A on Spam Filtering Classification: A Majority Voting like Approach

Youngsu Dong, Mourad Oussalah and Lauri Lovén
Center for Ubiquitous Computing, University of Oulu, PO Box 4500, 90010 Finland

Keywords: Machine Learning, Text Mining, Majority Voting.

Abstract: Despite the improvement in filtering tools and informatics security, spam still cause substantial damage to public and private organizations. In this paper, we present a majority-voting based approach in order to identify spam messages. A new methodology for building majority voting classifier is presented and tested. The results using SpamAssassin dataset indicates non-negligible improvement over state of art, which paves the way for further development and applications.

1 INTRODUCTION

Global email traffic is constantly growing (The Radicati Group, 2017), and along with it, the number of unsolicited email ("spam") is on the rise (Shams and Mercer, 2013). Spam is sent for various reasons: for promotion campaigns, advertisement, spreading backdoors or malicious programs, to name but a few. Excessive amounts of spam are not only harming individuals by creating frustrating situations for the user, but also generate major problems for the sustainability of the SMEs and IT services.

To cope with the increasing spam traffic, filtering techniques to discriminate genuine from spam emails require constant improvement. However, also spam itself evolves, adjusting to the changes in the filters. This evolution together with the high variability in the textual contents of spam messages makes the development of new algorithms and methodologies challenging, with standard pre-processing and natural language processing techniques often showing their inherent limitations.

This motivates extensive work in development of anti-spam techniques. The latter can be classified (Chuan, et al., 2005) into i) a content-based approach, where the email body is tested for selected key-words, or patterns that are typical for spams; ii) a header-based approach, which requires collecting email addresses of known spammers and known non-spammers; iii) a protocol-based approach, introducing new procedures for sender authentication such as visiting specific websites, collecting personal codes, adding new entries to DNS servers, or the the greylisting approach (Harris, 2017), where the receiving mail server requires an unknown sender to resend the email again later; or iv) a social network based approach, with a graph based metric such as the clustering coefficient is used to test the likelihood of spam (speculating that spammers send thousands or even millions of messages).

This paper focuses on the first class, namely, the content-based approach. We propose a new method using an ensemble of classifiers.

A classifier-based approach for spam-detection is nothing new. For example, Amayri and Bouguila (2010) conducted an extensive study of the use of support vector machines (SVM) in spam filtering. They analyzed the performance of SVMs in relation to a variety of kernels and feature selection methods, and concluded that the SVM recognizes spam messages well. This also reinforces previous results of SVM to text classification (Joachims, 1998).

Further, Metsis et al. (2006) explored various variants and feature selection in Naive Bayes (NB) for spam filtering, using multiple public email corpora for testing the approach. The study concluded that Flexible Bayes and Multinomial Naive Bayes classifier models outperformed several traditional spam filtering models.

In the same spirit, Androutsopoulos et al. (2000) investigated the effect of attribute-set size, training-corpus size, lemmatization, and stop-lists on the performance of NB spam-detection.

In addition to approaches using a single classifier to determine spam email, there are many studies employing multiple classifiers. For instance, Saberi et al. (2007) employed several classifiers, combining their results for better performance. In more detail, an

Dong Y., Oussalah M. and LovÃĺn L.
A on Spam Filtering Classification: A Majority Voting like Approach.
DOI: 10.5220/0006581102930301
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KDIR 2017)*, pages 293-301
ISBN: 978-989-758-271-4
Copyright © 2017 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

ensemble of K-Nearest Neighbours (KNN), NB, and Poisson classifiers reported a considerable improvement in accuracy compared to the result of any classifier alone involved in the study.

Similarly, Tretyakov (2004) combined NB, SVM, and Perceptron classifiers. They, too, conclude that the accuracy with respect to spam misclassified as legitimate emails (false negative) was improved by employing an ensemble of classifiers.

This paper advocates using multiple classifiers for spam detection. However, our approach differs from the state of art in several ways. First, to ensure the generality of the approach, our research is constrained by the configuration employed in state or art approaches on SpamAssassin dataset, which eases comparison. Second, our choice of individual classifiers was motivated by both the intensive testing across multiple classifiers as well as the reported results in the literature. Third, instead of the binary spam/ham classification, a finer-grained classification model opens up the problem of analogy between binary classification and multi-class classification problem. This is motivated by the presence of subcategorization; namely, "hard legitimate emails" in the original dataset.

Section 2 of this paper details the background of the approach. Section 3 deals with the majority-voting based classifier approach. Section 4 highlights setting of the experiment. The results and discussion are reported in Section 5. Finally, conclusion and perspective work are described in Section 6

2 BACKGROUND

2.1 Preprocessing

Building a spam filter requires a sizeable collection of spam and ham messages ("corpus") to train and test the filter. Most classification methods require that the corpus is first preprocessed. This process usually involves steps such as converting e-mails to plain text, removing headers, html components and non-conventional symbols/characters (e.g. non-standard symbols, URLs, or non-textual inputs such as multimedia files or images), tokenizing the message body into words, and removing "stopwords" (i.e. common words such as "a", "an" or "the", that convey very little information). Further, using e.g. the Porter-Stemming algorithm (Willett, 2006), word suffixes may be stripped to remove the most common morphological and inflexional word endings.

Building a spam filter requires a sizeable collection of spam and ham messages ("corpus") to train and test the filter. Most classification methods require that the corpus is first preprocessed. This process usually involves steps such as converting e-mails to plain text, removing headers, html components and non-conventional symbols/characte rs (e.g. non-standard symbols.

2.2 Text Feature Selection

The choice of features, translating the textual input into a numerical representation, plays key-role in evaluating the performance of the classifier of choise. The Vocabulary *V* consists of all the words (usually in their primitive forms) available in the preprocessed corpus. In the "bag of words" representation, each message (document) $M = (M_1, M_2, \ldots, M_{|V|})$ is represented by the subset of *V* contained in the message. A vector representation of size |V| can be utilized for this purpose. In the binary word feature representation, the component M_i is assigned the value 1, if it is present in the message and zero otherwise. Similarly, in words-count (a.k.a TF; Term Frequency) model, the value of M_i corresponds to the frequency of occurrence of the underlying word in the message.

Term Frequency-Inverse Document Frequency (TF-IDF) is another simple yet effective model to represent the message (Salton, and McGill, 1986). TF-IDF is related to the term frequencies of the document, with the weight M_i determined by the product of the term-frequency and the inverse-document frequency. Specifically,

$$M_i = f_{i,M} * \log\left(\frac{|C|}{f_{i,C}}\right) \quad (1)$$

when $|C|$ is the size of the corpus (total number of messages), $f_{i,M}$ denotes the frequency of the ith word of *V* in the message M and $f_{i,C}$ for the number of documents (messages) in the corpus containing the ith word.

2.3 Cross-Validation

Cross-validation is a method that helps overcome data scarcity in constructing a classifier model while preventing the model from overfitting. In general, we often divide the corpus into three subsets: the *training set*, the *validation set*, and the *test set*. The proportion of each set is chosen depending on the experimental condition and the size of the original corpus. Similar to many other related studies, we adopted the 20:20:60 ratio for testing, validating and training. One acknowledges, though, that this can also be

problematic in case of data sub-sampling due to lack of high quality training dataset.

2.4 Classifiers and Modelling

Three classifiers have been selected by our study: Support Vector Machine, Naïve Bayes and Decision Tree. The rationale behind this choice is twofold. First, many related spam-detection filters have been built with these classifiers, offering a nice opportunity for comparison. Second, a simple test with a collection of readily available classifiers (scikit-learn toolkit in python was employed for this purpose) reveals that these classifiers systematically score high in terms of classification accuracy. A short detailed description of the configuration of these classifiers with respect to spam detection task is described next.

Support Vector Machine (SVM) is a classification method that maps class examples (e.g., messages) to points in space and aims to maximize the margin around the hyperplane separating the classes (Vapnik, 1995). It is proven to be quite robust and perform substantially well in applications related to text mining and categorization (Joachims, 1998).

In the context of a spam / ham classification problem, given a training set of n points (or messages) $(\overrightarrow{M_i}, y_i)$ where $\overrightarrow{M_i} \in \mathbb{R}^{|V|}, y_i \in \{-1, 1\}$, SVM attempts to find the "maximum-margin hyperplane" that divides the group of points $\overrightarrow{M_i}$ (of dimension |V|) for which $y_i = 1$ (categorized as spam) from those for which $y_i = -1$ (categorized as ham), so that the distance between the hyperplane and the nearest point $\overrightarrow{M_i}$ from either group is maximized. More formally, let **W** and **b** be the vector normal to the hyperplane and its displacement relative to the origin, respectively, then the decision boundary can be found by solving the following constrained optimization problem

$$Min\ \frac{1}{2}\|W\|^2 \qquad (2)$$

Subject to

$$y_i\left(W^T\overrightarrow{M_i} + b\right) \geq 1 \quad \forall i \qquad (3)$$

The preceding generates a quadratic programming problem that can easily be solved using numerical optimization packages. Several other variants of the above optimization problem have been put forward in order to accommodate non-linear separation through a set of predefined Kernels, or soft-margin optimization criterion, among others (Amayri and Bouguila, 2010).

Naive Bayes (NB) is reported to achieve the best common selection for the problem of text classification and spam filtering (Metsis, Androutsopoulos and Paliouras, 2000). Its principle is based in estimating the conditional probability and statistical independence of the individual features. More formally, a class ς (either spam or ham) is assigned to a message *M* based on the posterior probability *P(ς|M).* For example, class "Spam" is selected if and only if *P(ς=Spam|M) > P(ς=Ham|M)*). Using Bayes' theorem,

$$P(\varsigma|M) = \frac{P(M|\varsigma)P(\varsigma)}{P(M)} \propto P(M|\varsigma)P(\varsigma)$$
$$\propto P\left(M_1, M_2, \dots, M_{|V|}\middle|\varsigma\right)P(\varsigma) \qquad (4)$$

In the case of binary features (M_i is assigned 1 if word V_i is present in message, zero, otherwise), we have

$$P(\varsigma|M) \propto P(\varsigma)\prod_{t=1}^{|V|}\left[b_t P(V_t|\varsigma) + (1-b_t)\left(1-P(V_t|\varsigma)\right)\right] \qquad (5)$$

with b_t =1 if V_t is present in a message, and zero, otherwise.

Now (5) defines a model for generating document feature vectors of class ς, in which the document feature vector is modelled as a collection of |V| weighted coin tosses, the t^{th} having a probability of success equal to $P(V_t|\varsigma)$. Individual probabilities $P(V_t|\varsigma)$ and $P(\varsigma)$ are estimated using the training set, as the ratio of the number of messages of class ς in which word V_t occurs to the total number of messages of that class and the relative frequency of messages of class ς (with respect to total number of messages in the training set), respectively.

A classical multinomial model feature, say, **x** in which x_t is the count of the number of occurrences of word V_t in a message, is often employed to account for the frequency of words. In this respect, the counterpart of (5) is given by

$$P(\varsigma|M) \sim P(\varsigma|\boldsymbol{x}) \propto P(\varsigma)\prod_{t=1}^{|V|}[P(V_t|\varsigma)]^{x_t} \qquad (6)$$

Prior class probabilities are estimated similarly to Bernoulli's model using the training set, while probabilities $P(V_t|\varsigma)$ are computed using the multinomial model vector $\mathbf{x}_i$ of each ith message as:

$$P(V_t|\varsigma) = \frac{\sum_{M_i:M_i\in\varsigma} x_{it}}{\sum_{s=1}^{|V|}\sum_{M_i:M_i\in\varsigma} x_{is}} \qquad (7)$$

where the sum in the numerator expression of (7) is over all messages M_i of the training set whose class is ς.

Especially, multinomial NB has been reported to perform relatively well in the text classification domain (McCallum and Nigram, 1998) , Metsis, Androutsopoulos and Paliouras, 2006). The method is found to be of particular interest when messages contains repetitive wordings that are fully ignored in Bernoulli model (McCallum and Nigram, 1998).

Decision Tree Classifier (DTC) is a non-parametric classification method based on acyclic directed graphs with hierarchical structure, from the highest node (root) to terminal nodes (leafs) that represent document category (spam or ham). While the internal nodes of a decision tree denote the different attributes, the branches between the nodes tell us the possible values that these attributes can have in the observed samples.

A common strategy to build a decision tree is based on entropy and information gain. More specifically, for a message M and using the same notations, its entropy is given by:

$$H(M) = -P(\varsigma|M)log_2P(\varsigma|M) \\ -\big(1-P(\varsigma|M)\big)log_2\big(1-P(\varsigma|M)\big) \quad (8)$$

Similarly, given a training set D of all messages, the information gain of the i[th] term of the vocabulary is given by:

$$I(D,i) = H(D) - \left(\frac{|D_i|}{|D|}H(D_i) + \frac{|D_{\bar{i}}|}{|D|}H(D_{\bar{i}})\right) \quad (9)$$

where D_i stands for the set of messages in D that contain the ith element, and $D_{\bar{i}}$ stands for the messages in D that do not contain the ith element of the vocabulary. The entity under bracket in (9) corresponds to the expected entropy when the ith attribute was used to partition the data. Therefore, the algorithm selects the attribute that yields the minimum entropy (so, maximizing the information gain) in order to split the dataset into left and right subtree. In other words, the information gain is calculated for each term of the vocabulary so that the term that maximizes the information gain is selected as a root node. The process is repeated on the subtrees until the resulting dataset is pure, e.g., only contains one single category (leaf node).

3 ENSEMBLE CLASSIFIER

Classifiers ensemble is a method leveraging a combination of classifiers. The method often performs better than a single classifier, provided appropriate design and handling approach (Ruta and. Gabrys, 2005). Research in ensemble classifier started in early seventies with pioneer work of Tukey (1997), and continued with results such as the AdaBoost algorithm and the theoretical foundations of information fusion theory.

However, research is far from reaching a steady state from both theoretical and practical perspectives, motivating this study. We focus on independent individual classifiers with an identical training set, reducing the problem to finding appropriate adjudication function that links the outcome of individual classifiers.

Majority Voting (or one of its various refinements), where the classification of an unlabeled instance follows the class that obtains the highest number of votes, is well established and commonly employed for this purpose (Ruta and. Gabrys, 2005), building on acknowledged Condorcet's Jury theorem. Indeed, the majority vote method with independent classifiers is guaranteed to give a higher accuracy than individual classifiers when each individual classifier has a probability of p > 0.5 to yield a correct output. Such reasoning is also incorporated in bagging algorithm (boostrap aggregation) (Breiman, 1996).

The simplest approach is a standard majority voting, where each classifier is assigned one single vote. However, to take into the distinct performance levels of individual classifiers, the votes may be weighed. For this purpose, various metrics have been suggested to determine a classifier's weight. We follow Opitz and Shavlik (1996)'s intuitive idea, setting the weight proportional to the classifier's accuracy performance on the validation set. More formally, let $Z = \{z_1, z_2, \dots, z_n\}$ be the set of labelled data in a validation set. For each classifier D_i, one can construct an n-dimensional binary vector $Y_i = \{y_{1i}, y_{2i}, \dots, y_{ni}\}$, with

$$y_{ki} = \begin{cases} 1 \;\; if \;\; z_k \text{ is correctly labelled by } D_i \\ 0, \qquad\qquad \text{Otherwise.} \end{cases} \quad (10)$$

Therefore, the accuracy of D_i on a validation set Z is given by

$$A_i = \sum_{k=1,n} y_{ki} \quad (11)$$

The latter can be normalized with respect to accuracy of all classifiers to yields a weighting factor as

$$\alpha_i = \frac{A_i}{\sum_j A_j} \quad (12)$$

On the other hand, instead of looking at global classifier accuracy, it is possible to focus on the accuracy for each class. Accordingly, (11) can be modified to

$$A_i^c = \sum_{k=1,n} y_{ki}^c \qquad (13)$$

where y_{ki}^c takes a value if classifier D_i correctly classifies instance z_k in class c, otherwise $y_{ki}^c = 0$.

After normalization (12), this gives the weight α_i^c with respect to each class label c (i.e. spam and ham classes).

This, the output of the majority voting for an unknown instance *x* can be written as:

$$Class(x) = \arg max_c \sum_i \alpha_i . I_{y_i(x)}^c \qquad (14)$$

where $I_{y_i(x)}^c$ is an indicator variable indicating whether the outcome of classifier D_i for input *x* falls in class *c* or not.

Weighing the votes with the class-specific accuracies α_i^c, we can rewrite (14) as

$$Class(x) = \arg max_c \sum_i \alpha_i^c . I_{y_i(x)}^c \qquad (15)$$

In the context of our study, we deliberately restricted the number of classifiers to Naives Bayes, SVM and Decision Tree. This choice is justified by the wide popularity of the above classifiers in the related research, easing comparative analysis. Further, these classifiers performed best in our tests.

4 EXPERIMENTAL SETTING

4.1 Email Corpus

We used the SpamAssassin email corpus. It is a historical and standardized public corpus, appearing in a variety of studies from the past to present (Chuan, et al., 2005), (Zhang, Zhu and Yao, 2004), (Bratko, et al., 2006), (Katakis, Tsoumakas and Vlahavas, 2010). All elements of email (heading, body, etc.) were considered in the classification process.

The corpus consists of a total of 6100 legitimate (ham) and unsolicited (spam) emails. The legitimate emails are sub-categorised into hard legitimate emails (spam resembling structure; subscribed promotions) and generic emails. Spam ratio is 30%, and there are 250 hard legitimate emails as highlighted in Figure 1. An example of instance of emails is depicted in figures 2-4.

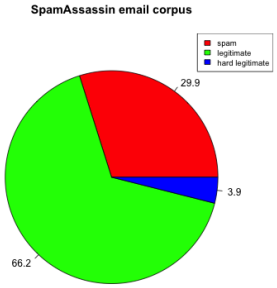

Figure 1: Statistics of the corpus used in the experiment.

```
Can someone explain what type of operating system Solaris is... as ive never
seen or used it i dont know wheather to get a server from Sun or from DELL i
would prefer a linux based server and Sun seems to be the one for that but
im not sure if Solaris is a distro of linux or a completely different
operating system? can someone explain...

Kiall Mac Innes
```

Figure 2: A legitimate email example.

```
<!-- NEWS -->
<table bgcolor="#FFCC99" border="0" cellspacing="0" cellpadding="0"
width="100%"><tr><td><p class="title">
  07.10.2002 GnomeREPORT </p></td></tr></table><p>
<p class="sidebar"><table cellpadding="10" cellspacing="0" border="0"><tr><td>
<font style="font-size: 10pt" face="Verdana, Tahoma, Arial" size="2">
<b>WE NEED YOU TO BE AT GNOMEDEX!</b> That's what your employer should be
telling you. Microsoft Pocket PC experts will be sharing tips on improving
workplace productivity with portable devices. Combined with Proxim's on
location wireless networking solutions, a Pocket PC can keep you and your
co-workers communicating no matter where you are in the building. Learn
how to bring Wi-FI and PDA technologies together for the ultimate in
portable computing.
<a href="http://www.gnomedex.com">Register for Gnomedex today</a>!
</font></td></tr></table></p>
Reader Jack Macdonald has some advice for you would-be bards in the audience:

<p>"You mention Karaoke being featured at Gnomedex and you
might want to point out to other Gnomies that they could
practice their Karaoke on my <a
href="http://www.geocities.com/Broadway/Lobby/1009/">Red Hot
Karaoke Page</a> site. There are the words and music for
more than 1100 songs available on the site and by
downloading the Yamanha Midiplug (a link is provided) they
can even change the key of the song to suit their voice
range as well as speed up or slow down the music to their
taste. In other words, to have the same experience as they
would have with professional Karaoke equipment but at no
extra cost! At this price (zero) a lot of gnomies going to
Gnomedex could be singing up a storm by being properly
prepared for your Karaoke."

<p>Travel day on Wednesday as I head down to Ankeny, Iowa to
help out a former co-worker and avid Gnomie with some server
and networking stuff. The tough part is remembering
everything I'll need. It never fails, though. I always,
always, always forget something. I can make lists two weeks
in advance and I'll end up 30 miles down the road with a
handprint on my forehead when I come to the realization that
I left something important behind. What will it be this
time, I wonder? I'll let you know on Thursday.
```

Figure 3: A hard legitimate email example.

4.2 Configurations

Prior to applying the majority voting rule (14-15), it is important to tune the parameters of individual classifiers such that the overall performance level is likely maximized. On the other hand, because of sensitivity of the performance with respect to the distribution of hard-legitimate emails, the number of hard ham (spam-like legitimate email) is equally allocated to the train and test set. On the other hand, since the performance of the algorithms depends on the configuration of the training set, generated at random and changing at each run, we deliberately repeated the running of the algorithm one hundred times. The mean and standard deviation values are therefore reported for individual classifier results.

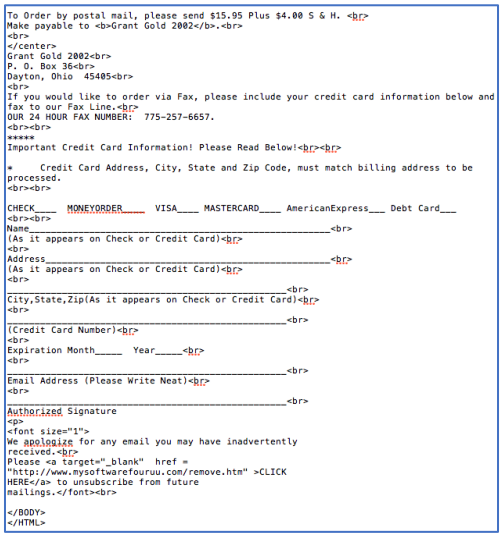

Figure 4: A spam mail example.

We used with most standard textual features (e.g., TF-IDF, TF, Binary) to restrict the scale of our study and emphasize the expected gain from utilizing some unexplored properties of the dataset, as well as the limit of the majority-voting like mechanism.

We implemented classifiers using the Scikit-learn library of the Python script language. The library is available as open-source and built specifically for Python, which eases the preprocessing that includes tokenization, cleanup, among others, and post-processing tasks.

5 EXPERIMENTAL SETTING

5.1 SVM

Scikit-learn allows different types of kernel functions for an SVM model. While some studies state that the majority of text classification problems can be solved by a linear kernel (Joachims, 1998), (Zhang, Zhu and Yao, 2004), others argue that the performance varies across different parameters and kernel settings (Amayri and N. Bouguila, 2010). To make a fair assertion, we ran a set of validation tests, modelling classifiers with different kernel functions provided by Scikit-learn library. A comparative result is presented in Figure 5 and Table 1.

Figure 5 shows that given a training set sampled from the corpus, the linear kernel (with TF-IDF features) performs best. We investigated also the features, with TF-IDF compared to binary and TF features in Table 1. The results show the TF-IDF features consistently performed very well on all kernel types with, justifying the choice in further studies.

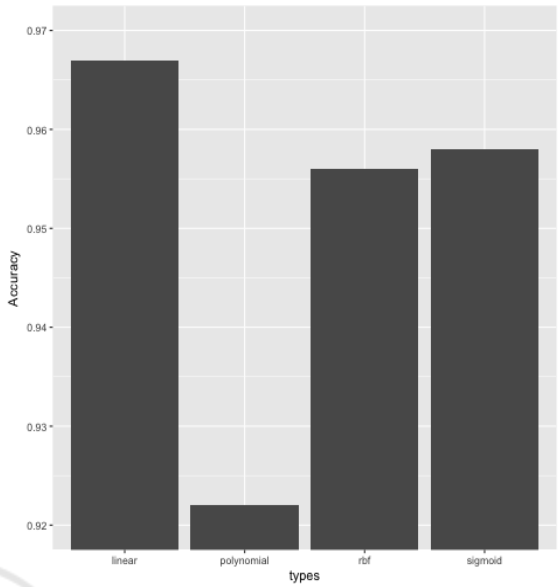

Figure 5: SVM performance evaluation by four different kernel functions applied: Linear, Polynomial, RBF, and Sigmoid; kernel degree is set to 1 and 0.7 for gamma value.

Table 1: Results of cross-validation, with SVM kernel models with different feature selection approaches. The total hard ham (spam-like legitimate email) is joined in the training set.

	Linear	*Polynomial*	*RBF*	*Sigmoid*
Binary	**0.95 (∓ 0.31**	0.94 (∓ 0.32)	0.70 (∓ 0.00	0.70 (∓ 0.00)
TF	**0.94 (∓ 0.31**	0.92 (∓ 0.29)	0.69 (∓ 0.00	0.69 (∓ 0.00)
TF-IDF	**0.94 (∓ 0.28**	0.91 (∓ 0.25)	**0.94 (∓ 0.28**	0.93 (∓ 0.28)

5.2 Naïve Bayes

Similarly to SVM, we tested the performance of the NB classifier with respect to the three most commonly employed features (binary features, TF, TF-IDF). We also compared the multinomial and multivariate based models (Schneider), Eyheramendy, Lewis and Madigan, 2003) of NB implementation. The results are summarized in Table 2.

Table 2: NB model comparison by feature selection methods; results of cross-validation.

	Multinomial	*Multivariate*
Binary	**0.97 (∓ 0.11)**	0.92 (∓ 0.14)
TF	**0.93 (∓ 0.07**)	0.92 (∓ 0.13)
TF-IDF	0.86 (∓ 0.07)	**0.92 (∓ 0.13)**

Table 2 shows that the multinomial model performs better with binary word feature model, while the performance of the multivariate model seems to be more consistent regardless of the selection of the feature models.

We investigated also the alpha-parameter of the multinomial classifier model (since it showed a better performance in average; Table 2), with the accuracy on the validation set for various values of alpha reported in Fig. 6. The result indicates a model with the default alpha value a≤1 yields the best performance.

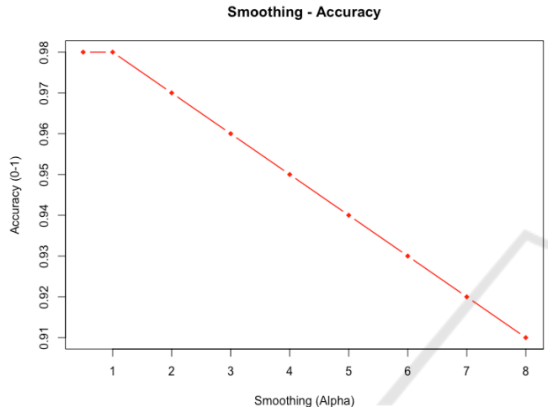

Figure 6: NB performance measure by incremental alpha value.

5.3 Decision Tree Classifier

The DT classifier results for the three distinct features are summarized in Table 3. The results show that given the corpus, DTC performs consistently well regardless of applied feature selection approach.

Table 3: DTC Accuracy with respect to various features and two impurity measures.

	DT Accuracy
Binary	0.96 (∓ 0.12)
TF	**0.97 (∓ 0.09)**
TF-IDF	0.96 (∓ 0.10)
Entropy	**0.98**
Impurity	
Gini Impurity	0.97

We also tested the impurity criterion (Tan, Steinbach. and Kumar, 2006) by the DTC algorithm to see whether entropy or gini-impurity based criterion is preferred. The results shown in Table. 3 supports the default entropy criterion for the DT classifier.

5.4 Majority-Voting Classifier

An ensemble of three classification algorithms (SVM, DTC, and NB) is evaluated to measure the performance of the classification method on the test set. First, we further tuned individual classifier models using cross-validation in order to maximize their performance levels. The results displayed in Table 4 distinguishes between the case where the "hard-ham" cases are shifted dominantly to the test-dataset and the case where they are uniformly distributed. More specifically, the two experiments allocate the number of hard-ham in different proportion, where the first experiment puts the entire hard-ham into test data set, while the second one selects the sample data on the given input parameter basis (random selection for each categories of emails). The result shows that the majority voting method performs remarkably well as compared to individual performance levels. Especially in the first case, the majority-voting based classifier provided more robustness and a higher average performance in the final decision. For the sake of notation simplification in Table 4 and subsequent ones, one shall denote by the accuracy, precision, recall, F1-score by A, P, R an F.

In Table 5, we tested the majority voting according to the weightings (14) and (15), accounting for individual classifier performance on the validation set.

Table 4: Classifier model performance comparison.

	SVM		*NB*		*DTC*		*SVM+NB+DTC*	
*Results of the classifiers tested on data set **with hard-ham**.*								
A	.8788		.8825		.8831		**.8875**	
	H	S	H	S	H	S	H	S
P	**.9975**	.7036	.9801	**.7215**	.9853	.7140	.9936	.7180
R	.8338	**.9948**	**.8535**	.9566	.8462	.9683	.8463	.9865
F	.9083	.8241	.9124	.8225	.9104	.8219	**.9140**	**.8311**
*Results of the classifiers trained and tested on **global** data set.*								
A	**.9922**		.9795		.9754		.9915	
	H	S	H	S	H	S	H	S
P	**.9956**	.9909	.9785	.9847	.9814	.9677	.9917	**.9910**
R	.9959	**.9903**	.9934	.9515	.9856	.9582	**.9960**	.9815
F	**.9957**	**.9905**	.9858	.9678	.9834	.9629	.9938	.9862

A: Accuracy, P: Precision, R: Recall, F: F1-score, H: Legitimate email (Ham), S: Spam email.

The result shown in Table 5 indicates that a standard majority voting scheme outperforms the weighted majority classifier regardless of the weighing ((14) or (15)) used. To explain the result, Table 6 exhibits the accuracy of individual classifiers on spam and ham class category (the global classifier accuracy being the average of the two accuracies).

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Table 5: Comparison of weighted majority classifiers.

	Majority (14)		*Majority (15)*	
*Results of the classifiers tested on data set **with hard-ham**.*				
A	**0.8789(0.002)**		0.8851(0.0012)	
	H	S	H	S
P	**0.9973(0.001)**	0.6971(0.01)	0.9918(0.0013)	0.7072(0.004)
R	0.8347(0.003)	**0.9943(0.003)**	0.8495(0.002)	0.9812(0.002)
F	0.9087	0.8195	**0.9151**	**0.8219**
*Results of the classifiers trained and tested on **global** data set.*				
A	**0.9926(0.001)**		0.9913 (0.002)	
	H	S	H	S
P	0.9948(0.002)	0.9879(0.007)	0.9922(0.002)	**0.9892(0.005)**
R	0.9944(0.003)	0.9890(0.006)	**0.9952(0.002)**	0.9826(0.004)
F	**0.9945**	**0.9884**	0.9936	0.9858

A: Accuracy, P: Precision, R: Recall, F: F1-score, H: Legitimate email (Ham), S: Spam email.

Table 6: Confidence table: weight for each class is given based on f1-score of the class of each classifier. LC: confidence value on legitimate emails, SC: confidence value on spam emails.

	F1-score		*Confidence*	
	H	S	HC	SC
SVM	**0.975**	**0.944**	**0.337**	**0.341**
MNB	0.963	0.916	0.333	0.331
DT	0.953	0.902	0.329	0.326

Table 6 clearly indicates that SVM outperforms other classifiers in terms of classification accuracy on the validation set, which, in turn, makes the result of the weighted classifiers very much biased by the outcome of SVM, and, therefore, fails to capture the diversity among the different classifiers.

5.4 Use of Subcategorization

The SpamAssassin public corpus contains 250 emails labelled as "hard-ham". For the hard-ham emails, the content resembles a spam email. This makes it difficult for a model to make the classification decision. Indeed, all of our models struggled with classifying hard-ham: including the hard-ham emails in a test set pulled down the accuracy of classifies up to 11% (Table 4).

To tackle this, we defined hard-ham as an independent class apart from the legitimate and ham emails. Accordingly, we trained the individual classifiers to classify all three classes. The result exhibited in Table 7 shows a substantial improvement in accuracy of around 0.991 (SVM) and 0.988 (SVM+NB+DTC) in comparison to the data present in Table 5.

Table 7: Classification performance of models trained and tested on the multi-class base.

	SVM			*NB*			*DTC*			*SVM+NB+DTC*		
A	**0.991**			0.971			0.974			0.988		
	H	S	HH	H	S	HH	H	S	HH	H	S	HH
P	**.99**	.98	.97	.96	.97	**.99**	.98	.96	.93	.98	.98	**.99**
R	**.99**	**.99**	.88	**.99**	.95	.70	.97	.96	**.92**	**.99**	.98	.86
F	**.99**	.98	**.92**	.97	.96	.82	.97	.96	**.92**	.98	.98	**.92**

A: Accuracy, P: Precision, R: Recall, F: F1-score, H: Legitimate email (Ham), S: Spam email, HH: Spam-like legitimate email (Hard-ham).

This result raises further research questions for the use of subcategorization in order to enhance the classification result. For example, given a binary classification problem that categorizes classes c1 and c2, if elements of class c1 can further be split into c3 and c4 classes, how does the classification of c2, c3, c4 enhance or degrade the initial binary classification problem. Without speculating on the result, the problem is ultimately linked to the nature of features and the quality of the training set, especially with respect to the subcategory dataset. This constitutes a part of our future investigation in this context, which lies down promising theoretical foundation. Table 8 summarizes the performance of the binary and multi-class classification.

The performance level achieved competes with state of the art results obtained using the same corpus.

For instance, Chuan, et al., (2005) reported a spam precision and a spam recall of 98.97 and 93.58 respectively. A lower classification performance is also reported in Bratko et al., (2006).

Table 8: Comparison of model performance, measured by different metrics.

	NB	*ANN-BP*	*ANN-LVQ*	*SVM+NB+DTC (Multi class)*	*SVM+NB+DTC (Binary class)*
Spam precision	97.63	98.42	98.97	98	**99.50**
Spam recall	86.48	91.26	93.58	98	**99.01**

6 CONCLUSION

This paper addressed the usefulness of majority-voting based classification strategy for spam identification with specific focus to SpamAssassin corpus combining machine learning and natural language processing based techniques. Three most commonly employed classifiers; namely, Support vector machine, Naïves Bayes and Decision Tree have been implemented and tested with textual features. The result shows that standard majority voting strategy can increases the performance of individual classifiers in terms of accuracy, precision, recall metrics. Nevertheless, the use of weighted

majority classifier according to accuracy of individual classifiers on validating set fails to display the expected improvement. On the other hand, the study also reveals the importance of exploring the subcategorization that exists in the original dataset, where substantial improvement has been noticed, which brings the achieved accuracy marginally outperforming many of the state of the art results employing the same dataset. This also opens interesting perspective work in order to explore the theoretical foundation of such mechanism.

REFERENCES

R. Shams, and R. E. Mercer, "Classifying spam emails using text and readability features.," in Data Mining (ICDM), 2013 IEEE 13th International Conference on, 2013.

The Radicati Group, Inc., "Email Statistics Report, 2015-2019," Retrieved Augest 14, 2017 from Radicati's database.

Z. Chuan, et al., "A LVQ-based neural network anti-spam email approach.," *ACM SIGOPS Operating Systems Review*, vol. 39, no. 1, pp. 34-39, 2005.

E. Harris, "The Next Step in the Spam Control War: Greylisting by Evan Harris," 21 08 2003. [Online]. Available: http://projects.puremagic.com/greylisting/whitepaper.html. [Accessed 2 08 2017].

O. Amayri and N. Bouguila, "A study of spam filtering using support vector machines.," *Artificial Intelligence Review*, vol. 34, no. 1, pp. 73-108, 2010.

T. Joachims, "Text categorization with support vector machines: Learning with many relevant features.," in *Machine learning*, 1998.

V. Metsis, I. Androutsopoulos and G. Paliouras, "Spam filtering with naive bayes-which naive bayes?," *CEAS*, vol. 17, pp. 28-69, 2006.

I. Androutsopoulos, et al., "An evaluation of naive bayesian anti-spam filtering.," 2000.

A. Saberi, M. Vahidi and B. M. Bidgoli, "Learn to detect phishing scams using learning and ensemble? methods.," in *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*, 2007.

K. Tretyakov, "Machine learning techniques in spam filtering.," *Data Mining Problem-oriented Seminar*, vol. 3, no. 177, pp. 60-79, 2004.

P. Willett, "The Porter stemming algorithm: then and now.," *Program*, vol. 40, no. 3, pp. 219-223, 2006.

G. Salton, and M. J. McGill, "Introduction to Modern Information Retrieval.," 1986.

V. Vapnik, "The nature of statistical learning theory," 1995.

A. McCallum and K. Nigram, "A comparison of event models for naive bayes text classification," *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41-48, 1998.

D. Ruta and B. Gabrys, "Classifier selection for majority voting.," *Information fusion*, vol. 6, no. 1, pp. 63-81, 2005.

J. W. Tukey, "Exploratory data analysis," 1977.

L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123-140, 1996.

D. W. Opitz, and J. W. Shavlik, "Generating accurate and diverse members of a neural-network ensemble," 1996.

L. Zhang, J. Zhu and T. Yao, "An evaluation of statistical spam filtering techniques.," *ACM Transactions on Asian Language Information Processing* , vol. 3, no. 4, pp. 243-269, 2004.

A. Bratko, et al., "Spam filtering using statistical data compression models.," *Journal of machine learning research*, vol. 7, pp. 2673-2698, Dec 2006.

I. Katakis, G. Tsoumakas and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: an application to email filtering.," *Knowledge and Information Systems*, vol. 22, no. 3, pp. 371-391, 2010.

K. M. Schneider, "On word frequency information and negative evidence in Naive Bayes text classification.," in *Advances in Natural Language Processing*.

S. Eyheramendy, D. D. Lewis and D. Madigan, "On the naive bayes model for text categorization," *Citeseer*, 2003.

PN. Tan, M. Steinbach and V. Kumar., "Classification: basic concepts, decision trees, and model evaluation." in *Introduction to Data Mining*, 2006, pp. 145-205.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS