Intelligent Detection Approaches for Spam *

Guangchen Ruan and Ying Tan †
The State Key Laboratory of Machine Perception, Peking University
Department of Machine Intelligence, School of EECS
Peking University, 100871, P.R.China
{ruangc,ytan}@cis.pku.edu.cn

Abstract

This paper proposes intelligent detection approaches based on Incremental Support Vector Machine and Artificial Immune System for the spam of e-mail stream. In the approaches, a window is used to hold several classifiers each of which classifies the e-mail independently and the label of the e-mail is given by a strategy of majority voting. Exceeding margin update technique is also used for the dynamical update of each classifier in the window. A sliding window is employed for purge of out-of-date knowledge so far. Techniques above endow our algorithm with dynamical and adaptive properties as well as the ability to trace the changing of the content of e-mails and user's interests in a continuous way. We conduct many experiments on two public benchmark corpus called PU1 and Ling. Experimental results demonstrate that the proposed intelligent detection approaches for spam give a promising performance.

1. Introduction

Nowadays, as the popularity of the Internet, e-mails have become a very common and convenient medium for communications. However, the proliferation of spam, which is usually defined as unsolicited commercial e-mail (UCE) or unsolicited bulk e-mail (UBE), has caused increasingly serious problems to our normal communications. Numerous spam not only occupies valuable communications bandwidth and storage space, but also wastes user time to tackle with them. In addition, it also brings serious threats to the security of Internet, especially when spam's carrying with virus and malicious code.

To solve problems caused by spam, many solutions have been proposed to detect and filter out spam of e-mails. Here the authors classify them as three catalogs, i.e., simple approaches, intelligent approaches and hybrid approaches.

Simple approaches include munging, listing, aliasing and challenging [9]. Intelligent approaches play an increasingly important role in anti-spam in recent years for their ability of self learning and good performance, including Naïve Bayes [5], Support Vector Machine [4], Artificial Neural Network [7], Artificial Immune System [2] and DNA Computing [6]. Many researchers also propose hybrid approaches by combining two or more techniques in attempts to improve overall performance whilst overcoming the shortcomings of each single approach [9].

Support Vector Machine (SVM) proposed by V. Vapnik is a classification algorithm based on the Structural Risk Minimization principle in statistical learning theory. The goal of SVM is to find an optimal hyperplane for which the lowest true error can be guaranteed. SVM is quite attractive for its good generalization performance.

Artificial Immune Systems (AIS) inspired by the human immune system (HIS) have become an increasingly popular computational intelligence paradigm in recent years. AIS seeks to use the observed immune components and processes of HIS as metaphors to produce artificial systems that encapsulate some desirable properties of HIS. These AIS are then applied to solve complex problems in a wide variety of domains.

In this paper, we propose intelligent detection approaches of spam based on incremental SVM and AIS. To achieve the desirable dynamic and adaptive features, a window is used to hold several classifiers each of which classifies the e-mail independently and the label of the message is given by a strategy of majority voting. The exceeding margin update technique is also used for dynamical update of each classifier in the window. A sliding window is employed for purge of out-of-date knowledge so far. In such ways, the proposed approaches are able to trace the changing of e-mail stream and user's interests.

The paper is organized as follows. In Section 2, the principles and algorithm implementations are detailed. The ex-

*This work is supported by Natural Science Foundation of China (NSFC) under grant number 60673020.
†Corresponding author.

perimental results are reported in Section 3. Finally, conclusions are given in Section 4.

2 Principles and Algorithm Implementations

2.1 Message Representation

The first stage of pattern recognition is to preprocess the data. For e-mail classification, the question is how to represent the message of an e-mail. We define a feature to be a word in an e-mail or a message, and therefore the message can be represented as a feature vector composed of various words from the bag of words formed by analyzing the messages. There are serval methods to construct the feature vector, such as TF (Term Frequency), TF-IDF (Term Frequency–Inverse Document Frequency), and binary representation. Drucker et al. show that SVM with binary features outperforms other methods [4]. Here, we adopt binary representation of message in our algorithm. We use w_i to represent a word, and the message can be expressed as

$$message = (w_1, w_2, \cdots, w_m) \tag{1}$$

where m is the number of words in the bag, and w_i, $i = 1, 2, \ldots, m$ takes value of 1 or 0 to indicate whether it occurs in this message or not.

2.2 Dimension Reduction

If we extract each word appearing in messages to form a dictionary, the size of this dictionary may be tremendously large. Thus the dimension reduction of the feature space is required to avoid the curse of dimension. According to [10] the features that appear in most of the documents are not relevant to separate these documents because all the classes have instances that contain those features. In addition, the words used rarely give us few information during classification either. So, for simplification, we adopt the processing method in [2], namely discarding the features that appear less than 5% and more than 95% in all messages of the corpus.

2.3 Generation of Initial Classifier

We choose some parts of the corpus as our training set to generate the initial classifier comprised of *Detector Set* and *Memory Set*. Firstly we transform each message to a binary vector in the way we mentioned above. After the completion of pre-processing, these training vectors are used to train the SVM. We use *support vectors* of the trained SVM as *naïve detectors* to form the *Detector Set*.

The *Memory Set* is the set of detectors with better performance, which we call *memory cells*. We use *Detector Set* to classify the training data according to certain classification criterion (we will detail them in section 2.4). The *naïve detectors* whose number of correctly classified messages exceeds the threshold n_m set in advance are promoted to be memory cells, meanwhile they are removed from current *Detector Set*. In addition, each member of *Detector Set* and *Memory Set* has a label which is identical with that of the corresponding *support vector*, and we assign a *lifespan* for each memory cell.

2.4 Classification Criterion

Two kinds of classification criteria which can be regarded as different implementations of continuous detection are developed, and we take some performance comparisons between them in experiments. They are described as follows.

Hamming Distance: This approach is to calculate the *Hamming Distance* between the message to be classified and each *naïve detector* as well as *memory cell*. The one which has the minimum distance is added to a set called *Committee*. Each member of *Committee* votes in terms of its label, and the label of the message is given by majority voting. It is notable that sometimes the votes of two sides are equal, and we classify the message as non-spam in such circumstance. The reason why we take such policy is that misclassify a non-spam to spam is much more serious than the opposite. Because of the binary representation of the feature vector, the minimum *Hamming Distance* is equivalent to the minimum *Euclidean Distance*, and the way of decision making is equivalent to *Nearest Neighbor* (NN). In addition, there is an optional procedure called *mutation* in which each member of the *Committee* is mutated. The one that makes a correct decision is mutated to get closer to the message in the feature space. On the contrary, the one making a mistake is moved far away from the message. We implement this idea by changing some entries of the vector to be identical with or different from the corresponding entries of the message according to the mutation *rate* preset in advance.

SVM: This method is directly to use SVM to classify the message. Namely, to check the message is at which side of the *optimal hyperplane* and classify it accordingly. Difference that distinguish from Hamming Distance approach is no *Detector Set* and *Memory Set* for SVM approach. We just need to obtain *support vectors* from the trained SVM when generating the initial classifier.

2.5 Update of the Classifier

The task of on-line e-mail classification is actually to process data in stream mode. The contents of e-mails and user' interests change dynamically. Thus the classifier built

from previous data may not always be suitable for future data. For SVM, the number of *support vectors* is small compared to the total number of training examples, they provide a compact representation of the data, to which new examples can be added as they become available [3].

There are several techniques for the incremental learning of SVM , such as *Error-driven technique (ED)*, *Fixed-partition technique (FP)*, *Exceeding-margin technique (EM)*, and *Exceeding-margin+error technique (EM+E)* [8]. The experimental results on Large-noisy-crossed-norm data and real data set Pima taken from UCI Machine Learning Repository show that *EM* technique achieves similar error rate compared to other three whilst the number of support vectors is relatively small [3]. Therefore, we update the classifiers based on *Exceeding-margin technique (EM)* in our algorithms.

EM-Update: Given a model $classifier_t$ at time t, when a new message arrives, we check whether this data point exceeds the margin defined by the SVM of $classifier_t$ or not. If the condition is satisfied, the message is kept, otherwise it is discarded. Once the number of messages exceeding the margin is equal to or greater than n_e, the update of $classifier_t$ takes place. The *Detector Set* and *Memory Set* of $classifier_t$, together with n_e messages, are used as training data to obtain a new model $classifier_{t+1}$ at time $t+1$. The construction is very similar to the process described in section 2.3. The *memory cells* with positive *lifespan* are reserved during each update. When SVM is used as the classification criterion, we only need to construct the SVM_{t+1} of $classifier_{t+1}$, by using the *support vectors* of SVM_t and n_e messages as training data, because there are no *Detector Set* and *Memory Set*.

Through the duration of two successive EM-Updates, there are also some slight updates for *Detector Set* and *Memory Set*. We increase the number of correctly classified messages of *naïve detector* when it makes a correct classification. The one whose number of correctly classified messages exceeds the threshold n_m is added to *Memory Set* immediately. Meanwhile, the lifespan of each memory cell decreases one after each classification. The slight update above is not needed for SVM classification criterion.

2.6 Work Process of Sliding Window

We use EM technique to construct the incremental learning algorithm. Meanwhile, we have to forget data points no longer in active use. Thus, we use the model in [3] to maintain an representation of a *window* of recent batches of e-mails.

The *window* works as follows, we consider the incoming e-mail in batches with a given size *b*, and maintain *w classifiers* representing the previous 1, 2, ..., *w* batches. The *w classifiers* are updated incrementally and independently according to EM-Update technique as data become available. Let us denote the *w classifiers*, at time *t*, $classifier_1^t, classifier_2^t, \ldots, classifier_w^t$, respectively. When a new batch of e-mail comes in, at time *t + 1*, $classifier_w^t$ is discarded, the remaining $classifier_1^t, \ldots, classifier_{w-1}^t$ become $classifier_2^{t+1}, \ldots, classifier_w^{t+1}$, respectively. $classifier_1^{t+1}$ is newly created by only using the new batch of e-mail. This process can be formulated as

$$classifier_{i+1}^{t+1} = classifier_i^t, \quad 1 \leq i \leq w-1 \qquad (2)$$

where w is the size of the window.

Each *classifier* in *window* represents recent batches of e-mails seen so far. In details, $classifier_1^t$ represents the latest batch while $classifier_w^t$ represents e-mail stream of previous *w* batches. When a new e-mail arrives, we use each *classifier* in *window* to classify it independently using some criterion described in section 2.4. Each *classifier* has an equal weight. The label of the e-mail is given by a strategy of majority voting. This can be regarded as voting in another level, *classifiers* are "experts" with different knowledge and work together to give out the final decision.

A more complicated strategy of voting, namely weighted majority voting, can be employed. The weight of each classifier in the window could be set according to the feature of email stream and furthermore could be adjusted dynamically. When the contents of e-mail stream drift dramatically, we can increase the weights of "younger" classifiers such as $classifier_1^t, classifier_2^t$, etc, to reflect the immediate change. On the contrary, when the variation is slight, it will benefit from increasing the weights of "older" classifiers because they have used more data. In practice, the changing trend may be smooth or not, so the update of the weights is also dynamic.

Table 1. Parameters and their values of our proposed approaches, where w is the size of the window, b the size of the batch, n_e the number of messages exceeding the margin, n_m the threshold of promotion for a naïve detector, *lifespan* the lifespan of memory cell and *ratio* the mutation rate.

Parameter	Value	Range
w	3 or 5	≥ 1
b	60	≥ 1
n_e	30	≥ 1
n_m	5	≥ 1
lifespan	60	≥ 1
ratio	5%	$[0, 1]$

Table 2. Performances on corpus PU1 with window size 3

Methods	Acc (%)	Pre (%)	Rec (%)	MR (%)
M1	80.76	77.25	79.47	18.24
M2	83.45	82.85	78.48	12.68
M3	95.78	95.42	94.91	3.54

Table 3. Performances on corpus PU1 with window size 5

Methods	Acc (%)	Pre (%)	Rec (%)	MR (%)
M1	81.57	79.20	78.54	16.07
M2	86.23	87.33	80.25	9.12
M3	96.16	96.40	94.77	2.75

Table 4. Performances on corpus Ling with window size 3

Methods	Acc (%)	Pre (%)	Rec (%)	MR (%)
M1	86.09	56.44	77.31	12.16
M2	90.89	77.02	64.3103	3.82
M3	97.04	97.57	84.29	0.42

Table 5. Performances on corpus Ling with window size 5

Methods	Acc (%)	Pre (%)	Rec (%)	MR (%)
M1	87.04	59.10	77.96	11.16
M2	92.26	83.68	66.19	2.55
M3	97.65	97.10	88.48	0.53

3 Experiments

3.1 Corpus Used In Experiments

Two corpus used to test our proposed approaches in this paper are the PU1 corpus and Ling corpus [1] (They may be downloaded from http://www.iit.demokritos.gr/skel/iconfig/).

PU1 corpus consists of 1,099 messages, with spam rate 43.77%. Ling corpus consists of 2,893 messages, with spam rate 16.63%. All the messages in both corpus have header fields, attachments and HTML tags removed, leaving only subject line and mail body text. In PU1, each token is mapped to an unique integer to ensure the privacy of the content while keeps its original form in Ling. Each corpus is divided into ten partitions with approximately equal amount of messages and spam rate. There are four versions of the corpus: with or without stemming and with or without stop-word removal. Stop-word removal is a procedure to remove most frequently used words such as 'and', 'for', 'a' and the stemming is the process of reducing a word to its root form (e.g., 'teacher' becomes 'teach'). Androutsopoulos et al. demonstrate that stop-word removal and stemming may not promote a statistically significant improvement [1], so we adopt the original version without stemming and stop-word removal in our experiments.

3.2 Performance Measures

We adopt following measures as performance indices in our experiments. Accuracy is defined as the percentage of messages classified correctly. Precision is defined as the proportion of the number of correctly classified spam messages to the number of messages classified as spam. Recall is defined as the proportion of the number of correctly classified spam messages to the number of messages originally categorized as spam. Miss Rate is the proportion of wrongly classified legitimate messages to the number of messages originally categorized as legitimate messages.

3.3 Experimental Results

The parameters of our algorithm and their values we adopt in experiments are shown in Table 1 and a legal range for each parameter is also given. Linear kernel is adopted as the kernel function and the weights of classifiers in the window are same. All experiments are conducted on a PC with CPU of AMD Athlon 3200+ and 448M RAM. Classification criteria of Hamming Distance (with and without mutation) and SVM are compared in our experiments.

We choose different combinations of partitions in each corpus to construct training set and testing set, window size is 3 or 5. Figure 1 shows the accuracy, precision and recall on PU1, using partitions 1-2 (219 messages) as training set and partitions 3-10 (880 messages) as testing set, and 5 as window size. M1, M2 and M3 represent classification criterion of Hamming without mutation, Hamming with mutation and SVM, respectively. The miss rates of M1, M2 and M3 are 17.58% , 10.91% and 3.03%, respectively. The average performances of the experiments conducted on different combinations of partitions of corpus are listed in Table 2 to Table 5. Acc, Pre, Rec and MR are abbreviations for accuracy, precision, recall and miss rate, respectively.

Figure 2(a) shows on PU1, the variation of classifier's *Detector Set* and *Memory Set* during its lifetime from initially generated by batch 1 to removed from *window*, using

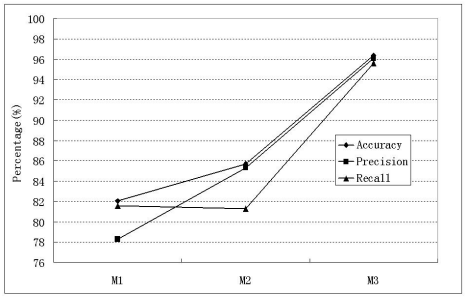


Figure 1. Accuracy, Precision and Recall on Testing Set of partitions 3-10 (880 messages), using partitions 1-2 (219 messages) as Training Set with window size 5

hamming with mutation as classification criterion and 5 as window size. Figure 2(b) shows, for SVM classification criterion, variation of classifier's *support vectors* during its lifetime from initially generated by batch 1 to removed from *window*. Abscissa 1 denotes the generation of the classifier, other abscissas indicate each time when we use *EM-Update* mechanism to update the classifier.

4 Conclusions

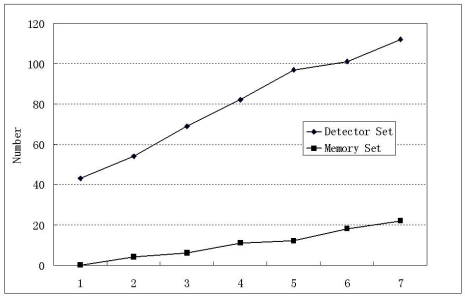
In this paper, intelligent detection approaches based on incremental SVM and AIS are proposed for spam of e-mail stream. Techniques of EM-Update and sliding window are employed to trace the changing of the content of e-mails and user's interests in a continuous way. Hamming Distance(with and without mutation) and SVM which can be regarded as different implementations of continuous detection are developed and tested on corpus PU1 and Ling. Experimental results show that our proposed approaches give impressive performance and will be effective tools in practical applications for spam detection in future.

References

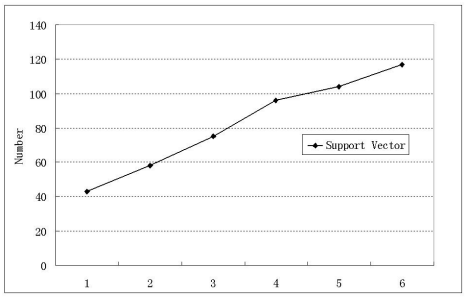
[1] I. Androutsopoulos, J. Koutsias, and K. V. Chandrinos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–167, 2000.

[2] G. B. Bezerra and T. V. Barra. An immunological filter for spam. *International Conference on Artificial Immune Systems (ICARIS 2006), LNCS*, pages 446–458, 2006.

[3] C. Domeniconi and D. Gunopulos. Incremental support vector machine construction. *Proceedings of IEEE International Conference on Data Mining*, pages 589–592, 2001.



(a) Size of Detector Set and Memory Set



(b) Size of Support Vectors

Figure 2. Variation of Detector Set, Memory Set and Support Vectors, using partitions 1-5 (549 messages) as Training Set and partitions 6-10 (550 messages) as Testing Set

[4] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[5] Y. Li, B. Fang, and L. Guo. Research of a novel anti-spam technique based on users's feedback and improved naive bayesian approach. *2006 International Conference on Networking and Services (ICNS 2006)*, pages 16–21, July 2006.

[6] I. Rigoutsos and T. Huynh. Chung-kwei:a pattern-discovery-based system for the automatic identification of unsolicited e-mail messages(spam). *In Proceedings of the first Conference on Email and AntiSpam*, 2004.

[7] I. Stuart, S. H. Cha, and C. Tappert. A neural network classifier for junk e-mail. *Document Analysis Systems VI, Proceedings Lecture Notes in Computer Science*, pages 442–450, 2004.

[8] N. A. Syed, H. Liu, and K. K. Sung. Incremental learning with support vector machines. *International Joint Conference on Artificial Intelligence, (IJ-CAI)*, 1999.

[9] M. W. Wu, Y. Huang, and S. K. Lu. A multi-faceted approach towards spam-resistible mail. *Proceedings. 11th Pacific Rim International Symposium on Dependable Computing (PRDC'05)*, pages 208–218, Dec 2005.

[10] M. H. Zuchini. Aplições de mapas auto-organizáveis emmineração de dados e recuperação de informação. *Master's thesis, UNICAMP*, 2003.