classifiers use message probability thresholds to differentiate between ham and spam. There is no literature which specify these values and they are determined empirically. We find that, choosing the thresholds used for email spam classification leads to a drastic drop in classification accuracy when used with short message data.

3.2 Empirical Measurements

In this subsection we describe the method followed in testing email spam filtering algorithms on SMS data and finally we present our results and findings.

Reference Dataset. For empirical evaluation we combined the tagged SMS datasets published by Almeida et.al., [12], Nuruzzaman et.al., [27] and Delany et.al., [18]. Since all these datasets were constructed using NUS SMS corpus [16] and Grumbletext [2] they have many duplicates. To benefit our evaluation, we removed the duplicates to the best of our ability. This dataset consists of about 1450 tagged messages (730 ham and 721 spam) which we use for training the classifiers. The test set consists of about 700 messages. This re-purposed dataset is made available publicly here: `https://github.com/okkhoy/SpamSMSData`.

Evaluation Target. We use the reference dataset described above to evaluate the efficacy of reusing email spam filtering methods on short message data. In our evaluation, we considered three different types of tokenization mechanisms.

- Simple splitting of messages and using all the available words–naive approach.
- Standard email like tokenization–removing stop words and using standard lemmatizer like Wordnet lemmatizer available in NLTK library [8].
- Handling bi-grams and tri-grams while tokenization.

A discussion on using these three different variations in tokenization is presented later in this section.

Methodology. Most of the messages in the dataset are raw texts. We need to perform some pre-processing before using them in our experiments. As a part of *pre-processing* we remove all non-ASCII special characters which appear as a part text messages. These special characters mostly represent emoticons. One of the corpus had additional meta data like time of transmission and reception, obfuscated phone numbers etc., these additional data were removed as a part of pre-processing as they are not relevant in comparing various classifiers.

The next step in implementing a learning algorithm is *tokenization*. Tokenization is the process of dividing the message into semantically meaningful segments. Here we split the message into a set of words. Tokenization produces a set of words in the message, with no repeated tokens. This is a useful first level filtering because, having repeated words do not add any value to the classifier.

As a part of our contributions in this paper, we consider the usefulness of running Bayesian email spam filter on the short message dataset. Before delving into the details, we point out that these Bayesian classifiers use message probability thresholds to differentiate between ham and spam. For the purpose of evaluation we used SpamBayes [6], a popular spam filter application to classify a subset of the short message dataset in hand. Commercially deployed email filters like SpamBayes [6] use ham and spam thresholds as 0.2 and 0.8 respectively. There is no literature suggesting the use of these threshold values and we believe these are obtained empirically. When we used the same thresholds on short messages, the precision and recall values obtained were low. We anticipated this behavior as described earlier in this section.

Results with Original Thresholds. We begin the discussion of experimental results to determine the efficacy of using email spam filters on SMS with the effect of tokenization on the outcome of classification. Accuracy of the Bayesian classifier (used as an example) based on different tokenization approaches described above is listed in Table 3. It has to be noted that better tokenization yields better accuracy for each of the classifier discussed above.

As expected, using all available words weakened the classifier and the accuracy dropped to about 84%. Email like tokenization (stemming and lemmatization using Wordnet) yields better accuracy (around 87%) than using all available words. Treating bi- and tri-grams specially yielded us better results. However, the improvement is not very significant on this dataset, we got about 7% increase in the accuracy. We tested the performance of the classifier by retaining the word case versus ignoring the word case. We notice that ignoring word case gives better accuracy as against the former (90% vs 88%). This is because, treating same words in different case as different tokens result in reducing the word probability and hence, has a cascading effect on message probability.

Tokenization Method	TP	FP	FN	TN	Accuracy
Word Split	284	20	95	315	0.839
Email Like	276	28	67	343	0.867
N-Grams	284	20	20	390	0.944
Ignore Case	283	21	50	360	0.901
Retain case	278	26	60	350	0.880

TP: True Positive, FP: False Positive, FN: False Negative

Table 3: Accuracy Obtained by Different Tokenization Mechanisms

Results with Tweaked Thresholds. While discussing the usefulness of using email like spam filters on SMS dataset, we mentioned that commercially deployed email filters use ham and spam thresholds. In order to make up for the lack of words to work with, we need to tweak these thresholds. Since there is no available reference for the typical values of these thresholds, we empirically determine that by using threshold values of $(0.4, 0.6)$ for ham and spam respectively, yields good precision in classification. Our observations are tabulated in Table 4.

(H, S)	TP	FP	FN	TN	Precision	Recall	Accuracy
(0.1, 0.9)	37	267	207	203	0.122	0.152	0.336
(0.2, 0.8)	85	219	130	280	0.280	0.395	0.511
(0.3, 0.7)	231	73	68	342	0.760	0.773	0.803
(0.4, 0.6)	268	36	50	360	0.882	0.843	0.880

TP: True Positive, FP: False Positive
TN: True Negative, FN: False Negative

Table 4: Performance of Bayesian Classifier with Different Thresholds

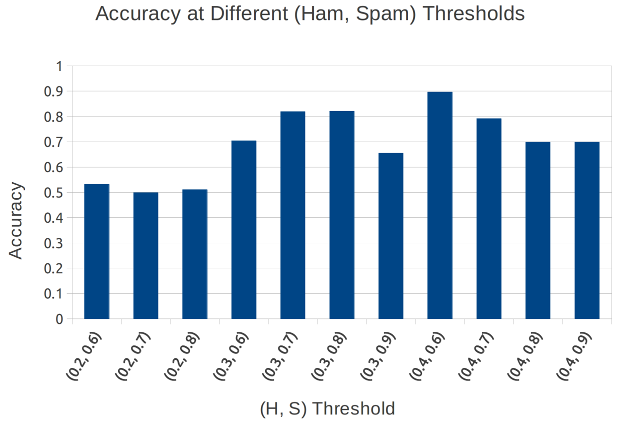


Figure 1: Accuracy of Classifier at Different Ham & Spam Thresholds

We plot the classifier accuracy at different (ham, spam) threshold combinations to visualize the classifier performance.

This plot is shown in Figure 1. We note that using thresholds (0.2, 0.8) from SpamBayes email classifier the classifier accuracy is around 51%. Reducing the thresholds to (0.4, 0.6) gives us better classification accuracy. Though reducing ham and spam thresholds improves performance of email spam filters on SMS data, it essentially means that the classification is based on weaker constraints.

Findings. Based on the discussions above, we summarize our findings here.

- Bayesian email spam filters cannot be used with SMS data as is.
- Tweaking ham and spam thresholds of email spam filters result in better performance on short message data. However, this performance on SMS data is not comparable with email classification accuracy.
- Reducing these thresholds means classification is based on weaker constraints.
- Special attention is needed on the tokenization mechanism used.

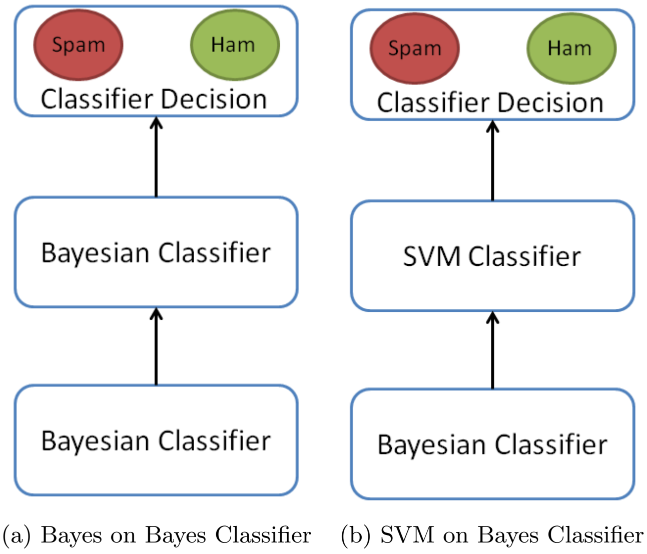
Threat to validity. First, our evaluation uses a single dataset consisting of ≈2100 messages. Extending such evaluation with additional data would help ascertain the validity of our findings. Second, the spam messages in the dataset were collected globally via Grumbletext while the ham messages were primarily obtained from the NUS SMS corpus. There is a variability in the language used in the two sets of messages. Ham messages are subject to language deviation owing to the vocabulary and style of communication used in Singapore. However, our work primarily concern comparison of the various classifiers using this dataset. Since the language bias is applicable to all the experiments described in this section, we expect that the effect of this bias would be nullified in the comparisons presented in this work.

4. IMPROVED SPAM CLASSIFIER FOR SHORT TEXT MESSAGES

As presented in Section 3, we find that using email spam filters as is on SMS data yields sub-optimal results. In this section, we explore the possibility of improving the spam classification accuracy for short text messages. Towards this direction we describe the implementation of our two-level classifier and present its performance in comparison with email spam filters. As can be seen from empirical results, we obtain better accuracy and precision using the two-level classifier.

4.1 Two-Level Stacked Classifier Architecture

In text classification, clustering followed by classification is quite common [35, 19]. The variation we implement in this work uses two levels of classifiers. Similar technique of stacking of classifiers has been used sparsely in email spam filtering earlier. Specifically, Sakkis et al., use a master-slave architecture for stacking k-nearest neighbor classifier over Naive Bayes [30]. However, our implementation of stacking is different. The first level Bayesian classifier records a subset of words whose individual probability is higher than a threshold (say 65%) and this set is further input to another round of classification. The second level classifier is invoked only when the output of first level contains words whose individual spam probability is higher than a threshold. The second level classifier complements the first level to achieve better classification accuracy. We implement two versions of the two-level stacked classifier: (i) using Bayesian classifiers at both levels, and (ii) using Bayesian classifier in first level and SVM in the second. Conceptual diagrams of these two versions of the two-level classifier are as shown in Figure 2.



(a) Bayes on Bayes Classifier (b) SVM on Bayes Classifier

Figure 2: Different Variations of the Two-level Classifier

We record the spam probabilities of individual words and message probabilities. We assume a hitherto unseen word to be equally likely being spam or ham. Hence, we associate a probability of 0.5 to a new word encountered. The nature of Bayesian approach corrects this probability value as learning progresses and more words are seen. The probability of each word is recorded in the form of a dictionary for future use during the test phase.

In case of SVM classifier, we need to represent messages as a feature vectors. The effect of tokenization is clearly evident in SVM. If the number of features are not reduced, the classifier becomes “heavy weight” and performance degradation sets in. Hence removal of stop words and stemming are useful in case of SVM classifier. We represent messages as a binary vectors. We generate a word-frequency table for spam and ham messages separately. These tables comprises of the mapping `word` ⇒ `frequency` as needed by the classifier.

In the learning phase, we set up a model for spam classification and train the classifier. For comparison purpose, we train the Bayesian classifier and SVM classifier separately. Results of each of the classifier are tabulated individually. The findings are reported in the next subsection.

4.2 Empirical Results

For empirical evaluation, we use the dataset described in Section 3.2. Our training set comprise of 730 ham messages and 721 spam messages. Test set consists of about 700 messages. We now consider the impact of using a two-level classifier for short message classification. In Table 5, we present a comparison of precision and accuracy of Bayesian filter with our two-level classifier. Again for different ham and spam message probability thresholds we run the two-level classifier described in previous section on the short text message dataset in hand. We observe that a two-level classifier running Bayesian filters at both levels yield better precision for each combination of (ham,spam) threshold. For instance, using the empirically determined threshold of $(0.4, 0.6)$ and running the stacked classifier gives an improved precision of about 91% as against a simple Bayesian classifier which gives only 88% precision. Using SVM in the second level provides better results as can be seen in Table 6.

	Naive Bayes		2-Level Classifier	
(H, S)	Precision	Accuracy	Precision	Accuracy
(0.1, 0.9)	0.122	0.336	0.280	0.508
(0.2, 0.8)	0.280	0.511	0.441	0.606
(0.3, 0.7)	0.760	0.803	0.799	0.819
(0.4, 0.6)	0.882	0.880	0.914	0.894

Table 5: Comparison of Bayesian Classifier with Two-Level Stacked Classifier on ≈700 messages

Table 6 provides an insight into the performance of various classifiers considered in our work. We use an SVM classifier based on LIBSVM [15]. The data representation used in this SVM classifier is similar to the SVM email classifier by Shelly [31]. It is well known that SVM outperforms Bayesian classifiers and the results we obtained reinforce this belief. We see that the precision of the two-level classifier using SVM in second level is marginally better than using Bayesian classifier (98.4% of SVM vs 94.7% of Bayesian). To see the impact of using additional levels instead of two levels in the classifier, we used SVM over the results obtained by two-level Bayesian classifier. Though the precision of SVM on Bayesian (98.4%) is almost same as compared to the three level classifier (98.7%), the recall or specificity is better in the latter. Also the overall accuracy of this setup (SVM on two-level Bayesian) is higher (98.8%) when compared to other classifiers (Table 7).

A comparison of accuracies of various classifiers is listed in Table 7. At this point we would like to bring in the comparison of using Naive Bayes classifier on short text messages and email datasets. Our experiments reveal that the best precision obtained by using Naive Bayes classifier to classify short text messages is 88–90%. However, literature shows that researchers are able to achieve near 100% precision using Naive Bayes classifiers on email datasets [13, 25]. Using stacked classifier on short messages we could achieve classification precision comparable to the email classifiers (98.4% of SVM on Bayes vs 98.9% presented in [25]). This comparison asserts the need for using a multilevel stacked classifier on short text messages.

5. RELATED WORK

In this section, we discuss few prevalent SMS spam filtering mechanisms and contrast our work with existing approaches. Among the prevalent methods, Naive Bayes seems to be a favorite among researchers. There have been various efforts like [27, 36, 22] using Naive Bayes content based classifiers to filter SMS spam messages. Nuruzzaman et. al., [27] were able to achieve accuracy over 90% using PGM based spam filters. They propose a method to use a small training set and improve the classifier accuracy using subsequent messages to train the classifier. Support vector machines (SVM) is the second most popular classification algorithm used in prior works in SMS spam filtering [34, 32, 36]. Some researchers have also used Hidden Markov Models for SMS spam filtering [29], however with lesser accuracy compared to Naive Bayes or SVM (89% accuracy of HMM vs close 92% accuracy of SVM). It has been noted that using the email "bag-of-words" approach to SMS spam filtering does not work really well [17]. Hence the challenge here is to see how the Naive Bayes and SVM based email spam classifiers perform on SMS spam messages. Delany et. al., [18] provide a topic based classification of SMS messages to various categories like dating, competitions etc. In their work they use results provided by various prior research work such as k-nearest neighbor, SVM, first order HMM on independent datasets to draw the conclusion.

Grier et. al., [23] perform a detailed study of Twitter spam in their work. Problems associated with blacklisting a sender or a URL which is carried in SMS spam messages are similar to those found in Twitter spam. Hence many of the spam apps which perform filtering based on blacklists/white-lists fail to curtail the impact of SMS spam. SMS spam is generally targeted towards resource constrained devices whereas Twitter streams originate from web based applications and are also available for viewing on the regular computing devices like PCs. It would be interesting to compare spam analysis techniques used in micro-blogging with SMS in future work.

Chakradeo et. al., [14] use multiple categories of data to analyze malware in Android market. We can look at the possibility of using such techniques to use both content-based and non-content based data for SMS spam filtering in future work.

Sakkis et al., [30] use a master-slave architecture for stacking k-nearest neighbor classifier over Naive Bayes to classify emails. However, in this work we implement a different form of stacking. We use the second level classifier to complement the first classifier to achieve better classification accuracy.

Our work is the first detailed study of popular Android apps for SMS spam filtering and insights based on our experiments show that these apps are ineffective in dealing with spam. We also explore the possibility of using Bayesian email classifiers on SMS and further, propose a two-level classifier to achieve improved spam filtering performance.

6. CONCLUSION

In this work we performed analysis of Android applications, email filters on SMS data and proposed an approach to improve spam classification accuracy for short text messages. Based on our experiments and empirical results we summarize the following.

- We provided a detailed study of the top twelve SMS spam filtering apps available on Google Play Store and conclude that most of these are ineffective on SMS spam messages.

- We drew a comparative study of using email spam filters on SMS spam, and highlight conceptual challenges in re-purposing traditional email spam filters for short text messages.

Classifier	True Positive	False Positive	False Negative	True Negative	Precision	Recall	F-Measure
Bayesian	268	36	10	400	0.882	0.964	0.921
SVM	292	12	26	384	0.961	0.918	0.939
SVM on Bayes	299	5	9	401	0.984	0.971	0.977
Bayes on Bayes	288	16	3	407	0.947	0.990	0.968
SVM & Bayes on Bayes	300	4	3	407	0.987	0.990	0.988

Table 6: Performance Comparison of Various Classifiers

Classifier	True Positive	False Positive	False Negative	True Negative	Accuracy
Bayesian	268	36	10	400	0.936
SVM	292	12	26	384	0.947
SVM on Bayes	299	5	9	401	0.980
Bayes on Bayes	288	16	3	407	0.973
SVM & Bayes on Bayes	300	4	3	407	0.990

Table 7: Accuracy of Various Classifiers

- Proposed a two-level classifier for short text message spam filtering and evaluate its advantage over the conventional classifiers which can be used for SMS spam filtering. We observed a good improvement in spam classification precision using the two-level classifier.
- Enumerated the challenges involved in using machine learning based email spam filtering mechanism to train the SMS spam classifier.

7. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers and our shepherd Patrick Traynor for their valuable feedback which helped bring the paper to its current form. This research is partially supported by research grant R-252-000-495-133 from Ministry of Education, Singapore. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the Ministry of Education, Singapore.

8. REFERENCES

[1] Google Play Store. [Online]. Available at: `http://goo.gl/y6wHO`.

[2] Grumbletext. [Online]. Available at: `http://goo.gl/10vdoa`.

[3] Java decompiler. [Online]. Available at: `http://goo.gl/Ryy94w`.

[4] Mobile operators brace for global surge in mobile messaging abuse. Online. Available at: `http://goo.gl/5an9q`.

[5] Spam: Unwanted Text Messages and Email. [Online]. Available at: `http://goo.gl/Kn1J9t`.

[6] SpamBayes. Online. Available at: `http://goo.gl/VMBw6`.

[7] Text Message (SMS) Spam and Caller ID Spoofing Spam. [Online]. Available at: `http://goo.gl/9QaRJs`.

[8] Wordnet lemmatizer. Online. Available at: `http://goo.gl/FqqSi`.

[9] SMS Spam and Mobile Messaging Attacks Introduction, Trends and Examples. [Online], Jan. 2011. Available at: `http://goo.gl/xSpCo`.

[10] dex2jar. [Online], October 2012. Available at: `http://goo.gl/bHYSO3`.

[11] Sophos mobile security. [Online], April 2013. Available at: `http://goo.gl/01YIu`.

[12] T. Almeida, J. Hidalgo, and A. Yamakami. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262, 2011.

[13] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *CoRR*, cs.CL/0006013, 2000.

[14] S. Chakradeo, B. Reaves, P. Traynor, and W. Enck. Mast: triage for market-scale mobile malware analysis. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, WiSec '13, pages 13–24, New York, NY, USA, 2013. ACM. doi: 10.1145/2462096.2462100.

[15] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://goo.gl/YtCU`.

[16] T. Chen and M.-Y. Kan. Creating a live, public short message service corpus: the nus sms corpus. *Language Resources and Evaluation*, pages 1–37, 2012. doi: 10.1007/s10579-012-9197-9.

[17] G. Cormack, J. Hidalgo, and E. Sánz. Feature engineering for mobile(SMS) spam filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, volume 23, pages 871–872, 2007.

[18] S. J. Delany, M. Buckley, and D. Greene. SMS spam filtering: Methods and data. *Expert Systems with Applications*, 39(10):9899–9908, 2012. doi: 10.1016/j.eswa.2012.02.053.

[19] I. S. Dhillon, S. Mallela, and R. Kumar. Enhanced word clustering for hierarchical text classification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*,

KDD '02, pages 191–200, New York, NY, USA, 2002. ACM. doi: 10.1145/775047.775076.

[20] L. Dryburgh and J. Hewett. Signaling System No. 7 (SS7/C7): Protocol. *Architecture, and Services*, 2004.

[21] W. Enck, P. Traynor, P. McDaniel, and T. La Porta. Exploiting open functionality in sms-capable cellular networks. In *Proceedings of the 12th ACM conference on Computer and communications security*, CCS '05, pages 393–404, New York, NY, USA, 2005. ACM. doi: 10.1145/1102120.1102171.

[22] J. M. Gómez Hidalgo, G. C. Bringas, E. P. Sánz, and F. C. García. Content based SMS spam filtering. In *Proceedings of the 2006 ACM symposium on Document engineering*, DocEng '06, pages 107–114, New York, NY, USA, 2006. ACM. doi: 10.1145/1166160.1166191.

[23] C. Grier, K. Thomas, V. Paxson, and M. Zhang. @spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 27–37. ACM, 2010.

[24] N. Jiang, Y. Jin, A. Skudlark, and Z.-L. Zhang. Greystar: Fast and accurate detection of sms spam numbers in large cellular networks using grey phone space. In *Proceedings of 22nd USENIX Security Symposium*, page pp, Washington, D.C., USA, 2013.

[25] X. Ma, Y. Shen, J. Chen, and G. Xue. Combining naive bayes and tri-gram language model for spam filtering. In Y. Wang and T. Li, editors, *Knowledge Engineering and Management*, volume 123 of *Advances in Intelligent and Soft Computing*, pages 509–520. Springer Berlin Heidelberg, 2011.

[26] I. Murynets and R. Piqueras Jover. Crime scene investigation: Sms spam data analysis. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, pages 441–452, New York, NY, USA, 2012. ACM. doi: 10.1145/2398776.2398822.

[27] M. Nuruzzaman, C. Lee, and D. Choi. Independent and personal sms spam filtering. In *2011 IEEE 11th International Conference on Computer and Information Technology (CIT)*, pages 429–435, Sept 2011. doi: 10.1109/CIT.2011.23.

[28] P. Paganini. Android malware for SMS spam botnet. [Online], December 2012. Available at: `http://goo.gl/I18bW`.

[29] M. Rafique and M. Farooq. SMS SPAM detection by operating on byte-level distributions using hidden markov models (HMMs). In *Proceedings of the 20th virus bulletin international conference*, 2010.

[30] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. *CoRR*, cs.CL/0106040, 2001.

[31] N. Shelly. SVM Spam Filter. Online. Available at: `http://goo.gl/rwHqP`.

[32] Y. Xiang, M. Chowdhury, and S. Ali. Filtering mobile spam by support vector machine. In *CSITeA'04: Third International Conference on Computer Sciences, Software Engineering, Information Technology, E-Business and Applications*, pages 1–4. International Society for Computers and Their Applications (ISCA), 2004.

[33] Q. Xu, E. Xiang, Q. Yang, J. Du, and J. Zhong. Sms spam detection using noncontent features. *Intelligent Systems, IEEE*, 27(6):44–51, 2012. doi: 10.1109/MIS.2012.3.

[34] Q. Xu, E. Xiang, Q. Yang, J. Du, and J. Zhong. SMS Spam Detection Using Noncontent Features. *IEEE Intelligent Systems,*, 27(6):44–51, Nov 2012. doi: 10.1109/MIS.2012.3.

[35] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 619–626, New York, NY, USA, 2008. ACM. doi: 10.1145/1390334.1390440.

[36] K. Yadav, P. Kumaraguru, A. Goyal, A. Gupta, and V. Naik. SMSAssassin: crowdsourcing driven mobile-based system for SMS spam filtering. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, HotMobile '11, pages 1–6, New York, NY, USA, 2011. ACM. doi: 10.1145/2184489.2184491.