BIN WANG[1,2] **GARETH J.F. JONES**[2] **WENFENG PAN**[1]

Using Online Linear Classifiers to Filter Spam Emails

[1] *Institute of Computing Technology, Chinese Academy of Sciences, China*

[2] *School of Computing, Dublin City University, Ireland*

Email: wangbin@ict.ac.cn gareth.jones@computing.dcu.ie panwenfeng@software.ict.ac.cn

Tel: +86-10-62565533 +353-1-7005559 +86-10-62565533

Fax: +86-10-62567724 +353-1-7005442 +86-10-62567724

This work was carried out while the first author was visiting Dublin City University supported by a China State Scholarship.

Abstract The performance of two online linear classifiers - the Perceptron and Littlestone's Winnow – is explored for two anti-spam filtering benchmark corpora - PU1 and Ling-Spam. We study the performance for varying numbers of features, along with three different feature selection methods: Information Gain (IG), Document Frequency (DF) and Odds Ratio. The size of the training set and the number of training iterations are also investigated for both classifiers. The experimental results show that both the Perceptron and Winnow perform much better when using IG or DF than using Odds Ratio. It is further demonstrated that when using IG or DF, the classifiers are insensitive to the number of features and the number of training iterations, and not greatly sensitive to the size of training set. Winnow is shown to slightly outperform the Perceptron. It is also demonstrated that both of these online classifiers perform much better than a standard Naïve Bayes method. The theoretical and implementation computational complexity of these two classifiers are very low, and they are very easily adaptively updated. They outperform most of the published results, while being significantly easier to train and adapt. The analysis and promising experimental results indicate that the Perceptron and Winnow are two very competitive classifiers for anti-spam filtering.

Keywords *Online Linear Classifier, Perceptron, Winnow, Anti-Spam Filtering*

1 Introduction

Spam emails are unsolicited messages that the receiver does not wish to receive. Increasingly today large volumes of spam emails are causing serious problems for users, Internet Service Providers, and the whole Internet backbone. It was reported that in May 2003 the amount of spam exceeded legitimate emails [1]. This means that more than 50% of transmitted emails are spam. Spam emails not only waste resources such as bandwidth, storage and computation, but also the time and energy of email receivers who must search for legitimate emails among the spam and take action to dispose of spam. As a result they can have a serious economic impact on companies. For example, a Ferris Research Report estimates that US companies spent $10 billion in lost productivity in 2003 due to spam emails [2]. Several proposals have been applied to reduce spam emails, ranging from technical to regulatory and economic [3]. In this paper we concentrate on technical approaches to addressing the problem of spam emails, specifically we explore automated anti-spam filtering methods.

From the perspective of a computer science researcher, especially a machine learning researcher, anti-spam filtering can be regarded as a binary Text Classification (TC) problem. The classifier must distinguish between legitimate emails and spam. However, anti-spam filtering is different from standard TC problems in at least the following four aspects:

1. The classes in anti-spam filtering are less topic-related than those in standard TC problem. Spam email is not based on definable topic classes. For instance, classes in standard TC, such as sports, education are related to specific definable subjects. This difference means anti-spam filtering may be more difficult than classifying more topic-related classes.
2. Because in a real email system a large volume of emails often need to be handled in a short time, efficiency will often be as important as effectiveness when implementing an anti-spam filtering method.
3. Some email receivers may pay more attention to the precision of the filtering. Some receivers would rather have a spam message be judged as legitimate, rather than a legitimate message judged as spam. This means different types of errors may have different costs in anti-spam filtering [4].
4. The contents of both the legitimate and spam email classes may change dynamically over time, so the anti-spam filtering profile should be easily

updatable to reflect this change in class definition. In this sense, anti-spam filtering is rather more like a dynamic adaptive filtering task than a static TC task.

Thus, an effective and efficient classifier, which can be easily and effectively updated, is the goal for anti-spam filtering. This paper is an effort towards this goal. Two online linear classifiers: the Perceptron and Winnow are investigated in our experiments. The results show that they are very competitive classifiers for anti-spam filtering tasks.

The remainder of the paper is organized as follows: Section 2 outlines relevant previous research in anti-spam filtering, Section 3 introduces the two linear classifiers used in our study: the Perceptron and Winnow, Section 4 describes our experiments, including details of the test collections, measures and experimental results, and finally our conclusions and future work are given in Section 5.

2 Related work

As we pointed out in Section 1, anti-spam filtering can be regarded as a specific type of TC problem. Many machine learning approaches have been explored for this task. For example rule-based methods, such as Ripper [5], PART, Decision tree, and Rough Sets, have been used in [6], [7] and [8]. However, pure rule-based methods haven't achieved high performance because spam emails cannot easily be covered by rules, and rules don't provide any sense of degree of evidence. Statistical or computation-based methods have proven more successful, and are generally adopted in mainstream work. Bayesian classifiers are the most widely used method in this field. Sahami *et al.* used Naïve Bayes with an unpublished email test collection [9]. In their work some non-textual features (e.g., the percentage of non-alphanumeric characters in the subject of an email) were found to improve the final performance. Following this work, many researchers including Androutsopoulos and Schneide applied Naïve Bayes in anti-spam filtering, their work is reported in [4][10][11][12] and [13]. Androutsopoulos *et al.* [10] found Naïve Bayes to be much better than a keyword-based method. Another of their findings was that Naïve Bayes performed comparably to a kNN classifier. Schneide *et al.* [13] compared two event models of Naïve Bayes: Multi-variate Bernoulli Model (MBM) and Multinomial Model (MM). They found that the two

models performed similarly over a large corpus, but that the MBM model outperformed the MM model over a smaller corpus, while MBM's computation is simpler.

In addition to the Naïve Bayes model, other Bayesian models have also been considered for anti-spam filtering. Androutsopoulos *et al.* [12] introduced a flexible Bayes model which is one kind of model for continuous valued features. Their experiments showed that the flexible Bayes model outperforms Naïve Bayes. Mertz[1] used an N-gram language model to evaluate the probabilities in a Bayesian model using an unpublished corpus. He found that a tri-gram model is the best choice when compared to Naïve Bayes and bi-gram Bayesian models.

Besides Bayesian methods, other machine learning methods, including Support Vector Machine (SVM), Rocchio, kNN and Boosting, have also been applied in the context of anti-spam filtering. Some interesting work based on Boosting approaches is described in [14] which applied Adaboost for filtering spam emails, and [12] which used another Boosting approach called LogitBoost. These boosting methods both achieved very good results on their evaluation tasks. A decision tree was introduced as a weak learner on both boosting methods. Other work has been reported based on kNN [11], SVM and Rocchio [6].

Because most experimental results have been achieved using different corpora and some corpora are not publicly available, it is difficult to compare them objectively. To address this problem, Androutsopoulos *et al.* developed a number of publically available corpora, e.g. PU1 and Ling-Spam, that can be freely downloaded from their web site[2], and subsequently many experiments have been reported using these corpora [4][7][10][11][12][13][14].

To the best of our knowledge, few online linear classifiers have been considered in the area of spam filtering, and the filter update process was always omitted in the above work. These reasons motivate us to explore the application of online linear classifiers featuring very easy and fast updates to the task of anti-spam filtering.

[1] See http://www-900.ibm.com /developerWorks /cn/linux/other/l-spamf/index_eng.shtml.
[2] http://iit.demokritos.gr/skel/i-config/downloads/

3 The Perceptron and Winnow

According to whether the decision surface in the feature space is a hyperplane or not, classifiers can be divided into two categories: linear classifiers and nonlinear classifiers. The most widely used classifiers are linear models. A linear classifier represents each class c_i with a class weight vector $\mathbf{w}_i$ and a threshold θ_i ($i=1,2,3,\ldots,|C|$), where C is the category set and |C| is the number of different categories. If a document d (represented with $\mathbf{d}$ in the same vector space as $\mathbf{w}_i$) satisfies $\mathbf{w}_i \bullet \mathbf{d} > \theta_i$, it belongs to[3] class c_i. Here, $\bullet$ means the internal product of two vectors. Particularly, in a two-class linearly separable situation (see Figure 1), a linear classifier seeks to find a hyperplane that correctly classifies all the examples. Formally, let the N training examples be $<\mathbf{x}_i, y_i>$, where $1 \leqslant i \leqslant N$, $\mathbf{x}_i$ is the k-dimension vector representation of the ith example, k is the number of features, y_i=+1 or –1, respectively means a positive or negative example. Then, a linear classifier can be represented by a k-dimension weight vector $\mathbf{w}$ and a scalar θ, for each i which satisfies

$$\begin{cases} \mathbf{w} \bullet \mathbf{x}_i - \theta \geq 0 & \text{if } y_i = +1 \\ \mathbf{w} \bullet \mathbf{x}_i - \theta < 0 & \text{if } y_i = -1 \end{cases} \qquad (1)$$

Formula (1) can be rewritten as

$$y_i(\mathbf{w} \bullet \mathbf{x}_i - \theta) \geq 0 \qquad (2)$$

[3] Here, a document can belong to more than one class. If a document is restricted to only one class, it belongs to the class with the highest score $\mathbf{w} \bullet \mathbf{d}$. The latter situation is used in our two-class case, which is suitable for anti-spam filtering.

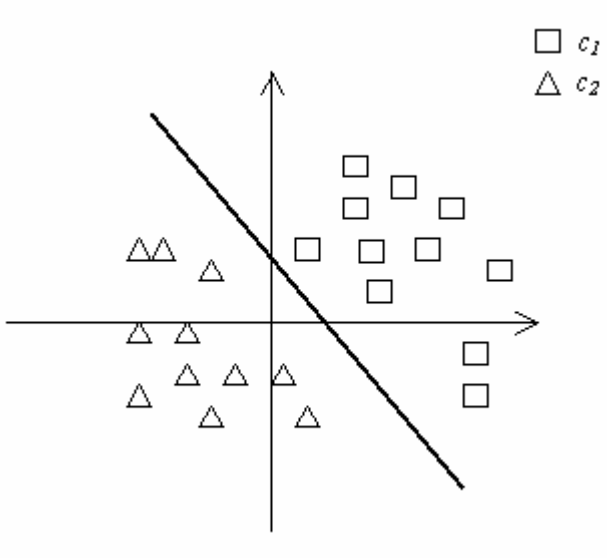

Fig. 1 Linearly separable examples (the solid line represents the discrimination hyperplane) of two classes: c_1 and c_2.

So linear classifiers try to find suitable values for **w** and θ that correctly separate all the training examples. Obviously, if they exist, the separate hyperplane is $\mathbf{w} \bullet \mathbf{x}_i - \theta = 0$. From Figure 1, we can also see that there is not a single discrimination hyperplane. A hyperplane with a large margin is selected in large margin linear classifiers such as SVMs. However, because there are few linear separated problems in the real world, the above problem is usually approximately solved by different optimization criteria. For example, the Least Mean Square (LMS) criterion leads to classifiers such as Widrow-Hoff [15], the "*minimum distance to class centroid*" criterion leads to Rocchio [16], and the minimum error criterion leads to the Perceptron [17] and Winnow [18]. The Perceptron and Winnow seek to find the **w** and θ values which generate minimum errors on training examples. Apart from Rocchio, the other three linear classifiers are online classifiers that adjust the weight vector after training each example. Online classifiers are appropriate for applications that start with few training examples or incrementally adjust the weight vector based on user feedback. Adaptive filtering is thus a task ideally suited to the use of online classifers.

The Perceptron was developed by Rosenblatt [17] in the late 1950's and early 1960's, while Winnow was introduced by Littlestone [18] in the late 1980's. The basic idea of both of these classifiers is to update the vector **w** driven by classification errors, which is why they are called error-driven methods. The implementation process of the Perceptron or Winnow are composed of four steps:

Step 1: Initiate $\mathbf{w} = \mathbf{w}_0$, a fixed threshold θ, the maximum number of iterations n, and let the iteration counter $j = 1$, s =0. The number of retained features is k, the number of training messages is N.

Step 2: In the jth iteration, for each training example $<\mathbf{x}_i, y_i>$ (i = 1,2,...,N), if $(\mathbf{w}_s \bullet \mathbf{x}_i + \theta) y_i < 0$, which means an error occurs, then $\mathbf{w}_{s+1} = update(\mathbf{w}_s, \mathbf{x}_i, y_i, \theta)$ and $s = s + 1$, where *update*() is an update function. Record the number of errors in the jth iteration, e_j. Let $j = j + 1$.

Step 3: If $j > n$, exit iteration and go to Step 4, else go to Step 2.

Step 4: Select the $\mathbf{w}$ with minimum e. The $\mathbf{w}$ after the last training example in the iteration is chosen in our experiments.

The difference between the Perceptron and Winnow is that they use different types of update function. In particular, the Perceptron uses an additive update function while Winnow uses a multiplicative one. When a positive example (y_i = +1) is incorrectly regarded as negative (denoted with type A^+ error) or a negative example (y_i = -1) regarded as positive (denoted with type A^- error). The basic Perceptron classifier will update the weights as follows:

$$\mathbf{w}_{s+1} = \mathbf{w}_s + \eta \mathbf{x}_i y_i \qquad (3)$$

While the function for Winnow is

$$\mathbf{w}_{s+1,r} = \mathbf{w}_{s,r} \times \delta^{y_i} \quad \text{if } \mathbf{x}_{i,r} \neq 0 \qquad (4)$$

where both η ($0 < \eta < 1$) and δ (>1) are positive constants called learning rates, and $\mathbf{w}_{s+1,t}$, $\mathbf{w}_{s,r}$ and $\mathbf{x}_{i,r}$ (usually $\mathbf{x}_{i,r} \geqslant 0$) are respectively the r-th ($1 \leqslant r \leqslant k$) component of vector $\mathbf{w}_{s+1}, \mathbf{w}_s$ and $\mathbf{x}_i$, where k is the number of retained features. From the above equations we can see that when a type A^+ error occurs (note that now y_i = 1), the weight vector will be increased by adding a positive vector(i.e., $\eta \mathbf{x}_i$ in equation 3) or each weight will multiply a number bigger than 1 (i.e., δ in equation 4), while a type A^- error corresponds to substracting or being divided by the same numbers as for a type A^+ error.

Winnow has some variant forms. The above basic form (equation 4) is called the positive Winnow since all weights are non negative. Another form is the balanced

Winnow, which uses the difference between two weight vectors $\mathbf{w}^+$ (called positive vector) and $\mathbf{w}^-$ (called negative vector) instead of the original $\mathbf{w}$ (i.e., $\mathbf{w} = \mathbf{w}^+ - \mathbf{w}^-$). The update function then becomes:

$$\begin{cases} \mathbf{w}^+_{s+1,r} = \mathbf{w}^+_{s,r} \times \beta \mathbf{x}_{i,r} \\ \mathbf{w}^-_{s+1,r} = \mathbf{w}^-_{s,r} \times \gamma \mathbf{x}_{i,r} \end{cases}, \text{ if } y_i = +1 \text{ and } \mathbf{w}_s \bullet \mathbf{x}_i < \theta \quad (5)$$

and

$$\begin{cases} \mathbf{w}^+_{s+1,r} = \mathbf{w}^+_{s,r} \times \gamma \mathbf{x}_{i,r} \\ \mathbf{w}^-_{s+1,r} = \mathbf{w}^-_{s,r} \times \beta \mathbf{x}_{i,r} \end{cases}, \text{ if } y_i = -1 \text{ and } \mathbf{w}_s \bullet \mathbf{x}_i \geq \theta \quad (6)$$

where β and γ are two learning rates of balanced Winnow, and β is a promotion parameter, γ a demotion parameter, $\beta > 1$, $0 < \gamma < 1$, $\mathbf{x}_{i,r} \neq 0$. It has been shown that the Balanced Winnow outperforms positive Winnow because the former allows positive and negative weights at the same time[4].

Both the Perceptron and Winnow have been shown to learn a linear separator if it exists [19], but they also appear to be fairly successful when there is no perfect separator [20][21]. The algorithms make no assumptions on the features. The advantages of the above online linear classifiers are that they are very simple and cheap to implement, can handle many irrelevant features [22], and can be updated very easily. The disadvantage is that they are theoretically only suitable for linearly separated two-class problems.

The Perceptron is not as efficient as Winnow due to its additive update process. But, it has been shown that when the examples are sparse, the Perceptron performs better, otherwise, Winnow is the better choice between the two classifiers. Another advantage for Winnow is that its error bound is smaller than that of the Perceptron [23]. Some text classification work has been done based on the Perceptron and Winnow [20][21][24][25][26][27]. But the results are not always encouraging. However, we believe that the anti-spam problem may be a good application for online classifiers, because of the large number of irrelevant

[4] See C. H. A. Koster, Document Classification. Url: http://www.cs.kun.nl/~kees/ir2/papers/h03.pdf

features in spam and legitimate emails, and the need for incremental adaptation in this task.

---

4 Corpora and preprocessing

In order to explore the effectiveness of the Perceptron and Winnow classifiers for spam filtering and compare these methods to those used in previous work, the investigation described in this paper makes use of two existing public benchmark collections and reports results using standard evaluation methods as described in this section.

4.1 Corpora

The use of benchmark corpora enables the effectiveness of methods to be compared fairly. For example, many TC experiments have been done using the Reuters corpora[5]. Two publicly available benchmark corpora for anti-spam filtering research, the PU1 and Ling-Spam collections, were used in our experiments reported in this paper. A number of existing studies on anti-spam filtering have been reported based on these two corpora [4] [7] [10] [11] [12] [13] [14]. Both corpora can be freely downloaded from the Internet[6].

The PU1 corpus consists of 481 spam messages and 618 legitimate messages, which are all real private personal emails. Because of privacy problems, these messages have been transformed to an "encrypted" version prior to distribution, in which the words are encoded with integers. Unfortunately, this transformation also limits the potential to explore language-based filtering techniques.

The Ling-Spam corpus was built up in a different way. The 481 spam messages are the same as the spam messages of PU1, but the 2412 legitimate messages were collected from a freely accessible newgroup, Linguist list. Thus Ling-Spam has no privacy problem and is not encrypted. Because the legitimate messages of Ling-Spam are more domain topic-specific than those of PU1, Ling-Spam is more appropriate for spam filtering of a topic-specific newsgroup.

For both PU1 and Ling-Spam, only the subject and body texts were retained, all the other information, such as attachments, HTML tags, and other header fields

---

[5] See http://about.reuters.com/researchandstandard/corpus/.
[6] http://iit.demokritos.gr/skel/i-config/downloads/.

were removed. Both PU1 and Ling-Spam have four versions available, these are denoted as: bared, lemmatized, stopword-eliminated, and lemmatized and stopword-eliminated. Thus each version is at a different level of lexical processing. We applied our classifiers using all these versions, with very similar results in each case. For this reason only the results for the bared form version are reported here. In the bared form version corpus, all the words, punctuation and other special symbols, e.g., "!" , "$", are treated as tokens, and were retained in our classifiers.

4.2 Feature selection

All the different tokens are regarded as candidate features. The original feature space is very large, for example the total number of the unique tokens of PU1 and Ling-Spam are respectively 24749 and 65694. To reduce the feature space and eliminate noise, as most classifiers do, we use feature selection. We investigated three different methods: IG (Information Gain), Odds ratio and DF (Document Frequency), which are respectively defined as follows:

$$IG(f) = \sum_{t\in\{0,1\}} \sum_{c\in\{c_1,c_2\}} P(f=t,c) \log \frac{P(f=t,c)}{P(f=t)P(c)} \qquad (7)$$

$$OddsRatio(f, c_1) = \frac{\dfrac{P(f \mid c_1)}{1-P(f \mid c_1)}}{\dfrac{P(f \mid c_2)}{1-P(f \mid c_2)}} \qquad (8)$$

$$\mathrm{DF}(f) = \{d \mid f \text{ occurs in } d\} \qquad (9)$$

where c_1 and c_2 refer to the spam class and legitimate class, d denotes a document, f is a candidate feature. $P(f|c)$ is the probability that the feature f is in class c, $f = 0$ or 1 respectively means f is absent or present.

IG and DF have been found to be the two top-performing feature selection methods for LLSF and kNN classifiers [28]. Our previous work on adaptive filtering based on Odds Ratio achieved good results [29]. In this investigation, we are interested in assessing the performance of these three methods with the online classifiers in the context of anti-spam filtering.

In our experiments each document d is represented with a binary vector $<x_1,x_2,..,x_k>$, x_i=+1 or 0 ($0 \le i \le k$), which respectively means the ith feature is present or absent in this document, k is the number of retained features after feature selection. Those features that actually occur in the document (x_i = 1) are called active features of this document.

4.3 Measures

Four traditional measures in text classification research are used for evaluation: *recall*, *precision*, *F1* measure and *accuracy*. For the following definitions let there be: a total of a messages, including a_1 spam and a_2 legitimate emails; b messages of which are judged as spam by an anti-spam filtering system, and among them, b_1 messages are really spam; and a total of a-b messages judged as legitimate, among which b_2 messages are really legitimate. The evaluation measures of this system are defined as follows:

$$recall = \frac{b_1}{a_1} \qquad (10)$$

$$precision = \frac{b_1}{b} \qquad (11)$$

$$F1 = \frac{2 \times recall \times precision}{recall + precision} \qquad (12)$$

$$accuracy = \frac{b_1 + b_2}{a} \qquad (13)$$

F1 combines recall and precision into a single measure and is frequently used in TC. Accuracy is widely used in many previous anti-spam filtering studies. However, we believe that it is greatly affected by the class distribution. For example, if the number of legitimate messages is much bigger than that of spam messages, the accuracy will not be greatly affected by spam messages. In other words, the accuracy may be very high even though the spam filtering performance is very poor due to the skew distribution. So the F1 measure is mainly used in our experiments, but accuracy results are included for comparison with existing work.

5 Experimental results

We performed four sets of experiments using the PU1 and Ling-Spam test corpora. In the first set different feature selection methods were compared. The relationship between the number of features and the performance was also investigated. The second set of experiments focused on the size of the training set. The number of training iterations was tested in the third set of experiments. In the last set of experiments, the Perceptron and Winnow were compared with Naïve Bayes. Some computational analysis of these methods is made in the comparison section.

The parameters of the classifiers were set to the same values for each test corpus for each classifier. All of the experiments were performed using 10-fold cross-validation, where each corpus was equally divided into 10 parts, 9 of them were used for training, while the other part was used for testing. Learning rates were set to α=0.5, β=1.8, γ=0.5. The number of iterations was set to 15, except in the experiments testing the number of iterations. For the Perceptron, the weight vector $\mathbf{w}$ was initialized to <1,1,…,1>, while for Winnow, the positive weight vector $\mathbf{w}^+$ and the negative weight vector $\mathbf{w}^-$ were initialized to <2,2,…,2> and <1,1,…1> respectively. For both classifiers, the threshold θ was initialized to the average number of active features in a training document, so for each document vector **d**, the initial value of $\mathbf{w} \bullet \mathbf{d}$ in the Perceptron classifier or $(\mathbf{w}^+ - \mathbf{w}^-) \bullet \mathbf{d}$ in Winnow classifier was comparable to θ.

5.1 Experiments with different feature selection approaches and feature set size

Our first experiments used different feature selection approaches for both the Perceptron and Winnow. Three feature selection approaches were applied: IG, DF and Odds ratio. The top-*k* words with the highest IG, DF or Odds ratio were selected as the final features. We then changed the number of features *k*. The range variations were divided into two parts. In the first part, *k* varied from 10 to 200 by 10, and then from 200 to 800 by 50; in the second part, it varied from 500 to 15000 by 500. The first part was also used for comparison with Naïve Bayes (in our experiments, Naïve Bayes doesn't perform well when the number of features is very large, see Figure 7). Due to the space limitation in a single figure, results

for the first part are shown in Figure 7, while results for the second part are shown in Figure 2.

From figure 2 we can see that for both the Perceptron and Winnow, IG and DF perform similarly well, and clearly much better than Odds ratio. This finding is similar to Yang's observation that IG and DF are two top-performing feature selection methods when using kNN and LLSF classifiers [28]. A reasonable explanation for this observation that IG and DF outperform Odds ratio, is that Odds ratio biases toward favoring those features which are good at representing spam class, whereas features good for representing the legitimate class are also very useful in the Perceptron and Winnow classifiers because they allow negative weights.

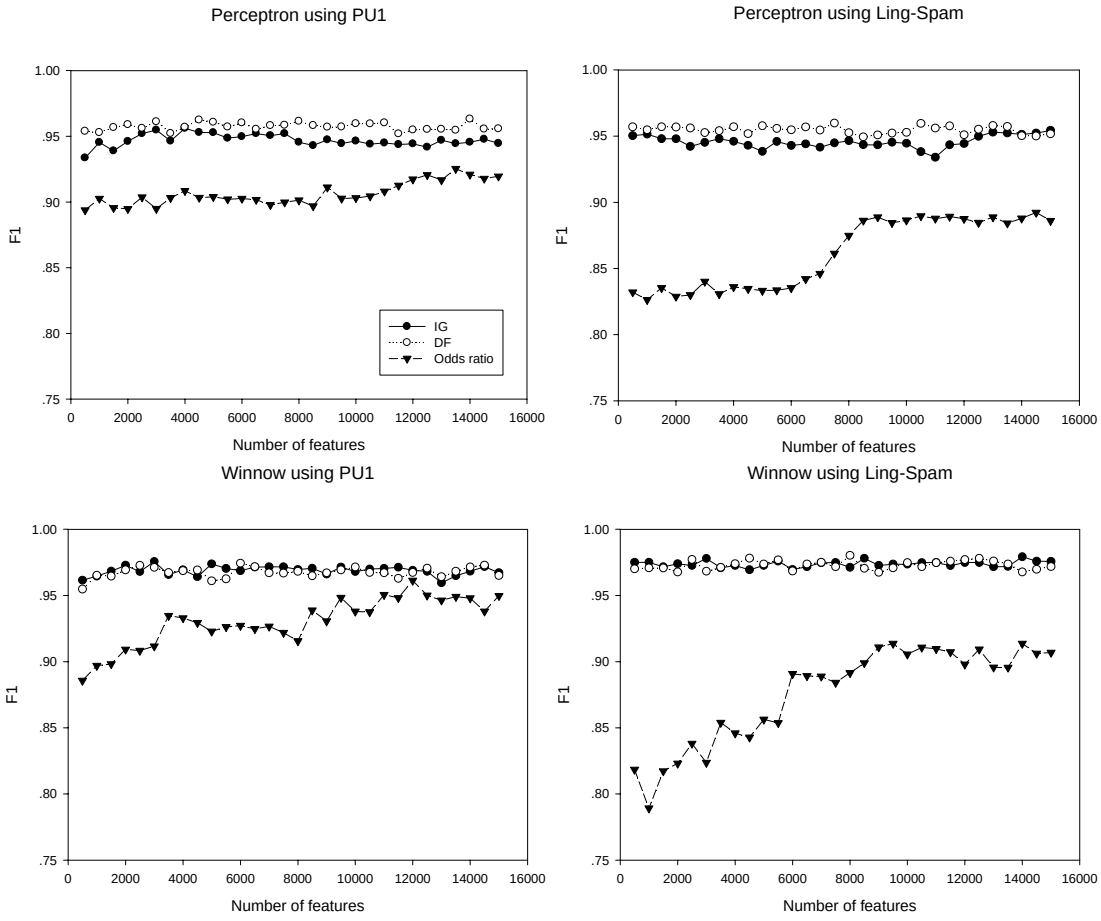

Fig. 2 F1 results with varying the number of features.

It is also shown that both the Perceptron and Winnow perform very stably when the number of features varies from 500, especially using IG and DF. When the number of features is smaller, the performance improves rapidly to a steady level while the number of features increases (see Figure 7). This means that the two

classifiers can perform very well under a relatively small feature space, thus the computational cost can be reduced.

Another interesting observation relates to the results using Odds ratio (see those lines with triangles in Figure 2). It is obvious that the results using Ling-Spam are much worse than those using PU1. We believe the reason for this is that the Odds ratio biases toward favoring the features that are effective at representing the spam class, but for Ling-Spam the features selected from spam emails are not good for classifying legitimate emails and spam. Therefore, for the Ling-Spam corpus, when the number of features increases, the performance first keeps low and then increases because some other useful features are involved. But when the number of features increases to a certain level, some unuseful features are introduced and the performance doesn't improve further.

In order to check the generality of the feature selection results, two more corpora—PU3 (2313 legitimate emails and 1826 spam) and PUa (571 legitimate emails and 571 spam)were used. These can be downloaded from the same website as PU1, but are very different from PU1 and Ling-Spam. In fact, each of PU1, PU3 and PUa was derived from a different user [12]. Results using PU3 and PUa are shown in Figure 3. These results are similar to those results using PU1 and Ling-Spam (see Figure 2). Thus it can be concluded that our observation for feature selection is consistent, at least for these four very different corpora.

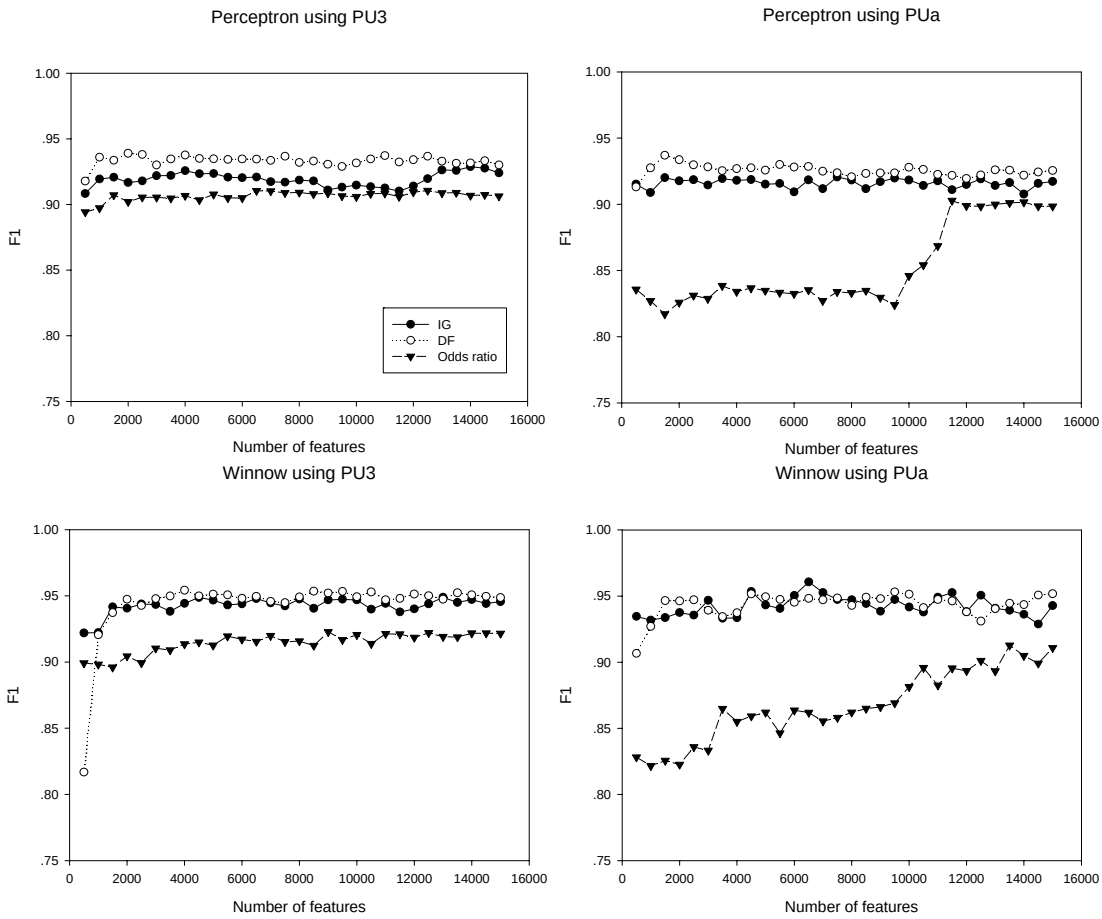

Fig.3 Experimental results using PU3 and PUa

5.2 Experiments with different size of training corpus

In the second set of experiments we investigated the performance of the Perceptron and Winnow when varying the size of the training corpus. We used *t*% of the training examples, where *t* varied from 10 to 100 by 10. The top 2000 features[7] with highest IG scores were selected, and the learning rates are α=0.5, β=1.8, γ=0.5. The number of training iterations was set to 15.

[7] In fact, we found that our results are insensitive to different values of parameters, including the number of features, learning rates, and the number of training iteration. So, here we just choose the values for instantiation without special purpose.

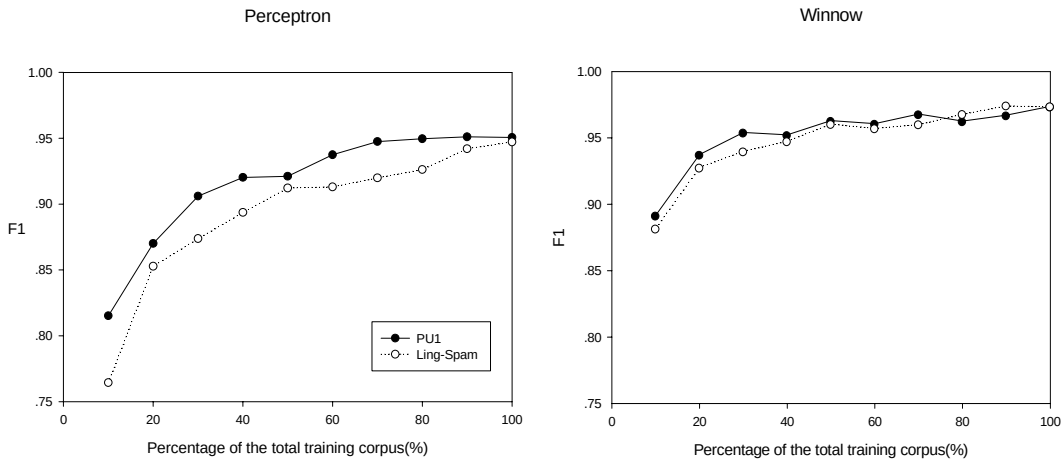

Fig. 4 F1 results with different size (represented by the percentage of the total training corpus) of the training corpus. The 100% size of PU1 training corpus is 989, while the value is 2603 for Ling-Spam.

Figure 4 shows that when the size of the training corpus increases, the performance of both the Perceptron and Winnow also increases. But that the F1 performance only increases slowly compared to the increase in the size of the training set. For Winnow, the increase is less than that of the Perceptron. The figure shows that it is unnecessary to use a very large training corpus to get a relatively high performance (above 90%) when using these two algorithms in anti-spam filtering. However, if higher performance is wanted, more training documents are needed.

5.3 Experiments with different number of training iterations

In these experiments, we investigate how the number of training iterations affects the performance. Figure 5 shows the results of our experiments, the number of iterations varied from 1 to 20 in increments of 1.

From Figure 5 we can see that the performance improves when the number of iterations increases at first, and after that it remains stable again. It can be concluded that to train the Perceptron or Winnow, it is not necessary to train the examples many times. In both cases the stable performance level is reached after around 8 iterations.

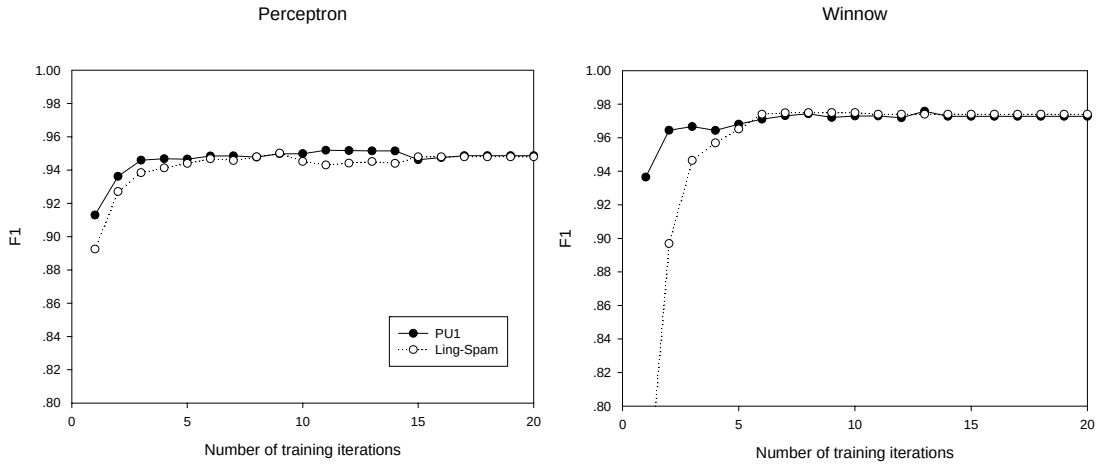

Fig. 5 F1 results with different numbers of training iterations for the Perceptron and Winnow, using PU1 and Ling-Spam.

5.4 Comparison experiments

In these experiments, we compare the performance between the Perceptron and Winnow, and compare them with Naïve Bayes classifier. Some computational and impelmentation analysis is also give for comparison.

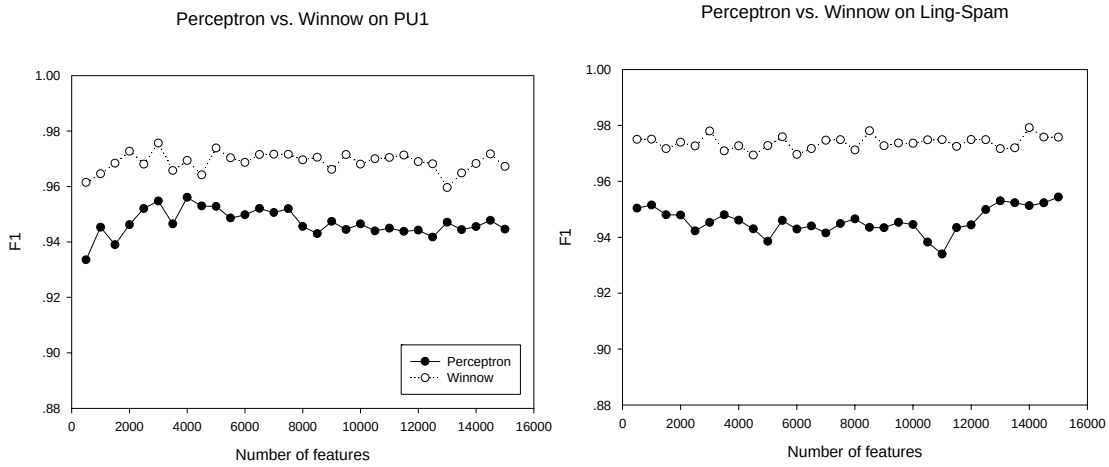

Fig. 6 Comparative results of the Perceptron and Winnow, using PU1 and Ling-Spam

Figure 6 shows the comparison between the Perceptron and Winnow on both PU1 and Ling-Spam, using IG. Learning rates are α =0.5, β =1.8, γ =0.5. The number of iterations is set to 15. The figure is recompiled from the previous results in figure 2. The Winnow performs slightly better than the Perceptron.

We then compared the Perceptron and Winnow with Naïve Bayes, which is believed to be one of the fastest and most effective classifiers for anti-spam filtering [4][7][9][11][12][13][14]. Keyword-based methods are usually used in anti-spam software and can be regarded as good baseline classifiers for our

research. However, Androutsopoulos *et al.* [10] have already demonstrated Naïve Bayes to be much better than a widely used keyword-based method, so we directly choose Naive Bayes as our baseline classifer. In our experiments, we implemented both the Multi-variate Bernoulli Model (MBM) and the Multinomial Model (MM), see Section 2, forms of Naïve Bayes, and found that the MBM model slightly outperformed the MM model, especially using the PU1 corpus. Therefore, we used the results of the MBM model for our Naïve Bayes classifier. Figure 7 shows the comparative results. The performance of all three classifiers is shown with varying number of features first from 10 to 200 by 10, and then from 200 to 800 by 50.

From Figure 7, we can see that when the number of features is small, Naïve Bayes performs comparably to the Perceptron and Winnow filters, but that when the number of features increases, the performance of Naïve Bayes falls, while the performance of the Perceptron and Winnow increase and then, as shown previously (see Figure 2), become very stable as the number of features increases. A reasonable explanation for this behaviour is that anti-spam filtering is a task with many irrelevant features, but Naïve Bayes can't perform very well if many irrelevant features are considered to estimate the probabilities. However, from Figure 2, we can see clearly that the Perceptron and Winnow can handle such conditions very well. These experiments were repeated using PUa and PU3 corpora, and although these two corpora are rather different from PU1 and Ling-Spam, we got very similiar results. Thus, we believe our approaches can be generalized for other spam corpora.

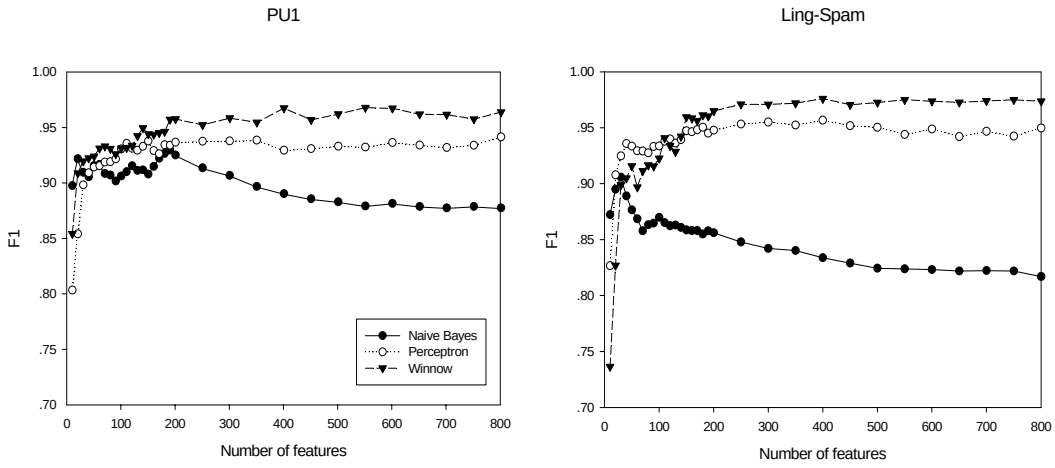

Fig. 7 Comparative results of Naïve Bayes, the Perceptron and Winnow classifiers

Table 1 lists the best F1 results that we found using the above three classifiers. It should be pointed out that with different parameters the results may change a

little. But as shown before, the performance of the Perceptron and Winnow filters are quite stable. Some other previous results using PU1 and Ling-Spam can be found in [4][7][10][11][12][13][14]. From Table 1, we can see that for the PU1 corpus, the best F1 result so far was 97.90% using Adaboost classifiers [14], while our results for the Perceptron and Winnow are respectively 97.40% and 97.57%. For Ling-Spam, the best Accuracy result so far is 99.2% still using the Adaboost classifier[8], while our results of the two classifiers are 98.89% and 99.31%. Compared to previous studies, our results are very competitive. Further analysis later in this section shows that our classifiers are much cheaper than Adaboost, and that they can easily be updated, thus our approaches are more suitable for spam filtering.

Table 1 The best F1 results of NB, the Perceptron, Winnow and some top previous results

Corpus	Classifier	Recall	Precision	F1	Accuracy
PU1	NB	98.55	87.53	92.69	93.18
	Perceptron	97.29	97.55	97.40	97.72
	Winnow	97.09	98.17	97.57	97.91
	Adaboost	97.09	98.73	97.90	N/A
	SVM	95.60	97.08	N/A	96.70
Lingspam	NB	90.65	97.94	94.06	98.13
	Perceptron	94.60	98.75	96.58	98.89
	Winnow	97.72	98.14	97.92	99.31
	AdaBoost	96.7	98.6	N/A	99.2

Table 2 lists the computational complexity of the training (excluding feature selection) and classification of the three classifiers.

Table 2 Computational complexity of three classifiers in anti-spam filtering. N is the number of training messages. k the number of features, n the number of iterations in the Perceptron and Winnow training, t the number of iterations in the Boosting methods.

Learning method	Training	Classification
NB	$O(kN)$	$O(k)$
Perceptron	$O(nkN)$	$O(k)$
Winnow	$O(nkN)$	$O(k)$

[8] See M. DeSouza, J. Fitzgerald, C. Kemp and G. Truong, "A Decision Tree based Spam Filtering Agent", http://www.cs.mu.oz.au/481/2001_projects/gntr/index.html, 2001

Adaboost with decision tree	$O(tkN^2)$	$O(t)$

Naïve Bayes is believed to be one of the fastest classifiers (either for training or test phases) among those used in anti-spam filtering [12]. Its computational complexity is $O(kN)$ for training and $O(k)$ for test. Both the Perceptron and Winnow also have a $O(k)$ time complexity for test. So using either of these classifiers to filter spam messages is very fast.

Although Adaboost classifiers using decision trees as weak learners can get very good results, their training time is much longer than other classifiers. So it is not very suitable when handling large number of Email messages.

For both the Perceptron and Winnow, when training, the worst situation is that in each iteration for each training example, the weight vector is updated. The complexity of each update is $O(k)$. Thus the total update complexity is $O(nkN)$. The sorting complexity for searching the minimum error number is $O(n^2)$. Because usually $n<<k$, the training complexity of the Percepron and Winnow is $O(nkN)$. In fact, the two methods are both error-driven and the number of active features are small for each message, so the actual number of update features at each iteration is much smaller than k (e.g., when the top 5000 IG features are selected for Ling-Spam or PU1, the average number of active features in training documents are 134 or 155 respectively). Meanwhile, as shown previously, n, k and N can be reduced to small values while still achieving good results. In other words, the practical filters based on these two classifiers are very fast for both the training and test phases.

Obviously, the space costs are also very cheap for these two classifiers, since they only store the weight vectors. Another advantage is that they are very easily updated since they are online classifiers. When users change their definition for spam emails, the Perceptron and Winnow can easily be updated through user feedback.

6 Conclusion and future work

In this paper, two online linear classifiers, the Perceptron and Winnow, have been investigated in the context of anti-spam filtering under two benchmark corpora,

PU1 and Ling-Spam. We studied their performance when varying the number of features along with three different feature selection methods: Information Gain (IG), Document Frequency (DF) and Odds ratio. The size of training set, and the number of training iterations were also investigated for both classifiers. The experimental results show that both the Perceptron and Winnow perform much better when using IG or DF than using Odds ratio, and that they are insensitive to the number of features and the number of training iterations and not greatly sensitive to the size of training set. It was also shown that Winnow slightly outperforms the Perceptron, and that both of them perform much better than Naïve Bayes. We also found the same conclusions when using two additional test corpora PUa and PU3, which are very different from PU1 and Ling-Spam.The theoretical and implementation computational complexity of these two classifiers are very low, and they can very easily be adaptively updated. The analysis and promising experimental results indicate that the Perceptron and Winnow are two very competitive classifiers for anti-spam filtering.

As we pointed out in the Introduction section, users may pay more attention to the precision than the recall. Our future work will include considering the above classifiers under a cost-sensitive framework. Meanwhile, we believe that many other features, including non-textual features, such as hyperlinks, can improve the filtering performance, and we will take them into account. Other work is currently focused on developing a benchmark corpus for Chinese spam email filtering. We will collect and build up a text corpus, and more information, including structural information such as hyperlinks, will be retained for applying more approaches, e.g., some Web information retrieval techniques. An interesting question, which we will be exploring, is the extent to which our results for English corpora prove that these techniques will be effective for all languages?

Another problem is that the volume of data here is very small. Can we trust these results over a larger volume of messages? And when spam changes or evolves, can the system cope with the evolving spam? We hope to address these questions using the spam Email research corpus developed by NIST for the Spam Detection task at the annual TREC conference in 2005 (TREC2005) in the near future.

7 Originality and Contribution

To the best of our knowledge, this is the first application of two linear classifiers the Perceptron and Winnow, which are moderate classifiers in traditional text categorization, to the task of spam filtering. Feature selection approaches are compared based on public benchmark corpora. Filtering performance, computation complexity, application issues are compared with previous filtering approaches. Experimental results show that these two classifiers outperform existing methods for spam filtering and they are very fast and suittable for real use.

Acknowledgements This paper is supported by a grant of China 973 project No. 2004CB318109 and a Beijing Science and Technology Planning Program No. D0106008040291. We would like to thank the reviewers for providing valuable comments and advices.

References

[1] Vaughan Nichols, S.J. (2003) Saving Private E-mail. IEEE Spectrum 40(8): 40-44

[2] Whitworth, B., Whitworth, E. (2004) Spam and the Social-Technical Gap. IEEE Computer 37(10): 37-45

[3] Hoffman, P., Crocker, D. (1998) Unsolicited bulk email: Mechanisms for control. Technical Report Report UBE-SOL, IMCR-008, Internet Mail Consortium

[4] Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C.D.,Stamatopoulos, P. (2000) Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach. In Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp 1-13

[5] Cohen W. (1995) Fast effective rule induction. In Machine Learning: Proceedings of the Twelfth International Conference, pp 115-123

[6] Drucker, H., Wu, D., Vapnik V.N. (1999) Support Vector Machines for Spam Categorization. IEEE Transactions on Neural Networks 20(5): 1048-1054

[7] Hidalgo, J.M.G. (2002) Evaluating Cost-Sensitive Unsolicited Bulk Email Categorization. In Proceedings of ACM Symposium on Applied Computing, pp 615-620

[8] Yang, L., Xiaoping, D., Ping, L., Zhihui, H., Chen, G., Huanlin, L. (2002) Intelligently analyzing and filtering spam emails based on Rough Set. In Proceedings of 12th Chinese Computer Society Conference on Network and Data Communication, pp ????

[9] Sahami, M., Dumais, S., Heckerman, D., Horvitz, E. (1998) A Bayesian approach to filtering junk e-mail. In Proceedings of AAAI Workshop on Learning for Text Categorization, pp 55-62

[10] Androutsopoulos, I., Koutsias, J.,Chandrinos, K.V.,Spyropoulos, C.D. (2000) An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Encryp ted Personal E-mail Messages. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 160-167

[11] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V.,Paliouras, G., Spyropoulos, C.D. (2000) An Evaluation of Naive Bayesian Anti-Spam Filtering. In Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, pp 9-17
[12] Androutsopoulos, I., Paliouras, G., Michelakis, E. (2004) Learning to Filter Unsolicited Commercial E-Mail. Technical report 2004/2, NCSR "Demokritos"
[13] Schneider, K. (2003) A Comparison of Event Models for Naive Bayes Anti-Spam E-Mail Filtering. In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics, pp 307-314
[14] Carreras, X., Marquez, L. (2001) Boosting Trees for Anti-Spam Email Filtering. In Proceedings of European Conference Recent Advances in NLP, pp 58-64
[15] Lewis, D.D.,Schapire, R.E.,Callan, J.P.,Papka, R. (1996) Training algorithms for linear text classifiers. In Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval, pp 298-306
[16] Rocchio, J. (1971) Relevance feedback in information retrieval. In the SMART Retrival System: Experments in Automatic Document Processing, pp 313-323, Prentice Hall Inc.
[17] Rosenblatt, E. (1988) The Perceptron: a probabilistic model for information storage and organization in the brain. Psych. Rev. 65 (1958) 386-407; reprinted in: Neurocomputing (MIT Press, Cambridge, MA. 1988)
[18] Littlestone, N. (1988) Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning 2(4): 285-318
[19]Grove, A.J.,Littlestone, N., Schuurmans, D. (1997) General Convergence Results for Linear Discriminant Updates, In Annual Workshop on Computational Learning Theory, Proceedings of the tenth annual conference on Computational learning theory, pp 171–183
[20] Dagan, I., Karov, Y., Roth, D (1997) Mistake-driven learning in text categorization. In Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing, pp 55–63
[21] Ng, H. T., Goh, W. B., Low, K. L (1997) Feature selection, perceptron learning, and a usability case study for text categorization. In Proceedings of the 20th ACM International Conference on Research and Development in Information Retrieval, pp 67–73
[22] Zhang, T. (2001) Regularized Winnow methods. In Advances in Neural Information Processing Systems 13, pp 703-709
[23] Kivinen, J Warmuth, M.K., Auer, P. (1997) The Perceptron algorithm versus Winnow: linear versus logarithmic mistake bounds when few input variables are relevant. Artificial Intelligence, .97(1-2): 325-343
[24] Bel, N., Koster, C.H.A., Villegas, M. (2003) Cross-Lingual Text Categorization. In Proceedings the 7th European Conference on Digital Library, Springer LNCS 2769, pp 126-139
[25] Liere, R., and Tadepalli, P. (1998) Active Learning with Committees in Text Categorization: Preliminary Results in Comparing Winnow and Perceptron. In Learning for Text Categorization, Technical Report WS-98-05, AAAI Press
[26] Schütze, H., Hull, D. A., Pedersen, J. O (1995) A comparison of classifiers and document representations for the routing problem. In Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval, pp 229–237
[27] Wiener, E. D., Pedersen, J. O., Weigend, A. S (1995) A neural network approach to topic spotting. In Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval, pp 317–332
[28] Yang, Y., Pedersen J.P (1997) A Comparative Study on Feature Selection in Text Categorization. In Proceedings of the Fourteenth International Conference on Machine Learning, pp 412-420
[29] Xu, H., Yang, Z., Wang, B., Liu, B., Cheng, J., Liu, Y., Yang, Z., Cheng, X.,Bai, S (2002) TREC 11 Experiments at CAS-ICT: Filtering and Web. In the Proceedings of the Eleventh Text Retrieval Conference, pp 105-115

About the authors

Bin Wang is currently an associate professor in the Institute of Computing Technology, Chinese Academy of Sciences, China. He received a computer science degree in 1993 and then accomplished his master thesis in 1996, from Wuhan University. He obtained his PhD in 1999 in the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include information retrieval and natural language processing. He became an IEEE member in 2000. He is also a member of China Computer Society and Chinese Information Processing Society.

Gareth J. F. Jones received the B.Eng and PhD degrees from the University of Bristol. He subsequently conducted research at the University of Cambridge and as a Toshiba Fellow in Japan. He was then a faculty member with the Department of Computer Science, University of Exeter. He is currently with the School of Computing, Dublin City University where he is a Principal Investigator in the Centre for Digital Video Processing. His research interests are in applications of information retrieval including multimedia and multilingual content. He has published more than 100 papers including Best Paper Awards at ACM SIGIR and ACM Multimedia.

Wenfeng Pan is currently a Manager Assistant in the Technology Department of Dalian Commodity Exchange. He obtained his Master degree in 2004 in the Institute of Computing Technology, Chinese Academy of Sciences. His research interests includes information retrieval and data mining.