Not So Naïve Online Bayesian Spam Filter

Baojun Su
Institute of Artificial Intelligence
College of Computer Science
Zhejiang University
Hangzhou 310027, China
freizsu@gmail.com

Congfu Xu
Institute of Artificial Intelligence
College of Computer Science
Zhejiang University
Hangzhou 310027, China
xucongfu@zju.edu.cn

Abstract

Spam filtering, as a key problem in electronic communication, has drawn significant attention due to increasingly huge amounts of junk email on the Internet. Content-based filtering is one reliable method in combating with spammers' changing tactics. Naïve Bayes (NB) is one of the earliest content-based machine learning methods both in theory and practice in combating with spammers, which is easy to implement while can achieve considerable accuracy. In this paper, the traditional online Bayesian classifier are enhanced by two ways. First, from theory's point of view, we devise a self-adaptive mechanism to gradually weaken the assumption of independence required by original NB in the online training process, and as a result of that our NSNB is no longer "naïve". Second, we propose other engineering ways to make the classifier more robust and accuracy. The experiment results show that our NSNB does give state-of-the-art classification performance on online spam filtering on large benchmark data sets while it is extremely fast and takes up little memory in comparison with other statistical methods.

Introduction

Spam is an ever-increasing problem. The number of spam emails is increasing daily - Spamhaus estimates that 90% of incoming email traffic is spam in North America, Europe or Australasia. By June 2008, 96.5% of email received by businesses was spam. Added to this, spammers are becoming more sophisticated and are constantly managing to outsmart "static" methods of fighting spam. There are a variety of popular methods for filtering and refusing spam such as DNS-based blackhole lists (DNSBL), greylisting, spamtraps and so on.

The approach of content-based spam filtering has shown promising accuracy and generality in combating with spammers. In content analysis, we regard spam filtering as a special subproblem of binary text classification - the actual message text are taken as input, various machine learning methods are chosen to obtain the results of classification. Content-based filtering has already been widely deployed in a majority of real spam filtering systems.

Existing Problems

It is proved that content analysis could help the system to get more accurate prediction, but there is a controversy in choosing learning algorithms. Most of existing applied systems (e.g. Bogofilter, Spambayes, Spamassassin) integrate NB (Graham 2003) as learning algorithm, because NB is easy to code, lightweight, fast, and able to adapt to a particular person or organization. On the other hand, researchers have developed various learning methods. These methods, especially Support Vector Machines (SVMs) (Drucker, Wu, and Vapnik 1999), Logistic Regression (LR) (Goodman and tau Yih 2006), and Dynamic Markov Compression (DMC) (Bratko and B.Filipic 2005) claim much higher precision than NB. The NB's advocators, in turn, complain about the huge time or memory consuming of these new methods. For example, SVMs typically require training time that is quadratic in the number of training example, so they are not so practical for large-scale email systems; DMC typically requires 2GB memory, so it is a heavy burden to applied email systems.

Contributions

In this paper, we demonstrate an improved version of online Bayesian classifier, which can exceed or near the record on several large benchmark data sets of spam email at the same time holds the precious property of the original NB: lightweight, and fast.

Besides, the algorithm behind our system is self-adaptive to get rid of the hypothesis of independence gradually, which do not use the complex graph model like Bayesian network. So it can be used in a more generalized background, where Bayes' theorem works but is difficult to calculate directly.

Preliminaries

A Naïve Bayes filter estimates the joint probability over a set of features $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ and a class C by making the "naïve" assumption that each feature is conditionally independent,

$$P(\mathcal{X}, C) = P(C) \prod_{i=1}^{n} P(X_i|C). \tag{1}$$

And in the context of spam filtering, take advantage of Bayes' theorem, we can estimate the email's spam likelihood by

$$P(\text{spam}|\mathcal{X}) = \frac{P(\text{spam}) \prod_{i=1}^{n} P(X_i|\text{spam})}{P(\mathcal{X})}.$$

Likewise,

$$P(\text{ham}|\mathcal{X}) = \frac{P(\text{ham}) \prod_{i=1}^{n} P(X_i|\text{ham})}{P(\mathcal{X})}.$$

We do not need to calculate $P(\mathcal{X})$, if we examine the quotient of the two above formulas, and note $P(\text{spam}|\mathcal{X}) + P(\text{ham}|\mathcal{X}) = 1$, then

$$\frac{P(\text{spam}|\mathcal{X})}{1 - P(\text{spam}|\mathcal{X})} = \frac{P(\text{spam}) \prod_{i=1}^{n} P(X_i|\text{spam})}{P(\text{ham}) \prod_{i=1}^{n} P(X_i|\text{ham})}, \quad (2)$$

if the odds exceed some threshold, then the email is classified as spam.

In practice, $P(\text{spam})$ and $P(\text{ham})$ are estimated by the spam and ham email's frequency respectively, while $P(X_i|\text{spam})$, and $P(X_i|\text{ham})$ are estimated by feature X_i's frequency in each category. The detail of how to calculate frequencies varies among different variations of NB (Metsis, Androutsopoulos, and Paliouras 2006). In most cases, there are zeros among $P(X_i|\text{ham})$, so a smooth parameter λ is always introduced to avoid division by 0 (Graham 2002).

Online Naïve Bayes

To make spam filter practical, the filter must have the ability to adapt to spammers' changing tactics. So spam filtering is typically deployed and tested in an online setting, which proceeds incrementally (Cormack and Lynam 2006). The filter classifies a new example, is told if the prediction is correct, updates the model accordingly, and then awaits a new example.

Email Representation

A plenty of methods have been developed to extract features from text, especially when the text may include meta contents in header information like email. And different methods lead to different performances. Simply tokenization is most popular, but cannot obtain good results for spammer's intentional obfuscation or good words attack. OSBF-LUA introduced a complicated feature extraction algorithm called OSBF (Siefkes et al. 2004), which won the TREC 2006 spam track competition also needs to tokenize the email as first step. But in some other languages e.g. Chinese, tokenization itself is a very hard problem. Here we adopted a very simple but competitive method, n-gram. An n-gram vector is a vector x with a unique dimension for each possible substring of n total characters. No feature selection is used. And the feature vector was constructed by binary representation which indicates if a feature does not occur in this document, the according weight is 0, and if a feature occurs in this document, the according weight is 1.

Not So Naïve Bayes (NSNB)

In traditional NB, the calculation accuracy losses greatly in equation (1), because it is based on the hypothesis that *all features are conditionally independent*, which is far from the fact obviously. Though the results are always better than expected, we can imagine even better results if independent hypothesis can be get rid of. Some researchers (Friedman, Geiger, and Goldszmidt 2007) have brought Bayesian Network in the general classification problem which no longer needs the rigorous independent hypothesis, and consequently obtain convincing results. But in the context of spam filtering, or general text classification problems, there are typically 10^4 to 10^6 features, that it is impractical to solve such a huge graph.

Here we demonstrate a simple yet practical trick to relieve conditional independence to some degree. Consider the joint probability, and try to expand it using the chain rule in order.

$$\begin{aligned} P(\mathcal{X}|C) &= P(X_1, X_2, \ldots, X_n|C) \\ &= P(X_1|C)P(X_2, X_3, \ldots, X_n|X_1, C) \\ &= P(X_1|C)\theta(\mathcal{X}, X_1, C)P(X_2, X_3, \ldots, X_n|C), \end{aligned}$$

where θ is an unknown but does existing distribution. In general, a notation $\mathcal{B}_i$ is introduced to represent the *set* of features expanded before feature X_i. We have

$$P(\mathcal{X}|C) = \prod_{i=1}^{n} P(X_i|C)\theta(\mathcal{X}, C, X_i, \mathcal{B}_i).$$

Notice that the above equation is accurate, not an approximation. The problem lies how to calculate θ, which is trivially set to 1 in traditional NB.

We can see that it is nearly impossible to calculate θ directly, as there exists too much uncertainty. Especially for $\mathcal{B}_i$, we can only illustrate it in theory. In order to approximate θ, here we also make some parameter reduction (but not all of them are ignored), that we approximate $\theta(C, X_i)$

$$P(\mathcal{X}|C) = \prod_{i=1}^{n} P(X_i|C)\theta(C, X_i). \quad (3)$$

It is not accurate now, but is still a much better approximation to $P(\mathcal{X}|C)$ than equation (1). In practice, instead of calculating $\theta(C, X_i)$ directly, we devise an online response mechanism to let a given $\theta(C, X_i)$ change by itself. Added to original NB, we give every feature two parameters, ham_confidence, and spam_confidence, and for a given feature f, the ham_confidence approximates $\theta(ham, f)$, while the spam_confidence approximates $\theta(spam, f)$. Before online learning, all the confidence parameters are set to 1, if mistake occurs, the system deduces the corresponding confidence parameter. In a single classification process, we use equation (3) instead of equation (1).

To control changes of the confidence parameters, we introduce another two supportive parameters α and β, which are called learning rate. If a ham mail is misclassified, then

$$\text{ham\_confidence} = \text{ham\_confidence} \times \beta,$$

while if a spam mail is misclassified, then

$$\text{spam\_confidence} = \text{spam\_confidence} \times \alpha.$$

In practice, we only store one joint confidence factor

$$\mathbf{cf} = \frac{\text{spam\_confidence}}{\text{ham\_confidence}}.$$

And now, in the online process, if a ham mail is misclassified, then

$$\mathbf{cf} = \mathbf{cf}/\beta,$$

while if a spam mail is misclassified, then

$$\mathbf{cf} = \mathbf{cf} \times \alpha.$$

We see the approximation and adaption of the confidence factors are quite rough, but we all know that standard NB empirically can achieves considerable strong results even directly neglect features' probability relations which may be potentially important in classification. Here we do not need to follow the hypothesis of independence of features and in return, we do obtain more stronger results on large benchmark corpus (Table 1).

	Spamassassin	CRM114	Bogofilter	NSNB
score	0.0590	0.0420	0.0480	0.0079

Table 1: Results on TREC 2005 Spam Track corpus, while the first three filters are traditional Naïve Bayes filters. Score reported is (1-ROCA)%, where 0 is optimal.

Besides the introduction of confidence factor and learning rate which boosts the traditional NB from theory's point of view, we also propose three other engineering ways to build a more robust and accuracy classifier:

- Unlike traditional smoothing technique, we choose a smooth parameter small enough.
- Introducing the logistic function and a scale parameter to scale the results.
- Using thick threshold to adjust learning model.

Smoothing

As mentioned above, in our algorithm, we modified equation (2) as

$$\frac{P(\text{spam}|\mathcal{X})}{1 - P(\text{spam}|\mathcal{X})} = \frac{P(\text{spam}) \prod_{i=1}^{n} P(X_i|\text{spam})\theta(\text{spam}, X_i)}{P(\text{ham}) \prod_{i=1}^{n} P(X_i|\text{ham})\theta(\text{ham}, X_i)}.$$

In order to avoid underflow arisen from too many multiply operations, we take the logarithm, and log odds can be represented as following

$$\log \frac{P(\text{spam})}{P(\text{ham})} + \sum_{i=1}^{n} \log \frac{P(X_i|\text{spam})\theta(\text{spam}, X_i)}{P(X_i|\text{ham})\theta(\text{ham}, X_i)}. \quad (4)$$

To estimate $P(\text{spam})$ as

$$P(\text{spam}) = \frac{N(\text{spam})}{N(\text{spam}) + N(\text{ham})},$$

where $N(\text{spam})$ stands for number of known spams, $P(\text{ham})$ can be calculated in a similar way. And to estimate $P(X_i|\text{spam})$ as

$$P(X_i|\text{spam}) = \frac{N(\text{spam email includes } X_i)}{N(\text{spam})},$$

so we have

$$\frac{P(X_i|\text{spam})}{P(X_i|\text{ham})} = \frac{N(\text{spam email includes } X_i)N(\text{ham})}{N(\text{ham email includes } X_i)N(\text{spam})}.$$

In fact some features only appear in spam email, so the above equation is not always calculable. By applying the *lidstone's law*, we introduced a smooth parameter ε, and instead of calculating the above equation, we calculate

$$\frac{N(\text{spam email includes } X_i) + \varepsilon}{N(\text{ham email includes } X_i) + \varepsilon} \cdot \frac{N(\text{ham}) + 2\varepsilon}{N(\text{spam}) + 2\varepsilon}.$$

Traditionally the smooth parameter ε is set to 1 or 0.1, while we found 1.0×10^{-5} is a much better choice (Figure 1). It is reasonable because the smaller the ε is, the more precise the approximation is, when ε is 0, it is exactly the original problem. But on the other hand, if ε is set too small, the result will be unstable.

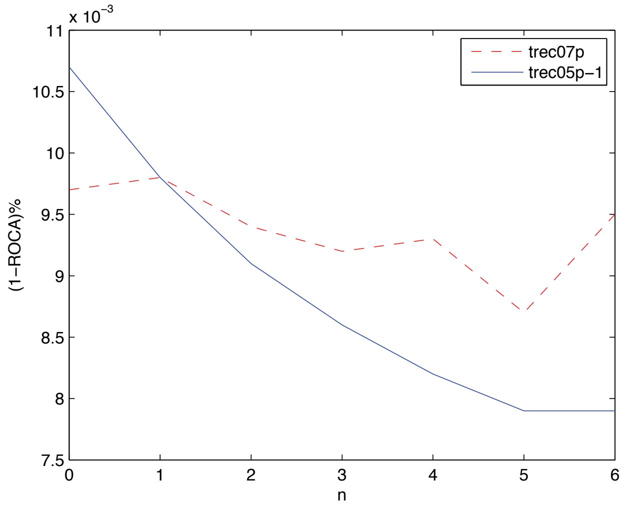

Figure 1: The smooth parameter $\varepsilon = 10^{-n}$. We can see $n = 5$, i.e. $\varepsilon = 10^{-5}$ is optimal.

Result Scale

After calculating equation (4) and obtaining the result p_{prime}, we now have a primary judgement on a coming mail. $p_{prime} \in \mathcal{R}$, and if $p_{prime} \in \mathcal{R}_{+}$, the mail is judged to be spam or judged to be ham. But it is not over, in most of the time, the probability of spam is needed. So we should take some scale technique to map $\mathcal{R}$ onto $(0, 1)$, here we choose the logistic function:

$$logistic(x) = \frac{1}{1 + e^{-x}}$$

Because $|p_{prime}|$ is always too large that calculated p is very close to 0 or 1, which may depress calculation accuracy (we can address this issue from Figure 2). At the same time, it is unsuitable for thick threshold retraining. So here we use a so called *scale parameter* to prevent $|p_{prime}|$ going too large, which also holds the mail's relative position on axis. In practice, the scale parameter is set to 2500. So, we calculate the final score by

$$p = logistic(\frac{p_{prime}}{\text{scale parameter}}). \quad (5)$$

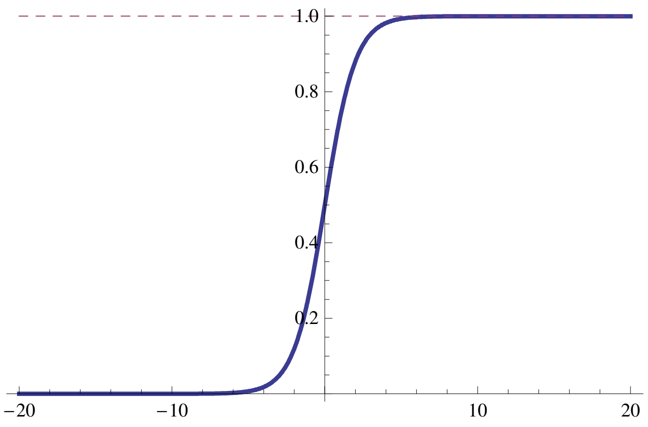

Figure 2: Logistic Function. We see points far from y-axis are unreliable, that they are too close to the two parallel lines $y = 0$ and $y = 1$.

Thick Threshold

In our implementation, we use a thick threshold for learning. Training instances are retrained even if the classification is correct when the determined score is near the threshold. Two additional positive parameter ϵ^+ and ϵ^- are chosen to define an interval $(0.5-\epsilon^-, 0.5+\epsilon^+)$, that if the predict spam probability $P(\text{spam}|\mathcal{X})$ falls in it, we regard the email is not well classified, and should modify the Bayesian model accordingly. After using modified model to give a new score, the classifier repeat the thick threshold procedure until the email is well classified. In this way, a large margin classifier will be trained that is more robust when classifying borderline instances. In NSNB, we set $\epsilon^- = \epsilon^+ = 0.25$.

Online Model Adjusting

Here we will illustrate the whole online precess of NSNB. In the beginning we establish two hash tables. One stores the number of features' occurrence in spam email, while the other stores the number of features' occurrence in ham email. All features' confidence factors are set to 1 initially. When an email comes, we obtain the email's vector form $\mathcal{X}$ by feature extraction at first, and then calculate $\mathcal{X}$'s spam prime probability P_{prime} using equation (4) with smoothing technique. After result scale, we obtain the final score P using equation (5). If $P \leq 0.5$, the email is considered to be ham, and if $P > 0.5$, the email is considered to be spam. In an optimal situation, we get the feedback immediately. Then if the email is misjudged, or $P \in (0.5 - \epsilon^-, 0.5 + \epsilon+)$ (by applying thick threshold), we just add all features in $\mathcal{X}$ to the corresponding hash table, and set all the new features' confidence factor to 1, meanwhile the confidence factor of all features in $\mathcal{X}$ are going to be adjusted by multiplying or dividing by learning rate depends on the email's class. So the model now has been adjusted, and we repeat the above process until the email is *well classified* (obtained right prediction and $P \notin (0.5-\epsilon^-, 0.5+\epsilon^+)$). Then the filter awaits another email to come until all the emails are classified. The pseudo-code of whole online process is shown as Algorithm 1.

```
Input: Dataset $U = (\mathcal{X}_1, y_1), \ldots, (\mathcal{X}_n, y_n)$, learning
       rate $\alpha$, $\beta$, thick threshold $\epsilon$
Set all features' cf to 1;
for each $\mathcal{X}_i \in U$ do
    calculate the score $p_i$;
    if $p_i > 0.5$ then
        report spam;
    end
    else
        report ham;
    end
    while $y_i$ is ham and $p_i > (0.5 - \epsilon^-)$ do
        train $\mathcal{X}_i$;
        for all features in $\mathcal{X}_i$ do
            feature's cf $\times =$ $\alpha$;
        end
    end
    while $y_i$ is spam and $p_i < (0.5 + \epsilon^+)$ do
        train $\mathcal{X}_i$;
        for all features in $\mathcal{X}_i$ do
            feature's cf $/ =$ $\beta$;
        end
    end
end
```

Algorithm 1: Pseudo-code for online learning

Experiment Results

We have demonstrated our new Bayesian online spam detection algorithm, which is more reliable than traditional Naïve Bayes theoretically. In this section we conduct a series of experiments on several benchmark email corpus and compare with other machine learning algorithms or applied systems.

Experimental Setting

We conduct several experiments on four email datasets for spam filtering. They are 2005-2007 TREC Spam Filtering Track public datasets, which have also been used frequently in previous works (TREC 06 has two corpus - TREC 06p is an English corpus, TREC 06c is a Chinese corpus).

As mentioned in Preliminaries, in our system, we use 5-gram for feature extraction, and no other feature selection is used. To determine features' weights, binary feature scoring is used, which has been shown to be most effective for a variety of spam detection methods. Furthermore, with email data, we reduce the impact of long message by considering only the first 2000 characters of each header and body. Same features in email's header and body are used distinctively.

We first make some tests on these corpus using our algorithm with different parameter setting, and try to find a series of parameters which maximize the Area under ROC curve, which will be used in the later experiments.

In order to certify our method's reliability, we compare our results to TREC's winner each year. All the TREC participants' results are gathered from official TREC's overview (Cormack and Lynam 2005) (Cormack 2006) (Cormack and Lynam 2007). And we also compared both running time and (1-ROCA)% with ROSVM (Sculley and Wachman 2007), which claims to give state-of-the-art performance on email spam. The results show that our algorithm are more practical.

Though spam filtering is a binary text classification problem, the accuracy of the filter at the 50-50 decision point is usually not a good performance metric. Because the cost of losing legitimate mail is much higher than receiving spam. We choose ROC curve (Fawcett 2004) as a standard of comparison, which is a typical method for evaluating spam filters.

Our program is coded purely in python. All the experiments are done on a typical PC with Core 2 Quad Q6600 and 4GB memory, which runs Ubuntu 8.10 as operating system and python 2.52 as the interpreter. We use the standard Linux shell commands *time* and *top* to get the program's running information. The online procedure is simulated by the canonical order with each of these corpus for fair comparison, and the feedback is given immediately after a mail has been classified. Notably, we do not use any other email source to pre-train the filter before any single test.

Learning Rate

In this experiment, we tune the learning rate $\alpha, \beta \in (0, 1]$ to different numbers, and pick up a set of $\{\alpha, \beta\}$ which minimize the (1-ROCA)%. To keep problem simple, we only consider the situation where $\alpha = \beta$. When α, β are set to 1, the algorithm regress to the original NB.

$\alpha = \beta$	Corpus	(1-ROCA)%	Time	Memory
1.0	TREC 05p	0.0229	$104m6s$	$590mb$
0.65	TREC 05p	0.0079	$18m27s$	$155mb$
1.0	TREC 06p	0.0684	$24m34s$	$289mb$
0.65	TREC 06p	0.0344	$7m27s$	$147mb$
1.0	TREC 06c	0.0229	$24m26s$	$533mb$
0.65	TREC 06c	0.0004	$10m16s$	$150mb$
1.0	TREC 07p	0.0112	$48m35s$	$562mb$
0.65	TREC 07p	0.0087	$24m54s$	$111mb$

Table 2: To keep the table readable (not too long), here we only illustrate $\alpha = \beta = 0.65$ which is optimal throughout the experiment. "Memory" is for the maximum memory consumed in the online training process. We can see the addition of confidence factor *drastically deduced* the total running time and memory consuming while obtaining better results.

Comparison with TREC Winners

In this experiment, we gather the results of TREC winners from the official overview to draw a comparison with our NSNB.

Winner of TREC 2005 (Bratko and Filipic 2005) used the character-level Markov models for spam filtering, which is adaptive statistical data compression model. Such models use statistics obtained from some training corpus to predict the probability of the next character in a stream of text.

Winner of TREC 2006's English corpus (Assis 2006) is an original NB classifier while used very skillful methods of feature extraction and feature selection to enhance the result. We should point here, unlike the OSBF-Lua (06 winner), we do not use *any* other priori knowledge before training.

Winner of TREC 2006's Chinese corpus (Sculley, Wachman, and Brodley 2006) used the perceptron algorithm with margins, which is an online linear classifier using inexact string matching in explicit feature space. The result they have got is near perfect, but we did even better.

Winner of TREC 2007 used a simple yet promising algorithm of Logistic Regression described in the paper of CEAS 2006 (Goodman and tau Yih 2006), and they obtained sound results on TREC 07 corpus.

	TREC 05p	TREC 06p	TREC 06c	TREC 07p
winner	0.0190	0.0540	0.0023	0.0055
NSNB	0.0079	0.0344	0.0004	0.0087

Table 3: Score reported is (1-ROCA)%, where 0 is optimal. Here we only list the results of immediate feedback. We have obtained much better results than TREC winners except for TREC 07.

Comparison with ROSVM

ROSVM (Sculley and Wachman 2007) is for Relaxed Online SVMs, which claims to hold state-of-the-art spam classification performance while is an order of magnitude more efficient than the normal Online SVM. We will show our algorithm can achieve similar if not better results, while is much more efficient than it.

	Corpus	(1-ROCA)%	Time	Memory
ROSVM	TREC 05p	0.0090	$6h52m$	-
NSNB	TREC 05p	0.0079	$18m27s$	$155mb$
ROSVM	TREC 06p	0.0240	$5h9m$	-
NSNB	TREC 06p	0.0344	$7m27s$	$147mb$
ROSVM	TREC 07p	0.0093	-	-
NSNB	TREC 07p	0.0087	$24m54s$	$111mb$

Table 4: Our NSNB beats ROSVM algorithm on most corpus except for TREC06p, and most significantly it is *20 to 40 times* faster than ROSVM.

Discussion

The comparison results shown in Tables 3,4 are striking in two ways. First, they show the performance of our NSNB can match or even exceed records on these large public benchmark corpus held by other sophisticated machine learning methods. Second, they show a dramatic disparity in computational cost. NSNB could handle tens of thousands of emails typically less than 20 minutes, and after that it only consumed around 150mb memory, which is so lightweight and fast that dramatically suitable for practical application. Above all, by now we do not use any pre-training technique like database which is widely used by other Bayesian classifiers, and we do find that most errors occur in the very beginning of online process, so we can imagine an even better accuracy of classification in practice if some received emails are labeled before the system is put into service. Interestingly, Experiment results show that NSNB has a surprising classification ability on Chinese corpus, that it makes less than 50 mistakes on TREC06c corpus (total emails : 64620) and less than 40 mistakes on SEWM08 corpus (total emails : 69300). We will try to find out the exact reason in our later work.

Conclusion and Future Work

We enhanced traditional NB by introducing a confidence factor to each feature, and in the online process, the confidence factor is changed when an email is not well classified. Throughout this process, the Naïve Bayes model gradually becomes to be not so naïve, i.e. we no longer need the conditional independent hypothesis after sufficient iterations. In the experiments, we can see the introduction of the confidence factor does brought better results, less time and memory consumption. This technique can also be used in other situations where Bayes' theorem holds but the result is hard to calculate as the relation of variables is too complex. During the online process, we also developed other techniques such as smoothing, result scale, and thick threshold to enhance the final results. The experiment results show our NSNB is very reliable for the attractive sound results yet runs *extremely fast* with much *less memory* consumption. Though in TREC06p corpus, and TREC07p corpus, the results obtained by NSNB are not the best on records, but they are close to the best results.

We do not use other material to pre-train the NSNB before the online process, and then lots of mistake occurs in the very beginning of online training process. We will try to resolve this problem and it is uncertain if our system is robust enough against the *good word attack* (Wittel and Wu 2004), that we must do more experiments on this matter.

Acknowledgements

This paper is supported by the National High Technology Research and Development Program of China (863 plan, No. 2007AA01Z197). We give thanks to Weike Pan, Guanzhong Lu, Peng Peng, and Juxin Liu for their great work in the design and development of our spam filtering system.

References

Assis, F. 2006. Osbf-lua - a text classification module for lua, the importance of the trainning method. In *The Fifteenth Text REtrieval Conference (TREC 2006)*.

Bratko, A., and B.Filipic. 2005. Spam filtering using compression models. *Technical Report IJS-DP-9227*.

Bratko, A., and Filipic, B. 2005. Using character-level markov models for spam filtering experiments for the trec 2005 spam track. In *The Fourth Text REtrieval Conference (TREC 2005)*.

Cormack, G. V., and Lynam, T. R. 2005. Trec 2005 spam track overview. In *The Fourteenth Text REtrieval Conference (TREC 2005)*.

Cormack, G. V., and Lynam, T. R. 2006. On-line supervised spam filter evaluation. http://plg.uwaterloo.ca/?gvcormac/spamcormack.html.

Cormack, G. V., and Lynam, T. R. 2007. Trec 2007 spam track overview. In *The Sixteenth Text REtrieval Conference (TREC 2007)*.

Cormack, G. V. 2006. Trec 2006 spam track overview. In *The Fifteenth Text REtrieval Conference (TREC 2006)*.

Drucker, H.; Wu, D.; and Vapnik, V. N. 1999. Support vector machines for spam categorization. *Neural Networks*.

Fawcett, T. 2004. Roc graphs: notes and practical considerations for researchers.

Friedman, N.; Geiger, D.; and Goldszmidt, M. 2007. Bayesian network classifiers. *Machine Learning*.

Goodman, J., and tau Yih, W. 2006. Online discriminative spam filter training. In *Third Conference on Email and Anti-Spam (CEAS 20006)*.

Graham, P. 2002. A plan for spam.

Graham, P. 2003. Better bayesian filtering.

Metsis, V.; Androutsopoulos, I.; and Paliouras, G. 2006. Spam filtering with naive bayes - which naive bayes? In *Third Conference on Email and Anti-Spam (CEAS 2006)*.

Sculley, D., and Wachman, G. M. 2007. Relaxed online svms for spam filtering. In *SIGIR'07*.

Sculley, D.; Wachman, G. M.; and Brodley, C. E. 2006. Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. In *The Fifteenth Text REtrieval Conference (TREC 2006)*.

Siefkes, C.; Assis, F.; Chhabra, S.; and Yerazunis, W. S. 2004. Combining winnow and orthogonal sparse bigrams for incremental spam filtering. In *PKDD 2004*.

Wittel, G. L., and Wu, S. F. 2004. On attacking statistical spam filters. In *First Conference on Email and Anti-Spam (CEAS 2004)*.