Classifying Spam Emails using Text and Readability Features

Rushdi Shams and Robert E. Mercer
Department of Computer Science
University of Western Ontario
London ON N6A 5B7, Canada
Email: rshams@csd.uwo.ca, mercer@csd.uwo.ca

***Abstract*—Supervised machine learning methods for classifying spam emails are long-established. Most of these methods use either header-based or content-based features. Spammers, however, can bypass these methods easily—especially the ones that deal with header features. In this paper, we report a novel spam classification method that uses features based on email content-language and readability combined with the previously used content-based task features. The features are extracted from four benchmark datasets viz. CSDMC2010, SpamAssassin, LingSpam, and Enron-Spam. We use five well-known algorithms to induce our spam classifiers: Random Forest (RF), BAGGING, ADABOOSTM1, Support Vector Machine (SVM), and Naïve Bayes (NB). We evaluate the classifier performances and find that BAGGING performs the best. Moreover, its performance surpasses that of a number of state-of-the-art methods proposed in previous studies. Although applied only to English language emails, the results indicate that our method may be an excellent means to classify spam emails in other languages, as well.**

***Keywords*—Spam classification, machine-learning application, anti-spam filter, performance evaluation, text categorization, text features, readability features, feature importance.**

I. Introduction

Since the publishing of *Spam!* in 1998 [1], the menacing onslaught of spam emails has grown exponentially. According to a recent survey, the number of spam emails sent out in March 2013 is about 100 billion [2]. This phenomenal quantity is 98% more than that from the end of the previous quarter. Among the drastic effects of spam emails are loss of individual productivity and financial loss of organizations. Anti-spammers, therefore, are putting forward efforts to prevent this potential threat to today's Internet.

Many studies agree that spam emails have attributable patterns [3][4]. Recognizing this, spammers are constantly introducing new techniques to obfuscate these recognizable patterns to fend off supervised anti-spam filters. Therefore, spam filtering is, and is likely to remain, an interesting machine learning application. Typically, an email exhibits two types of features: (i) header features and (ii) content-language features. Using either set of features to detect spam email has its pros and cons. For instance, Zhang *et al.* [5] empirically showed that the performances of a handful of machine learning algorithms are not satisfactory with text features. Lai and Tsai [6] found similar results, too—according to their study, header features have a lower *total cost ratio* than text features. Another typical argument to support the use of header features is that they are language independent (see, for example, [7] and [8]). However, Metsis *et al.* [9] exploited a language independent, content-based feature called *term frequency* (TF). Using this feature with different variants of the Naïve Bayes algorithm, they found results that are better than those found by a number of header-based methods. Likewise, studies are continuously reporting improved results with email-text features (see Prabhakar and Basavaraju [10], and Ma *et al.* [11])—most of which perform better than header features.

The proposed method utilizes text features that are long-established such as frequency of *spam words* and HTML tags as well as some that are new. The novelty of our work is that we introduce language-centric features such as grammar and spell errors, use of function words, presence of verbs and alpha-numerics, TF·IDF, and inverse sentence frequency. In addition, we use features related to message readability (e.g., reading difficulty indexes, complex and simple word frequency, document length, and word length). The features are extracted from four standard email datasets—CSDMC2010, SpamAssassin, LingSpam, and Enron-Spam. The features are then used by five well-known learning algorithms—Random Forest (RF), BAGGING, ADABOOSTM1, Support Vector Machine (SVM), and Naïve Bayes (NB)—to induce binary classifiers. From our extensive experiment, we find that the addition of text and readability features significantly improves classifier performance compared to that found with the commonplace features alone. Another notable finding is that the classifier induced by BAGGING performs the best on all of the datasets. In addition, its performance surpasses that of a number of state-of-the-art methods proposed in previous studies. Like Metsis *et al.* [9], our features are language independent. Whereas we derive the results solely from English emails, the proposed approach may be an excellent means to classify spam emails in any language.

The next section details the features used in this research. Following that, we describe the learning algorithms, performance evaluation measures, and datasets. In Section VI, we report the experimental findings, and in Section VII we provide some discussion of these results, especially as compared with previous filtering methods. Finally, Section VIII concludes the paper.

II. Feature Selection

Each email, in our experiment, is represented as $(\vec{x}, y)$, where $\vec{x} \in \mathbb{R}^n$ is a vector of n attributes and $y \in \{spam, ham\}$ is the label of the email. In our study, we explored 40 attributes to classify emails and therefore, $n = 40$. We have grouped

Groups	Features			
Traditional Features	Spam Words	HTML Anchors	HTML Non-anchors	HTML Tags
Text Features	Alpha-numeric Words	Verbs	Function Words	TF·ISF
	TF·IDF	Grammar Errors	Spelling Errors	Language Errors
Readability Features	FI	FRES	SMOG	FORCAST
	FKRI	Simple Word FI	Inverse FI	Complex Words
	Simple Words	Document Length	Word Length	$\text{TF·IDF}_{complex}$
	TF·IDF_{simple}			

TABLE I: The list of features used to classify spams in our experiment.

these features into three subgroups: (i) traditional features, (ii) text features, and (iii) readability features. The list of features used in our experiment is shown in Table I. The notation used in this section is "X_i : *name* = *feature value*", where X_i is used to label the graphs later in the paper, *name* is a descriptive name, and *feature value* indicates the feature value calculation. Some features are also normalized. These normalizations are discussed where warranted.

A. Traditional Features

Below are descriptions of the four typical features for spam classification that are used in our experiment.

1) Dictionary-based Features: The first feature in this group is the frequency of spam words. The selection of this feature is inspired by the interesting findings of Graham [12]. He showed that merely looking for the word *click* in the messages can detect 79.7% of spam emails in a dataset with only 1.2% false positives. To exploit this feature, we have developed a dictionary comprising 381 spam words[1] accumulated from various anti-spamming blogs. We treated each message as a bag-of-words and counted the frequency of spam words in them as follows:

X_1 : *Spam Words* = #*Spam Words*.

2) HTML *Features:* The frequency of HTML tags in emails is a common means to classify spams. We have subdivided this feature into three features: (i) frequency of anchor tags (i.e., number of close-ended tags <a> and), (ii) frequency of tags that are not anchors, $anchor'$ (e.g., <p> or
), and (iii) total HTML tags in the emails (e.g., sum of (i) and (ii)). To identify HTML tags in the emails, we used a Java HTML parser called *jsoup*[2]. Each feature is normalized by the length of the email, *N*, which is the number of sentences in the message. The detailed HTML feature calculations are:

$$X_2 : anchor = \frac{\#anchor\ tags}{N},$$

$$X_3 : anchor' = \frac{\#anchor\ tags'}{N}, \text{ and}$$

$$X_4 : total\ \text{HTML} = \frac{\#anchor\ tags + \#anchor\ tags'}{N}.$$

B. Text Features

These novel (except for *Verbs*) features focus on various aspects of the email text. These features into divided into two subgroups: (i) Word-level Features and (ii) Error Features.

1) Word-level Features: First, we considered the frequency of alpha-numeric words in the emails. We found reasonable evidence to select this feature during the development of the spam word dictionary (See Section II-A)—many of the dictionary entries are alpha-numerics. The apparent reasons for this include that spammers often advertise menacing websites by replacing literals of the legitimate website with numerics and vice versa. This text feature is calculated as follows:

X_5 : *Alpha-numeric Words* = #*Alpha-numeric Words*.

In their study, Orasan and Krishnamurthy [13] showed that a *verb* Part-of-Speech (POS) feature can be a strong identifier of junk emails; nonetheless, very few works have followed up. We are, however, interested in this particular POS feature. We used the Stanford POS Tagger[3] to tag the POS of each word in the emails and considered the frequency of verbs in each:

X_6 : *Verbs* = #*Verbs*.

Third, we used the frequency of function words[4] in the emails as a feature. As the definition of function words varies from task to task, we define them as follows: function words are very frequent non-content-bearing words such as articles, prepositions, adverbs, etc. We developed a *stoplist function* to count the frequency of function words in the emails:

X_7 : *Function Words* = #*Function Words*.

Fourth, we included another novel feature: TF·ISF, a simple summation of the product of TF and ISF, where the prior is the frequency of a term t in a message and the latter is its inverse sentence frequency. TF of term t can be calculated as follows:

$\text{TF}_t = 1 + log(frequency_t)$, if $frequency_t > 0$, 0 otherwise,

while its ISF is:

$\text{ISF}_t = log\ \frac{N}{\text{SF}_t}$,

where N is the message length and SF_t is the number of sentences with term t. Inverse sentence frequency is a relative measure of whether the term is common or rare in a single message. The TF·ISF of a message is calculated as follows:

X_8 : $\text{TF·ISF} = \sum_t \text{TF}_t \times \text{ISF}_t$, *for all* t *in the message*.

Our next feature in this group is TF·IDF which is similar to TF·ISF except that the inverse document frequency, IDF, measures whether a given term t is common or rare in an entire dataset. The IDF of a term t in the message can be found as follows:

$\text{IDF}_t = log\ \frac{D}{\text{DF}_t}$,

[1] http://cogenglab.csd.uwo.ca/sentinel/spam-term-list.html
[2] http://jsoup.org/download
[3] http://nlp.stanford.edu/software/tagger.shtml
[4] http://cogenglab.csd.uwo.ca/sentinel/function-word-list.html

where D is the total number of messages in the dataset and DF_t is the number of messages containing t. The TF·IDF feature is then calculated as follows:

X_9 : TF·IDF= $\sum_t \text{TF}_t \times \text{IDF}_t$.

The feature is then normalized by taking its square root.

2) Error Features: The next three features are concerned with the grammar and spelling errors present in the message. For each message, we counted the frequency of grammar and spelling errors using a Java API called LanguageTool[5]. By summing up these two errors, we introduced another feature named *Language Errors*:

X_{10} : *Grammar Errors* = $\#N_{ge}$,

X_{11} : *Spelling Errors* = $\#N_{se}$, and

X_{12} : *Language Errors* = $\#N_{te}$,

where N_{ge} is the number of sentences with grammar errors, N_{se} is the number of sentences with spelling errors and N_{te} denotes the total number of sentences with language-based errors (i.e., $N_{te} = N_{ge} + N_{se}$). This group of features, to the best of our knowledge, is novel for spam classification. It is to be noted that for training emails, this feature has been normalized respectively for hams and spams; for the testing emails for which the class label is unknown, traditional attribute normalization is performed. Surprisingly, we get unsatisfactory results if we use traditional attribute normalization for the training emails.

C. Readability Features

The second group of novel features that we use are readability-based features. Readability deals with the difficulty of reading a sentence, a paragraph, or a document. Two important parameters to calculate readability are simple and complex words. Simple words are those that have at most two syllables while complex words contain three or more syllables. Based on these two factors and others, five scores, among many, are used as yardsticks to assess the readability of text. In this section, we divided readability features further into two subgroups: (i) Score-based Features and (ii) Frequency-based Features.

1) Score-based Features: We measured the readability of each message in the datasets using the five aforementioned scoring methods. First, we used the *Fog Index (FI)* [14], which is the most popular score to measure readability. The scores of each message can be found as follows:

$$X_{13} : \textit{FI} = 0.4 \times \left(\left(\frac{\sum \textit{Words}}{N}\right) + 100 \times \left(\frac{\sum \textit{Complex Words}}{\sum \textit{Words}}\right)\right).$$

Second, we used the *Flesch Reading Ease Score (FRES)* [15], one of the oldest readability scores. The *FRES* of any given message can be found as follows:

$$X_{14} : \textit{FRES} = 206.835 - 1.015 \times \left(\frac{\sum \textit{Words}}{N}\right) - 84.6 \times \left(\frac{\sum \textit{Syllables}}{\sum \textit{Words}}\right).$$

The *SMOG index*, when first published, was anticipated as a proper substitute for *FI* due to its accuracy and ease of use [16]. It is the third readability score used as a feature:

$$X_{15} : \textit{SMOG Index} = 1.043 \times \sqrt{30 \times \frac{\sum \textit{Complex Words}}{N}} + 3.1219.$$

The *FORCAST index* was originally formulated to assess the reading skills required by different military jobs without focusing on running narratives [17]. The index, unlike others, emphasizes the frequency of simple words. The following is the way to calculate the FORCAST index of a message:

$$X_{16} : \textit{FORCAST Index} = 20 - \frac{W}{10},$$

where W is the number of simple words in a 150-word sample of the text.

A second instalment of a readability index proposed by Flesch and further investigated and modified by Kincaid [18] is known as the *Flesch-Kincaid Readability Index (FKRI)*. This feature can be calculated as follows:

$$X_{17} : \textit{FKRI} = 0.39 \times \left(\frac{\sum \textit{Words}}{N}\right) + 11.8 \times \left(\frac{\sum \textit{Syllables}}{\sum \textit{Words}}\right) - 15.59.$$

In addition, we have modified *FI* in two different ways to give two additional features. We chose to modify *FI* only, because empirical outcomes showed that these modifications of *FI* provide better results than those of other scores.

First, in X_{13}, in the place of complex words, we substituted the frequency of simple words:

$$X_{18} : \textit{FI}_{\textit{simple}} = 0.4 \times \left(\left(\frac{\sum \textit{Words}}{N}\right) + 100 \times \left(\frac{\sum \textit{Simple Words}}{\sum \textit{Words}}\right)\right).$$

Second, we took the arithmetic inverse of *FI* of the messages to get the feature *Inverse FI*:

$$X_{19} : \textit{Inverse FI} = \frac{1}{\textit{FI}}.$$

2) Frequency-based Features: We also considered the frequency of *Complex Words* and *Simple Words* in the messages as two more features:

X_{20} : *Complex Words* = $\#$*Complex Words*, and

X_{21} : *Simple Words* = $\#$*Simple Words*

Document Length, which has already been used in other features for normalizing, is also considered as a feature in our research. It simply denotes the number of sentences in a message:

X_{22} : *Document Length* = $\#$*Sentences*.

Word Length of a message is simply the average number of syllables per word. This feature is calculated as follows:

$$X_{23} : \textit{Word Length} = \frac{\#\textit{Syllables}}{\#\textit{Words}}.$$

Our last two features are the combination of TF·IDF, and frequency of simple and complex words. While X_9 deals with the TF·IDF of any term t, our second last feature $(\text{TF·IDF})_{\textit{complex}}$ deals with that of complex words. Finally, our last feature $(\text{TF·IDF})_{\textit{simple}}$ considers the TF·IDF of simple words. The formula to calculate these feature are given below:

X_{24} : $(\text{TF·IDF})_{\textit{complex}} = \sum \text{TF}_{\textit{complex}} \times \text{IDF}_{\textit{complex}}$, and

X_{25} : $(\text{TF·IDF})_{\textit{simple}} = \sum \text{TF}_{\textit{simple}} \times \text{IDF}_{\textit{simple}}$.

[5]http://www.languagetool.org/java-api/

Learning Algorithms	Parameters	
Random Forest (RF)	Maximum Depth: Unlimited	Number of Trees to be Generated: 10
	Random Seed: 1	
ADABOOSTM1	Number of Iterations: 10	Random Seed: 1
	Resampling: False	Weight Threshold: 100
BAGGING	Size of Bag (%): 100	Out of Bag Error: False
	Number of Iterations: 10	Random Seed: 1
Support Vector Machine (SVM)	SVM Type: C-SVC	Cost: 1.0
	Degree of Kernel: 3	EPS: 0.0010
	Gamma: 0.0	Kernel Type: Radial Basis
	Epsilon: 0.1	Probability Estimates: False
	Shrinking Heuristics: True	
Naïve Bayes (NB)	Use of Kernel Estimator: False	

TABLE II: Parameters of the learning algorithms used in this experiment.

As well, we used 14 more features that are calculated by excluding stopwords from the features X_1, X_5, X_6, X_8, $X_{13}-X_{21}$, and X_{23}. In the graphs these features have labels ending with a dot. It is noteworthy that the features X_2, X_3, and X_4 could not be extracted from the LingSpam and Enron-Spam datasets as the related information was removed from the messages by the dataset curators (refer to Table IV).

Since we analyzed the feature vectors for all the messages in the datasets, we have found that the distribution of the most of the feature values is not normal rather they exhibit either positive or negative skewness. The effect of this skewness was eliminated by using a logarithmic transformation of all the feature values. Once log-transformed, the distribution becomes normal. The newly found log values of any given feature are then normalized by the highest transformed value of that feature in the dataset; the resulting values are therefore in [0,1].

III. Learning Algorithms

We used the well-known learning algorithms Random Forest RF, Naïve Bayes (NB), Support Vector Machine (SVM), and two *meta-learning* algorithms viz. ADABOOSTM1 and BAGGING to induce binary classifiers. What follow are the reasons for choosing the algorithms in our experiment.

Among the learning algorithms, we have chosen Random Forest (RF) for three reasons. First, it has been used by a number of anti-spam filters because of its high spam-classification accuracy (see, for instance, [7] and [19]). Second, the algorithm runs efficiently on large data. Third and most importantly, the learning is fast—which is desirable for any *live* spam filter.

Our second algorithm, also an ensemble learning method, is called ADABOOSTM1. Although an ensemble method, ADABOOSTM1 is both simple and fast. The biggest advantage of this ensemble method is that it is less susceptible to training-data overfit. Last but not least, the algorithm has been reported to perform better than Naïve Bayes (NB) and Probabilistic TF·IDF for text categorization tasks [20]. The reasons to use *bagging* are primarily three. First, like ADABOOSTM1, BAGGING is simple and fast. Note that the speed of learning of this algorithm depends on the choice of the number of random samples of training data. Second, it is less susceptible to overfitting. Finally, it leads to improvements for unstable classification algorithms like trees [21]. In this experiment, RF has been chosen as the base algorithm for ADABOOSTM1 and BAGGING. Therefore, we name the classifiers generated by these two algorithms BOOSTED RF and BAGGED RF, respectively.

Support Vector Machine (SVM) is also a popular learning algorithm for spam detection. However, from the work of Zhang *et al.* [5], Lai and Tsai [6], Hu *et al.* [7], Qaroush *et al.* [19], and Ye *et al.* [22], it is evident that the performance of SVM is better with header features than with text features.

Like SVM, Naïve Bayes (NB) is a widely-used learning algorithm in the anti-spamming community. Benchmark anti-spam tools developed by Lai and Tsai [6], Hu *et al.* [7], Metsis *et al.*[9], and Qaroush *et al.* [19] use NB to generate classifiers because the algorithm is simple yet powerful enough to detect spams effectively. For instance, on many occasions, with simple features like TF·IDF, NB even outperformed quality learning algorithms like SVM.

The parameter setup for the learning algorithms used in this experiment is presented in Table II.

IV. Evaluation Measures

The evaluation of spam classification differs from many other classification tasks. A significant number of previous works rely on performance measures like precision, recall, F-score, and accuracy [7][8][10][11][19]. However, even a poor classifier can achieve overly-optimistic results on a skewed dataset and *appropriate* performances on such datasets are not reflected by the aforementioned measures. Because of the presence of skewness in the datasets, *cost-sensitive* measures like ham misclassification rate (FPR), spam misclassification rate (FNR) and total cost ratio (TCR) [23] [24] are becoming more popular. In addition, being a balanced measure that combines both FPR and FNR, *area under the ROC curve* (hereinafter, AUC) is also preferred by many [24]. Considering these facts, we choose to report seven evaluation measures—FPR, FNR, accuracy, precision, recall (or simply spam recall), F-score, and AUC. The measures are explained below.

All of the measures reported in this paper depend on the confusion matrix given in Table III. Precision is the fraction

		Actual	
		Spam	Ham
Prediction	Spam	$n_{s\to s}$	$n_{h\to s}$
	Ham	$n_{s\to h}$	$n_{h\to h}$

TABLE III: Confusion matrix for spam classification problem.

Dataset	Total Messages	Spam Rate	Text Pre-processed?	Year of Curation
CSDMC2010	4,327	31.85%	No	2010
SpamAssassin	6,046	31.36%	No	2002
LingSpam	2,893	16.63%	Yes	2000
Enron-1	5,172	29.00%	Yes	2006
Enron-2	5,857	25.54%		
Enron-3	5,512	27.21%		
Enron-4	6,000	75.00%		
Enron-5	5,175	71.01%		
Enron-6	6,000	75.00%		

TABLE IV: Brief description of the email datasets.

of spam predictions that are correct. Should the precision be bigger, the probability of misclassifying a legitimate mail as spam is smaller:

$$\textit{Precision} = \frac{n_{s\to s}}{n_{s\to s}+n_{h\to s}}.$$

Spam recall examines the fraction of spam emails being recognized. A bigger spam recall points out that the probability of misclassifying a spam as legitimate mail is smaller:

$$\textit{Recall} = \frac{n_{s\to s}}{n_{s\to s}+n_{s\to h}}.$$

F-score, simply, is the harmonic mean of precision and recall:

$$\textit{F-score} = \frac{2\times \textit{Precision}\times \textit{Recall}}{\textit{Precision}+\textit{Recall}}.$$

Accuracy, on the other hand, is the percentage of correctly identified spams and hams:

$$\textit{Accuracy} = \frac{n_{h\to h}+n_{s\to s}}{n_{h\to h}+n_{h\to s}+n_{s\to h}+n_{s\to s}}.$$

FPR denotes the fraction of all legitimate messages classified as spams:

$$\text{FPR} = \frac{n_{h\to s}}{n_{h\to s}+n_{h\to h}}.$$

In contrast, FNR is the fraction of all spams delivered to the user inbox:

$$\text{FNR} = \frac{n_{s\to h}}{n_{s\to h}+n_{s\to s}}.$$

Note that the lower the FPR and FNR, the better the performance. Considering the situation where users might accept spams to enter into their inbox but they do prefer their hams not to end up in the spam-traps, a higher FPR is more expensive than a higher FNR. To resolve this, we need a balance between FPR and FNR, which is the AUC in this case. The AUC is measured using the *Receiver Operating Characteristics (ROC)* curves. The ROC curve is a 2-D graph whose *Y-axis* is the *true positive rate* (which is indeed $1-$FNR) and *X-axis* is the FPR, and therefore depicts the trade-offs between the cost of $n_{s\to h}$ and $n_{h\to s}$.

V. Datasets

A number of standard email datasets are publicly available and widely used. In our experiment, we chose four of them: (i) CSDMC2010, (ii) SpamAssassin, (iii) LingSpam, and (iv) Enron-Spam. The reasons for choosing these datasets are manifold. Firstly, emails in these datasets have been sent out between 2000 and 2010. This provides an interesting test-bed that characterizes the change of language of emails spanning across a decade. Secondly, we are interested to evaluate our method with spam-skewed datasets. The reason behind this interest is because with ham-skewed datasets, even a poor classifier can achieve good FPR by classifying most of the emails, if not all, as hams (see, for example, Bratko *et al.* [24]). Therefore, we include Enron 4-6 in our experiment. Thirdly, we include LingSpam in our dataset because not only are its hams domain-specific but also its excerpts are of scholarly discussions on linguistics. As a result, we are expecting some features, *Error Features* (see Section II-B2) in particular, to be more useful for the LingSpam emails. Fourthly, we include datasets that are not explored by many (e.g., CSDMC2010 and Enron-Spam). Last but not least, the use of Enron-Spam gives us an opportunity to work with hams coming from a user's personal inbox.

A. Description

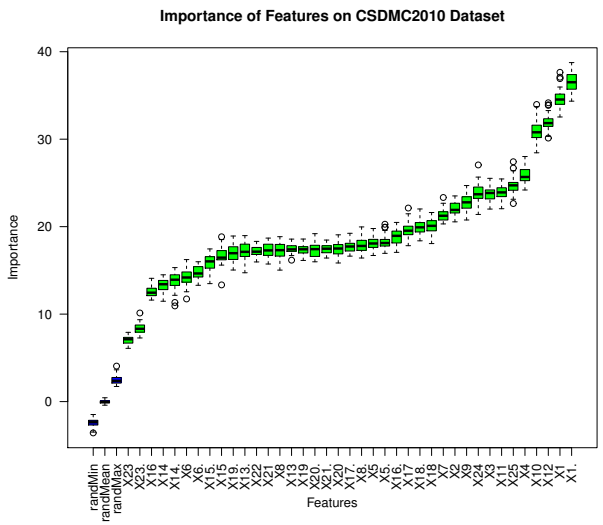
CSDMC2010[6], among the four, is the latest collection of emails. The spam rate of this dataset is reasonable—about 32%. Both hams and spams in this dataset are collected randomly (i.e., not from any particular inbox). CSDMC2010 is relatively new and except for the *ICONIP-2010* challenge participants, this dataset has not been explored by many. In contrast, SpamAssassin[7] is one of the most popular public datasets. Like CSDMC2010, the emails of SpamAssassin are also collected randomly. In addition, the spam rate of this dataset is almost equal to that of CSDMC2010. The LingSpam dataset [25] is both the smallest and oldest dataset used in this study. Moreover, its spam rate is smaller than the preceding datasets—only about 17%. It is, however, the odd one out of the four as the hams in this dataset are collected from the discussions of a linguistics forum; the spams, on the other hand, are collected randomly. Enron-Spam [9] is an email collection comprising six different datasets each of which contains ham messages from a single user of the Enron corpus. Of the six datasets, the bulk of the emails in Enron 1-3 are hams while the bulk of the emails in Enron 4-6 are spams. This collection is very different compared to the others as the hams of each dataset bear the characteristics of one individual. In this paper, we experiment on each of these six datasets and report the average performance. Table IV briefly outlines the datasets used in this experiment.

B. Pre-processing

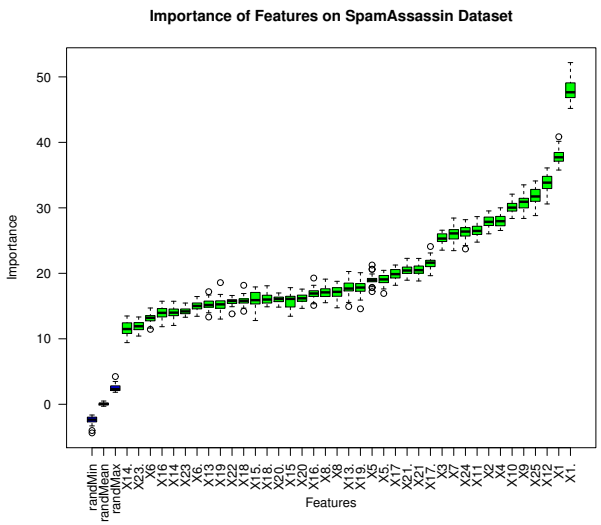
We noticed that spam emails have some features that can easily distinguish them from hams. Therefore, to provide a conservative estimate of our method's performance, we follow these pre-processing steps: First, symbols (like $ or ! signs) or spam words (like *porn*, *webcam*, or *lottery*) are excluded

[6]http://csmining.org/index.php/spam-email-datasets-.html

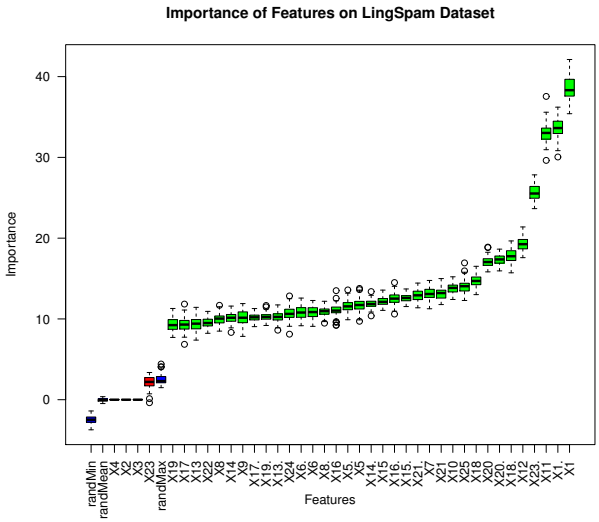
[7]http://spamassassin.apache.org/publiccorpus/



(a) The order of the features according to their importance for the CSDMC2010 dataset.



(b) The order of the features according to their importance for the SpamAssassin dataset.



(c) The order of the features according to their importance for the LingSpam dataset.

Fig. 1: Feature importance on three datasets according to the *Boruta* algorithm.

from the *subject* field of the emails. Second, *attachment* fields (if any) are excluded from the email texts. Third, non-ASCII characters present in the email texts not only are spam indicators but also the presence of such characters can change the readability of the entire message. As readability is one of our key features (see Section II-C), we remove the non-ASCII characters from the messages as well.

VI. Experimental Results

A. Feature Importance

In machine learning, the identification of features relevant for classification is important, so is the observation of results when features work in groups. This identification and observation give us a set of features that are important [26][27]. To measure feature importance, we use a *feature importance* measuring algorithm named *Boruta*[8] that uses a wrapper around Random Forest. The details of the *Boruta* algorithm are beyond the scope of this paper but can be found in the study by Kursa and Rudnicki [26]. We applied the algorithm on each of the datasets. Of note, due to limited space, the analysis of feature importance for the six Enron-Spam datasets are made available elsewhere[9].

Figure 1 exhibits the *boxplots* created with *Boruta* for the non-personalized email datasets. The *Y-axis* of each plot repre-

[8]http://cran.r-project.org/web/packages/Boruta/index.html
[9]http://cogenglab.csd.uwo.ca/sentinel/feature-importance.html

	Baseline	Baseline + Text Feature	Baseline + Readability Features	All
RF	0.060	0.034	0.038	0.040
BOOSTED RF	0.062	0.028	0.028	0.030
BAGGED RF	0.055	0.023	0.023	0.020
SVM	0.022	0.037	*0.023*	0.027
NB	0.031	0.065	0.093	0.101

(a) FPR of different groups of features combined with the baseline for the CSDMC2010 dataset.

	Baseline	Baseline + Text Feature	Baseline + Readability Features	All
RF	0.064	0.040	0.045	0.034
BOOSTED RF	0.071	0.029	0.034	0.026
BAGGED RF	0.061	0.026	0.028	0.022
SVM	0.055	*0.063*	*0.050*	*0.052*
NB	0.060	0.075	0.094	0.104

(b) FPR of different groups of features combined with the baseline for the SpamAssassin dataset.

	Baseline	Baseline + Text Feature	Baseline + Readability Features	All
RF	0.041	0.019	*0.023*	0.017
BOOSTED RF	0.044	0.018	0.016	0.017
BAGGED RF	0.040	0.011	0.015	0.009
SVM	0.0004	0.012	0.020	0.014
NB	0.063	0.091	0.214	0.218

(c) FPR of different groups of features combined with the baseline for the LingSpam dataset.

	Baseline	Baseline + Text Feature	Baseline + Readability Features	All
RF	0.404	0.164	0.245	0.174
BOOSTED RF	0.404	0.156	0.240	0.158
BAGGED RF	0.402	0.143	0.218	0.150
SVM	0.424	*0.371*	*0.442*	*0.416*
NB	0.408	0.304	0.376	0.336

(d) FPR of different groups of features combined with the baseline for the Enron-Spam dataset.

TABLE V: The effect of incrementing groups of features on spam classification for (a) CSDMC2010, (b) SpamAssassin, (c) LingSpam, and (d) Enron-Spam Datasets. The FPRs that are not statistically significant compared to the comparable baseline are written in *italics*.

sents feature importance (a measure specific to the algorithm) while the *X-axis* represents feature labels (see Section II). The features are sorted according to the ascending order of their importance in determining class labels. The key finding in the plots is that for all the datasets, the most important feature is *Spam Words* (X_1). Another interesting finding is that the *error features* (see Section II-B2) are close in importance. In addition, the TF·IDF of simple words (X_{25}) and the HTML features (X_2 and X_3, in particular) are important features for CSDMC2010 and SpamAssassin emails (Figures 1a and 1b). The relative position according to the importance of *Alpha-numeric Words* (X_5) is similar in all three datasets as is the *Verbs* feature (X_6). Interestingly, the *function word* feature (X_7) performed reasonably well—to the best of our knowledge not many consider this as a spam detection feature. Except TF·IDF of simple and complex words, most of the *readability features* are on the left side of the graph—seeming to be less important. Note that the significance of the readability features becomes evident when we, next, investigate the feature performance in groups.

We are also interested in reporting the importance of the features in groups. To do this, we used the traditional features (Section II-A1) as our baseline. These features are first used by the algorithms to classify the emails. Then, we added the text features and HTML features. As well, we added the readability features with the baseline. FPR is chosen as the measure to evaluate the classification. Figure 1 strongly suggests that the traditional features are important for spam classification. The interesting results shown in Table V, however, indicate the effect of combining the features. One common attribute of these results is that the ham misclassification rate decreases as the baseline features are combined with text and readability features. Another significant finding is that combining the readability features with the baseline may not produce the optimal result but combining these two groups of features with text features (i.e., using the entire feature pool) improves the ham misclassification rate. However, in that case the two exceptions are CSDMC2010 and Enron-Spam. Also from the results, the lowest FPR is achieved by the RFs—BAGGED RF, in particular. Although the FPRs for SVM and NB seem promising, their significantly low AUC indicates a possible overfit, especially for the CSDMC2010 and LingSpam datasets. The results for SVM and NB, therefore, are in doubt. The lowest FPR in Table V is achieved by BAGGED RF using all features on the most language homogeneous dataset, the LingSpam dataset.

Note that the FPRs that are not statistically significant

	FPR	FNR	Accuracy %	Precision	Recall	F-score	AUC
RF	0.040	0.092	94.338	0.914	0.908	0.911	0.980
BOOSTED RF	0.030	0.089	95.124	0.934	0.912	0.922	0.980
BAGGED RF	0.020	0.107	95.193	0.953	0.893	0.922	0.988
SVM	0.027	0.390	85.718	0.913	0.610	0.730	0.792
NB	0.101	0.396	80.471	0.737	0.604	0.662	0.855

(a) Evaluation measures of the full-featured spam classifiers for the CSDMC2010 dataset.

	FPR	FNR	Accuracy %	Precision	Recall	F-score	AUC
RF	0.034	0.093	94.707	0.923	0.907	0.915	0.979
BOOSTED RF	0.026	0.079	95.700	0.941	0.921	0.931	0.982
BAGGED RF	0.022	0.099	95.353	0.948	0.901	0.924	0.986
SVM	0.052	0.292	87.265	0.861	0.708	0.777	0.828
NB	0.104	0.558	75.373	0.660	0.443	0.529	0.847

(b) Evaluation measures of the full-featured spam classifiers for the SpamAssassin dataset.

	FPR	FNR	Accuracy %	Precision	Recall	F-score	AUC
RF	0.017	0.162	95.817	0.907	0.838	0.869	0.978
BOOSTED RF	0.017	0.162	95.886	0.910	0.838	0.871	0.977
BAGGED RF	0.009	0.193	95.956	0.944	0.807	0.868	0.986
SVM	0.014	0.341	93.156	0.907	0.659	0.760	0.822
NB	0.218	0.277	77.186	0.402	0.723	0.515	0.831

(c) Evaluation measures of the full-featured spam classifiers for the LingSpam dataset.

	FPR	FNR	Accuracy %	Precision	Recall	F-score	AUC
RF	0.174	0.072	91.681	0.887	0.927	0.906	0.960
BOOSTED RF	0.158	0.070	92.288	0.896	0.929	0.912	0.956
BAGGED RF	0.150	0.080	92.521	0.910	0.919	0.914	0.972
SVM	0.416	0.379	78.350	0.836	0.620	0.627	0.602
NB	0.336	0.302	73.344	0.680	0.697	0.686	0.750

(d) Average evaluation measures of the full-featured spam classifiers for the Enron-Spam dataset.

TABLE VI: Performances of the classifiers on (a) CSDMC2010, (b) SpamAssassin, (c) LingSpam, and (d) Enron-Spam Datasets.

compared to the comparable baseline are written in *italics*.

B. Classification Performance Evaluation

Treating each dataset independently, the real-valued features are extracted from each email of each dataset. Then, using a conventional stratified 10-fold cross-validation approach, five classifiers are generated using the five algorithms. The classifiers are then evaluated. In a K-fold cross-validation, the original dataset is randomly partitioned into K equal-sized folds or subsets. Then each classifier is trained on $K - 1$ folds and evaluated on the remaining fold. Stratification means that the class (i.e., ham or spam) in each fold is represented in approximately the same proportions as in the full dataset. The cross-validation process is then repeated until each of the K folds is used exactly once as the validation data. The final estimation of the classifier is the average of the K results from the folds.

Table VI shows the performance of the learned classifiers on spam email classification. The most striking attribute of this data is that the best ham misclassification rate is achieved by the BAGGED RF classifiers. In contrast, the best spam misclassification rate is attained by BOOSTED RF. To decide the best, we then refer to a balanced measure of the two: the AUC. According to the data shown in Table VI, BAGGED RF performs the best of all because it has the better AUC. A further indicator of the supremacy of BAGGED RF over the others can be found in Table VIc—its best FPR comes from LingSpam indicating that it performs best at identifying domain-specific hams. The interesting competition between BAGGED RF and BOOSTED RF continues on precision and recall. For all four datasets, BAGGED RF achieves the best precision while the best recall is scored by BOOSTED RF. For two datasets, SpamAssassin and LingSpam, BOOSTED RF achieves the best F-score. On the other hand, BAGGED RF ties with BOOSTED RF for CSDMC2010; for Enron-Spam it is BAGGED RF that attains the best F-score. Recall that, CSDMC2010 and SpamAssassin have similar characteristics as well as spam rates (see Section V). This is complemented by the results in Tables VIa and VIb. The data show that except for the SVM classifiers, others have similar FPRs. When it comes to accuracy, BAGGED RF again outperforms the other algorithms—the only exception is the SpamAssassin dataset.

Compared to the preceding results, the performances of SVM and NB are not satisfactory. Attaining very low FPR with a low AUC by SVM for CSDMC2010 and LingSpam indicates a possible training overfit. The reasons for this possible overfit are yet to be investigated. On the other hand, we examined the correlation among the features. It is evident that the features have inter-dependency[10]—which contradicts the independence assumption of the NB algorithm and can be a reason for the algorithm's poor classification performance.

The data in Table VId describe the performance of our method on the Enron-Spam dataset. The most intriguing finding is that on this dataset, the classifiers' FNRs are remarkably low. Inevitably, the FPRs of the classifiers on the dataset are the poorest. Although the AUC is still reasonable, the results are not quite as good as we expected. This phenomenon confirms that for personalized email data, our approach misclassifies

[10]http://cogenglab.csd.uwo.ca/sentinel/sentinel-attribute-correlations.html

	FPR	FNR	Accuracy %	Precision	Recall	F-score	AUC
RF	0.187	0.320	71.411	0.796	0.714	0.731	0.830
BOOSTED RF	0.159	0.329	71.510	0.805	0.712	0.732	0.843
BAGGED RF	0.125	0.369	69.473	0.809	0.695	0.713	0.855
SVM	0.059	0.578	55.772	0.799	0.558	0.570	0.681
NB	0.176	0.634	48.529	0.712	0.485	0.497	0.645

(a) Classifiers trained on ham skewed Enron 1-3 and tested on spam skewed Enron 4-6.

	FPR	FNR	Accuracy %	Precision	Recall	F-score	AUC
RF	0.458	0.077	64.494	0.808	0.645	0.661	0.865
BOOSTED RF	0.445	0.069	65.709	0.815	0.657	0.673	0.887
BAGGED RF	0.379	0.061	70.793	0.834	0.708	0.723	0.908
SVM	0.842	0.028	37.833	0.763	0.378	0.321	0.564
NB	0.734	0.089	44.048	0.732	0.440	0.425	0.717

(b) Classifiers trained on spam skewed Enron 4-6 and tested on ham skewed Enron 1-3.

TABLE VII: Performances of the classifiers on Enron-Spam.

a lot of legitimate emails. Two other notable aspects of the reported data are that on the Enron-Spam collection the classifiers have (i) the best accuracy and (ii) the poorest recall.

From the results of Table VI, it can be seen that except for Enron-Spam, the FPRs of the classifiers are much better than their FNRs. The most likely cause for the classifiers labelling hams more correctly than spams is that CSDMC2010, SpamAssassin, and LingSpam are ham skewed (Table V), To see whether the training on a skewed dataset affects the overall performance of the classifiers, we further tested with the Enron-Spam dataset. We first trained the classifiers with Enron 1-3 and tested them on Enron 4-6. And then, we trained the classifiers with Enron 4-6 and tested them on Enron 1-3. It is evident from the data in Table VII that training on a ham skewed dataset results in improved FPR while training on a spam skewed dataset brings about better FNR. A significant change to spam recall has also been observed from the results as spam recalls listed in Table VIIa are better than that in Table VIIb. In other words, the results suggest that no matter how we train our classifiers, either with spam or ham skewed training data, we need our test set to contain more spams than hams to get better spam recall. This finding is significant because a low recall results in low F-score and AUC. Overall, from this particular experiment, we can say that to observe the true performance of spam classifiers, reported work should test their system on a balanced dataset.

VII. Discussion and Related Work

In this section, our results are compared with some of the related work. Of the references mentioned throughout this paper, we only compare our results with the commensurate ones—those that used the same dataset. Moreover, the compared results are evaluated with a *student t-test* with a significance level set to 5% (i.e., $\alpha = 0.05$) to report if the differences are significant.

We have mixed results for the CSDMC2010 dataset. Qaroush *et al.* [19], for instance, investigated the performance of several learning algorithms on this dataset. They concluded that RF outperforms the rests. The reported spam recall in their paper is 0.958, which is significantly better than what we found (0.912) (Table VIa). Whereas their precision is similar to that of our approach, because of their high recall, their 0.958 F-score also outperforms our F-score of 0.922 (Table VIa). Surprisingly, we outperform them if we do a cost-sensitive analysis of our data. The AUC that we found for the dataset is 0.988 (Table VIa) which is better than what they found (0.981). An SVM-based spam filter developed by Yang *et al.* [28], on the other hand, reported 0.943 precision, 0.965 recall, and a promising AUC of 0.995. Among the three measures, we only obtained a better precision. Their second anti-spam filter uses a an NB classifier. This filter, interestingly, achieved 100% recall. Its precision of 0.935 and AUC of 0.976, however, was outperformed by our approach (Table VIa). Note that, the differences in the results are statistically significant.

By using 328 features, the filter developed by Ma *et al.* generates a Neural Network classifier. On the SpamAssassin dataset, they reported that both their precision and accuracy was 0.920. On the other hand, our approach achieved a 0.948 precision and 0.957 accuracy. Both of these results are statistically significant. Another Neural Network based filter developed by Srisanyalak and Sornil [29] uses immunity-based features from emails. The filter has been reported to be accurate 92.4% of the time. Our reported accuracy is better than this (Table VIb). The phenomenal FPR and FNR achieved by the filter developed by Bratko *et al.* (FPR=0.001 and FNR=0.012) indicates that our approach needs further improvement in these measures; our reported FPR and FNR are 0.023 and 0.079, respectively (Table VIb).

From previous studies, we found that the performance of the filters are relatively low on the LingSpam dataset. Prabhakar and Basavaraju [10], for instance, applied K-NNC and a data clustering algorithm called BIRCH on this dataset. Their filter achieved 0.698 precision, 0.637 recall, 0.828 specificity, and an accuracy of 0.755. In contrast, the data in Table VIc show that our approach has a precision of 0.944 with 0.838 recall, 0.990 specificity ($1 - \text{FPR}$), and 0.960 accuracy. Our reported AUC on LingSpam also outperformed that reported by Cormack and Bratko [30]; our AUC of 0.986 is significantly better than their AUC of 0.960. The recall we have on this dataset is much better than that reported by Yang *et al.* [28]; the precisions, however, are similar. Their NB-based filter achieved 0.943 precision and 0.820 recall. Surprisingly, the AUC of their filter (e.g., 0.992) significantly outperformed the AUC of our approach (Table VIc).

As mentioned in Section VI-B, our results with the Enron-Spam dataset are not satisfactory because of the *properly balanced* property of the dataset. The curators of the dataset,

however, reported a spectacular spam recall of 0.975 [9] while our best spam recall on the dataset is 0.929 with BOOSTED RF. Moreover, their reported ham recall is 0.972; ours is a mere 0.842 (Table VId). However, we have recently surpassed the results reported by Metsis *et al.* [9] using an anti-spam filter named SENTINEL [31] that we have developed using the ideas presented in this paper.

VIII. CONCLUSIONS

To sum up, we consider the task of email classification as a supervised machine-learning problem. The novelty of this work is the use of a set of features related to the readability of email texts. Because the features are language-independent, the method reported in this paper is potentially able to classify emails written in any language. The aforementioned features as well as the traditional ones are used to generate binary classifiers by five well-known learning algorithms. We then evaluate the classifier performances on four benchmark email datasets. The evidence from this study suggests that although traditional features are individually more important than the other feature types, the combination of all of the features produces the optimal results. Extensive experiments also imply that classifiers generated using meta-learning algorithms perform better than trees, functions, and probabilistic methods. Finally, we compare the results of our method with that of many state-of-the-art anti-spam filters. Although the performance of our method is not always superior to other filter-dataset instances, we find that our approach surpasses a number of them. Taken together, the results suggest that the method described in this paper can be a good means to classify spam emails.

Because our results suggest that meta-learning algorithms perform the best, further tests should be carried out to see the performance of classifiers generated by stacking several algorithms.

REFERENCES

[1] L. F. Cranor and B. A. LaMacchia, "Spam!" *Communication ACM*, vol. 41, no. 8, pp. 74–83, Aug. 1998.

[2] Commtouch, "Internet threats trend report," Commtouch, USA, Tech. Rep., April 2013. [Online]. Available: http://www.commtouch.com/uploads/2013/04/Commtouch-Internet-Threats-Trend-Report-2013-April.pdf

[3] J. Goodman, G. V. Cormack, and D. Heckerman, "Spam and the ongoing battle for the inbox," *Communications ACM*, vol. 50, no. 2, pp. 24–33, Feb. 2007.

[4] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," *Artificial Intelligence*, vol. 29, no. 1, pp. 63–92, Mar. 2008.

[5] L. Zhang, J. Zhu, and T. Yao, "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing*, vol. 3, pp. 243–269, 2004.

[6] C.-C. Lai and M.-C. Tsai, "An empirical performance comparison of machine learning methods for spam e-mail categorization," in *Fourth International Conference on Hybrid Intelligent Systems (HIS '04)*. USA: IEEE Computer Society, 2004, pp. 44–48.

[7] Y. Hu, C. Guo, E. W. T. Ngai, M. Liu, and S. Chen, "A scalable intelligent non-content-based spam-filtering framework," *Expert Systems Applications*, vol. 37, no. 12, pp. 8557–8565, Dec. 2010.

[8] J.-J. Sheu, "An efficient two-phase spam filtering method based on e-mails categorization," *International Journal of Network Security*, vol. 9, no. 1, pp. 34–43, 2009.

[9] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with Naive Bayes – Which Naive Bayes?" in *Third Conference on Email and Anti-Spam (CEAS 2006)*, USA, 2006.

[10] R. Prabhakar and M. Basavaraju, "A novel method of spam mail detection using text based clustering approach," *International Journal of Computer Applications*, vol. 5, no. 4, pp. 15–25, August 2010.

[11] Q. Ma, Z. Qin, F. Zhang, and Q. Liu, "Text spam neural network classification algorithm," in *2010 International Conference on Communications, Circuits and Systems*, China, 2010, pp. 466–469.

[12] P. Graham, "A plan for spam," Available on: http://paulgraham.com/spam.html, Aug. 2003.

[13] C. Orăsan and R. Krishnamurthy, "A corpus-based investigation of junk emails," in *Third International Conference on Language Resources and Evaluation (LREC-2002)*, Spain, May, 29 – 30 2002.

[14] R. Gunning, "Fog index after twenty years," *Journal of Business Communication*, vol. 6, no. 3, pp. 3–13, 1969.

[15] R. Flesch, "A new readability yardstick," *Journal of Applied Psychology*, vol. 32, pp. 221–33, 1948.

[16] G. H. McLaughlin, "Smog grading – a new readability formula," *Journal of Reading*, vol. 12, no. 8, pp. 639–46, 1969.

[17] J. S. Caylor, T. G. Stitch, L. C. Fox, and J. P. Ford, "Methodologies for determining reading requirements of military occupational specialities," Human Resources Research Organization, Alexandria, VA, Tech. Rep. 73-5, 1973.

[18] J. P. Kincaid, R. P. Fishburne, R. L. Rogers, and B. S. Chissom, "Derivation of new readability formulas (automated readability index, fog count, and Flesch reading ease formula) for navy enlisted personnel," Chief of Naval Technical Writing: Naval Air Station Memphis, Research Branch Report 8-75, 1975.

[19] A. Qaroush, I. M. Khater, and M. Washaha, "Identifying spam e-mail based-on statistical header features and sender behavior," in *CUBE International Information Technology Conference*. USA: ACM, 2012, pp. 771–778.

[20] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," in *Machine Learning*, 2000, pp. 135–168.

[21] L. Breiman and L. Breiman, "Bagging predictors," in *Machine Learning*, 1996, pp. 123–140.

[22] M. Ye, T. Tao, F.-J. Mai, and X.-H. Cheng, "A spam discrimination based on mail header feature and svm," in *Fourth International Conference on Wireless Communications, Networking and Mobile Computing (WiCom08)*, 2008, pp. 1–4.

[23] A. Veloso, "Lazy associative classification for content-based spam detection," in *Proc. of the Latin American Web Congress*. IEEE Computer Society, 2006, pp. 154–161.

[24] A. Bratko, G. V. Cormack, D. R, B. Filipic, P. Chan, T. R. Lynam, and T. R. Lynam, "Spam filtering using statistical data compression models," *Journal of Machine Learning Research*, vol. 7, pp. 2673–2698, 2006.

[25] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. Spyropoulos, "An evaluation of naive bayesian anti-spam filtering," in *Proc. of the Workshop on Machine Learning in the New Information Age*, 2000.

[26] M. B. Kursa and W. R. Rudnicki, "Feature selection with the Boruta package," *Journal of Statistical Software*, vol. 36, no. 11, pp. 1–13, 2010.

[27] R. Nilsson, J. M. Peña, J. Björkegren, and J. Tegnér, "Consistent feature selection for pattern recognition in polynomial time," *J. Mach. Learn. Res.*, vol. 8, pp. 589–612, May 2007.

[28] J. Yang, Y. Liu, Z. Liu, X. Zhu, and X. Zhang, "A new feature selection algorithm based on binomial hypothesis testing for spam filtering," *Knowledge-Based Systems*, vol. 24, no. 6, pp. 904–914, Aug. 2011.

[29] B. Sirisanyalak and O. Sornil, "Artificial immunity-based feature extraction for spam detection," in *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, (SNPD 2007)*, vol. 3, 2007, pp. 359–364.

[30] G. V. Cormack and A. Bratko, "Batch and online spam filter comparison," in *Conference on Email and Anti-Spam, (CEAS 2006)*, Mountain View, CA, July 2006.

[31] R. Shams and R. Mercer, "Personalized spam filtering using natural-language attributes," in *Proceedings of the 12th IEEE International Conference on Machine Learning Applications (ICMLA2013)*. Miami, USA: IEEE, 2013.