SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2009; **39**:1003–1024
Published online 22 April 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe.925

Better Naive Bayes classification for high-precision spam detection

Yang Song[1,*,†,‡], Aleksander Kołcz[2] and C. Lee Giles[3]

[1]*Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, U.S.A.*
[2]*Microsoft Live Labs, One Microsoft Way, Redmond, WA 98052, U.S.A.*
[3]*College of Information Science and Technology, The Pennsylvania State University, University Park, PA 16802, U.S.A.*

SUMMARY

Email spam has become a major problem for Internet users and providers. One major obstacle to its eradication is that the potential solutions need to ensure a very low false-positive rate, which tends to be difficult in practice. We address the problem of low-FPR classification in the context of naive Bayes, which represents one of the most popular machine learning models applied in the spam filtering domain. Drawing from the recent extensions, we propose a new term weight aggregation function, which leads to markedly better results than the standard alternatives. We identify short instances as ones with disproportionally poor performance and counter this behavior with a collaborative filtering-based feature augmentation. Finally, we propose a tree-based classifier cascade for which decision thresholds of the leaf nodes are jointly optimized for the best overall performance. These improvements, both individually and in aggregate, lead to substantially better detection rate of precision when compared with some of the best variants of naive Bayes proposed to date. Copyright © 2009 John Wiley & Sons, Ltd.

Received 28 March 2008; Accepted 24 February 2009

KEY WORDS: spam filtering; naive Bayes; cascaded models

1. INTRODUCTION

Despite numerous important developments in machine learning algorithms, the naive Bayes (NB) classifier [1,2], based on the simple assumption of class-conditional variable independence, has retained its popularity due to its computational tractability and competitive performance and easy

*Correspondence to: Yang Song, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, U.S.A.
†E-mail: yasong@cse.psu.edu
‡This work was done when the first author was an intern at Microsoft Live Labs Research.

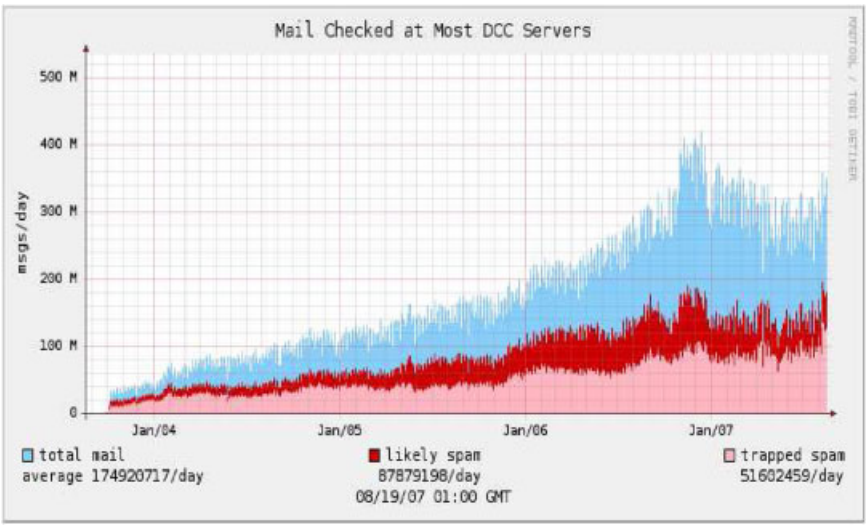

Figure 1. Statistics show that the spread of spam has become one of the most crucial problems on the web (http://www.dcc-servers.net/).

implementation. Importantly, the learning process of NB is extremely fast compared with current discriminative learners, which makes it practical for large real-world applications. Since the training time complexity of NB is linear to the number of training data, and the space complexity is also linear in the number of features, it makes NB both time and storage efficient for practical systems. In Web-related problems, NB has proved to be particularly effective for the task of *text classification* (TC), which is the process of assigning a document into one or more predefined categories. However, NB has an overconfidence deficiency for TC applications demanding low false positive (FP) rates, stemming from the violations of the independence assumption [3]. Given that low-FP constraints are quite common, improvements to NB in this regard are of significant practical importance. One such prominent application is email spam filtering, which has recently received much attention in the machine learning community [4–8].

Huge amounts of spam are being generated every day and waste significant Internet resources, as well as users' time. A conservative estimate states that *12.4 billion* spam emails were sent daily in the year 2006[§], compared to *31 billion* total emails sent in 2003. Figure 1 shows an example statistic covering the last 3 years. As can be seen, although spam filters generally perform well (see *trapped spam*), the total number of spam emails sent (see *likely spam*) is still increasing. In general, spam filtering is a complex problem that can be addressed in many different ways, ranging from the use of IP block/white lists to statistical content filters. In practice, large systems usually employ multiple techniques and aggregate them together for better results. A key component of most such systems provides spam detection based on email content, which can be seen as a specialized TC task, and also often uses NB as the underlying classifier. Improving the performance of NB in this important area is the focus of this paper.

[§] http://spam-filter-review.toptenreviews.com/spam-statistics.html.

DOI: 10.1002/spe

Table I. Statistics of data sets in our experiments. The details of the experiments are presented in Section 6.

Data set	Training set	Test set	Features
Trec-2005	30 727	61 455	208 844
Trec-2006	12 607	25 214	133 495
Hotmail	765 000	149 746	2 644 921

A major characteristic that distinguishes spam filtering from many other TC tasks is its highly asymmetric misclassification cost. Because misclassifying a legitimate email as spam (i.e. *FP event*) is usually much worse than missing a spam email (i.e. a *false negative* (*FN*) *event*), the misclassification cost should carefully be taken into account in the decision making process [3]. Although exact misclassification costs are sometimes hard to quantify, an acceptable spam filter is generally one that incurs very low false positive rates (FPRs) (e.g. ≤ 0.005). Recent research has begun addressing the problem of improving the performance of classifiers for low-FP problems [3,9]. In this paper we maintain this focus and propose several improvements that make NB a much more effective classifier in this context. Our contributions are as follows:

- We propose a statistical correlation measure as a new function for document term weighting, which we show to be very effective both on its own, and as a mixing function combining different types of term weights.
- Empirically, short instances (e.g. documents or emails) are more likely to be misclassified. We suggest class-dependent collaborative filtering (CF) for augmenting the features for short instances. The CF algorithm is very efficient as it scales linearly with the number of training instances.
- To improve the performance of the NB classifier at a specific FPR, we propose a cascade combining three NB classifiers. Here, the base classifier separates the training instances into two parts, and we apply linear programming to jointly optimize the decision thresholds of the second stage classifiers.
- Finally, we combine the aforementioned techniques together. Experimental results on three data sets, whose basic statistics are given in Table I, indicate that our model substantially outperforms the previously proposed NB variants and is able to outperform state-of-the art methods such as a linear Support Vector Machine (SVM). The improvements come without compromising the inherent scalability of the NB classifier.

The rest of the paper is organized as follows: Section 2 presents the related work on NB; Section 3 discusses our correlation-based method as well as its combination with other term weighting functions; Section 4 presents the class-dependent CF technique for augmenting short instances; Section 5 proposes the cascaded models as well as the entire framework for binary classification; Section 6 shows the experimental results; Section 7 compares our model with the most related cascaded models; finally we conclude in Section 8.

2. NAIVE BAYES FOR LOW-FP TC

From the machine learning perspective, spam filtering can be treated as a *binary* classification problem, where legitimate (good) emails are considered as *negative* (–) instances, and spam as

DOI: 10.1002/spe

positive (+) instances. Here, given a labeled training set $\{x^{(d)}, y^{(d)}\}_1^n$, where $x^{(d)} \in \mathbb{R}^m$ with m denoting the size of the vocabulary, n the size of the training data set, and $y^{(d)} \in \{-1, +1\}$, one seeks to find a function f that minimizes the expected loss of the mapping $x \mapsto \{-1, +1\}$. Under the bag of words model each email d is represented by a feature vector, where each attribute corresponds to the number of occurrences of the corresponding term in the email, i.e. $x^{(d)} = \{x_1^{(d)}, \ldots, x_m^{(d)}\}$, where $x_j^{(d)}$ is the frequency of feature x_j in email d.

Based on the assumption that document features are independent of each other given the class, NB estimates the posterior class membership of a document in terms of the class-conditional probabilities of individual features. A decision rule, such as *maximum a posteriori*, is then used to assign the document to a specific class. Although the theoretical assumption of feature independence is often violated in practice, numerous experiments and practical experience have shown that NB performs surprisingly well regardless [10].

Several flavors of NB have been proposed in the literature, including variants based on the multinomial, Poisson, and Bernoulli event models [2]. Among these, the multinomial NB [11] tends to be particularly favored in TC. Specifically, the multinomial model assumes that the bag of words representing a document is generated by sampling from a larger vocabulary according to a multinomial distribution. For a problem involving C classes, the model is captured by a parameter vector $\theta = \{\theta_1, \ldots, \theta_C\}$, where for class c, $\theta_c = \{\theta_{c1}, \ldots, \theta_{cm}\}$. Each θ_{ci} corresponds to the probability that term x_i appears in class c, i.e. $\theta_{ci} = P(x_i|c)$. Based on the feature independence assumption, the probability of a document $x^{(d)}$ for a given class c is given by:

$$P(x^{(d)}|\theta_c) = \frac{(\sum_i x_i^{(d)})!}{\prod_i x_i^{(d)}!} \prod_i (\theta_{ci})^{x_i^{(d)}} \tag{1}$$

One way to estimate the value of θ_{ci} is via a smoothed maximum likelihood estimate:

$$\theta_{ci} = \frac{\lambda + \sum_{x^{(i)} \in c} x_k^{(i)}}{\lambda m + \sum_k \sum_{x^{(i)} \in c} x_k^{(i)}} \tag{2}$$

which is called Lidstone smoothing [12]. When $\lambda = 1$, this is also known as Laplace smoothing [13].

While NB can naturally be applied to multi-class problems [14], the decision rule becomes particularly simple for binary classification tasks. Then the choice of whether to assign a document to the positive (+) or the negative (–) class depends on the ratio of the posterior probabilities of the two classes given the input document, i.e.

$$\frac{P(+|x^{(d)})}{P(-|x^{(d)})} = \frac{P(+)}{1 - P(+)} \frac{P(x^{(d)}|+)}{P(x^{(d)}|-)} \overset{\text{eq.1}}{\propto} \prod_i \left(\frac{\theta_{+i}}{\theta_{-i}}\right)^{x_i^{(d)}} \tag{3}$$

where θ_{+i} and θ_{-i} denote the probability of term x_i belonging to the positive and the negative class, respectively. Many improvements have been proposed to counter the simple assumptions of NB. While some of them make the induction of NB significantly more complex, and thus less attractive, others lead to substantial improvements at a much smaller cost. The latter category includes: document-specific feature selection, application of term weighting functions and document length normalization [15,16]. In particular, Rennie *et al.* [15] proposed to use $L2$ document length

DOI: 10.1002/spe

normalization coupled with traditional TFIDF term weighting, which is a document representation commonly used in Information Retrieval. More recently, Kołcz and Yih [17] suggested that for NB $L1$ document length normalization would be more appropriate. They also showed that further improvements in the performance can be gained by considering more complex term weighting functions, e.g. those combining supervised and unsupervised approaches. Their results provide a motivation for this work, particularly in the direction of introducing more powerful term weighting techniques while adopting $L1$ document length normalization. Accounting for the presence of document length normalization and term weighting changes the traditional bag-of-words document representation to one, where a document $x^{(d)}$ containing u unique terms is represented by a real-valued feature vector

$$x^{(d)} = \{t_1^{(d)}, \ldots, t_u^{(d)}\} \tag{4}$$

with $t_j^{(d)} = tw(x_j^{(d)}) / \sum_k tw(x_k^{(d)})$ and where $tw(\alpha)$ corresponds to term weight associated with term α.

3. ABSOLUTE CORRELATION RATIO (ACR) FOR SUPERVISED TERM WEIGHTING

In TC and information retrieval, a variety of term weighting methods have been proposed with the goal of improving the performance beyond that offered by using binary features or in-document term frequencies. Generally, they can be classified into two categories:

- *Unsupervised methods*: In this case no class labels for training instances are available and the statistics of a document corpus as well as each individual document are used to derive the term weights. These methods include Term Frequency (TF), Inverse Document Frequency (IDF) as well as their combination, i.e. TFIDF[¶] [18], which is usually found to perform better.
- *Supervised methods*: These methods use the functions normally used for attribute ranking in feature selection to provide term weights. Popular choices include Information Gain (IG) and χ^2 [19]. More recently, it has been suggested to use the outcomes of linear classifiers as scoring functions, with good results reported for, among others, SVM, odds-ratio (OR) derived from NB, and absolute log-odds (ALO), also by NB.

These methods can naturally be used to rank features as well as to provide numerical values for weighting features. It has also been observed that the combination of both supervised and unsupervised term weighting methods has the potential of outperforming either method [17]. For example, combining the unsupervised IDF approach with the supervised ALO scheme (i.e. $tw_k = IDF(x_k) \cdot ALO(x_k)$) showed better performance than either of them [17].

In this section, we propose a correlation-based term weighting method. Statistically, correlation indicates how strongly pairs of variables are related linearly. In the context of feature selection in TC, it can be used to measure the predictive power of a specific term with respect to the class variable. Indeed, the correlation function has been found quite effective as a feature ranking function

[¶] $TF_i^{(d)} = x_i^{(d)} / \sum_k x_k^{(d)}$, $IDF_i = \log|D| / \sum_d 1(x_i^{(d)} \neq 0)$, where $|D|$ is the total number of documents, and the denominator corresponds to the number of documents that contain term x_i.

DOI: 10.1002/spe

for high-dimensional data [20]. Specifically, the absolute correlation ratio (ACR) between term x_k and the class variable $y \in \{-1, +1\}$ is defined as

$$ACR(y, x_k) = \alpha(x_k) \cdot |\rho(y, x_k)| \tag{5}$$

where

$$\rho(y, x_k) = \frac{\sum_{d=1}^{n} (y^{(d)} - \bar{y})(x_k^{(d)} - \bar{x}_k)}{\sqrt{\sum_{d=1}^{n} (y^{(d)} - \bar{y})^2} \sqrt{\sum_{d=1}^{n} (x_k^{(d)} - \bar{x}_k)^2}} \tag{6}$$

and $\bar{y}$ and $\bar{x}_k$ correspond to the mean values of labels and term x_k, respectively, i.e.

$$\bar{y} = \frac{1}{n} \sum_{d=1}^{n} y^{(d)}, \quad \bar{x}_k = \frac{1}{n} \sum_{d=1}^{n} x_k^{(d)} \tag{7}$$

Here, α is a term-dependent smoothing parameter. Adding the parameter α to the correlation method is critical, since it is easy to verify that if a term only appears in a single document, its correlation with the class variable will always be 1. Recall that the absolute value of correlation is always between 0 and 1, where larger values indicate stronger correlation. Intuitively, a term that appears only once is not expected to have a strong predictive power and needs therefore to be down-weighted. This can be accomplished naturally by mixing the Document Frequency (DF) with the correlation function. Thus, the ACR used in this paper is defined as:

$$ACR(y, x_k) = \log(DF(x_k)) \cdot |\rho(y, x_k)| \tag{8}$$

It is easy to see that the proposed function is a combination of supervised and unsupervised weighting functions, since the correlation takes the class labels into consideration, while DF does not. Additionally, the correlation function itself can also be seen as a term weighting aggregator. Notice that if the feature values $x_k^{(i)}$ correspond to the TFIDF weights of these features in the training documents, the correlation measure transforms these unsupervised weights into supervised ones by taking the class label into account.

4. CLASS-DEPENDENT CF

In information retrieval, *query expansion* is sometimes applied to short queries to increase the quality of search results, since short queries are usually ambiguous and may increase the query/document mismatch rate. Similarly, in TC an instance with few features is often harder to classify correctly. This is encountered frequently in spam filtering where, for example, spam is often sent in the form of pictures with minimal accompanying text. Figure 2 shows the receiver operating characteristic (ROC) curves produced by NB for an example email data set (corresponding to Trec-2006, the details for which are provided in Section 6). We show the curves for two definitions of short instances, those for which document length was below 70 and 10 words, respectively. It is clear that shorter instances are more difficult to classify correctly than longer ones for this data set.

To improve the classification performance over short documents we propose a collaborative filtering (CF) technique [21] to predict the 'missing' terms for training instances with very few features. CF bears some similarity to query expansion whereby it uses the feature co-occurrence

DOI: 10.1002/spe

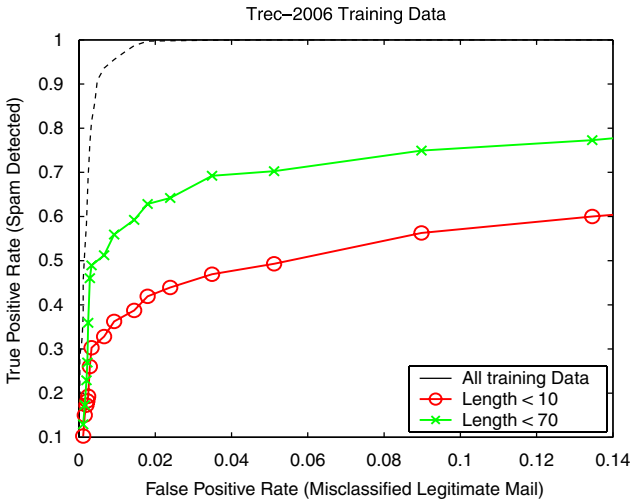

Figure 2. Motivation for using collaborative filtering for short instances. In the training set, short instances are more likely to be misclassified than longer ones. naive Bayes is used for learning here.

information when making recommendations [22]. CF techniques have been used widely in commercial recommender systems. In the context of TC, using CF for augmenting the feature space has also been observed to have a strong positive effect on the classification performance [23]. It should be noted that CF techniques are similar to dimension reduction methods (e.g. singular value decomposition (SVD) [24]) in terms of expanding short instances and shrinking long instances. One major difference is that by finding the latent (lower) dimension for feature space representation, SVD and other dimension reduction methods usually modify the original values in the feature vectors, while CF only suggests new values to the missing terms. The general process of CF consists of two steps:

1. *Collaborative*: Find instances that are similar to the target instance.
2. *Filtering*: Use the data from the existing instances to infer the data for the target instance.

The technique used in our approach can be described as follows. We first separate the training documents into regular and short documents (using document length threshold L_0). For the short documents, CF is then performed within each class. To find similar documents, we first rank all terms in class c according to a term weighting function f, i.e. $f(t_c) \rightarrow t'_c = \{t'_{c1}, \ldots, t'_{cm}\}$. Then for each short document $x^{(d)}$ in class c, we find the top g documents that share the top k common terms $\{t'_{c1}, \ldots, t'_{ck}\}$ with $x^{(d)}$, and *recommend* the terms from these documents to $x^{(d)}$, up to a limit of L_c terms. In our experiments, we used small k values such as 1 and 2. These augmented documents are then merged with regular documents before model learning continues. The procedure is outlined in Algorithm 1. To speed up the training on sparse data, we also indexed documents by their terms, which is similar to the inverted index method used in information retrieval.

One of the major costs of CF, regardless of the distance metric (e.g. Euclidean distance or cosine similarity), is the calculation of the weight matrix of all instances, which usually runs in $O(nm^2)$ time, where n is the number of documents and m the total number of features. This calculation is very expensive hence, it is normally done off-line. In our case, we restrict the number of top features

DOI: 10.1002/spe

Algorithm 1 Class-dependent Collaborative Filtering

1. **Input:** training data $(x^{(i)}, y^{(i)})_1^n, y^{(i)} \in \{1, \ldots, C\}$
2. length threshold: L_0, L_c
3. common term threshold: m
4. **Initialize** $S(f) \leftarrow \emptyset, f \in \{1, \ldots, C\}$
5. Begin Pre-processing Phase
6. **for** each $x^{(d)}$
7. **if** length$(x^{(d)} < L_0)$
8. $S(y^{(d)}) \leftarrow S(y^{(d)}) \cup x^{(d)}$
9. **end if**
10. **end for**
11. Begin Collaborative Filtering Phase
12. **for** each $S(f)$
13. **for** each $x_{fj} \in S(f)$
14. **do**
15. find the next most similar x'_{fj} that shares top m common ranked terms with x_{fj}
16. add term $t'_{fjk} \in x'_{fj}$ to x_{fj} **if** $t'_{fjk} \notin x_{fj}$
17. **until** length$(x_{fj} \geq L_c)$
18. **end for**
19. **end for**
20. **output** new training data $(x^{(d)'}, y^{(d)'})_1^n$

k to a very small number (i.e. $k \ll m$ and $k \ll n$) and the terms are ranked on a class-by-class basis, which reduces the cost of sorting to $O(m \log m)$. In all, the running time of the CF is bounded by $O(n + m \log m)$. Another reason for using only top-ranked features is because those features tend to be more important than others, e.g. documents of the same spam campaign may contain many random noise words while still sharing the same key spam words, which are likely to be considered as top-ranked features.

5. A TWO-STAGE CASCADED MODEL

The idea of combining multiple classifiers has been studied for decades. Several successful methods have been proposed in the literature, including Bayesian model averaging [25], bagging [26], boosting [27], as well as other ensemble techniques [28]. In most cases, these techniques rely on building multiple parallel classifiers over the same data set, or several highly related data sets. In contrast, we consider a cascade arrangement, where data used to train individual models represent subsamples of the original set, depending in composition on the previous elements of the cascade.

Cascaded models proposed in the literature [9,29] form either a chain or a tree. At each stage of a chain cascade, an instance can either be classified or passed to the next stage of the chain. Conversely, in tree cascades, the internal node classifiers direct the instances down the tree, with the leaf classifiers performing the final classification. In either case, each classifier of the cascade ensemble is trained using a different distribution of the data. Differences between individual methods depend to a large extent on how the data are split and what controls the size of the chain/tree. For example, in [30] a tree of NB models is built, where at each node the training data are further split

DOI: 10.1002/spe

according to the default threshold of the NB corresponding to that node. Splitting continues until an F-measure-based quality criterion stops improving. A two-stage cascade proposed in [9] sets the splitting threshold of the first classifier so as to satisfy a certain very low value of the FPR. This aims to ensure that the overall ensemble performs well in applications requiring low FP rates.

Our approach is motivated by those of Liu *et al*. [30] and Yih *et al*. [9]. Specifically, our two-stage model employs three classifiers, one at the first stage and two at the second stage. The base classifier $C0$ is trained using the entire training set. Each training instance is then scored according to how likely it is to belong to the positive class and the threshold is chosen to split the instances into two partitions, which are then used to induce classifiers $C1$ and $C2$. During classification, a test instance is first sent to the base classifier that calculates its score and then forwards it to either $C1$ or $C2$ based on the score value vis-a-vis the threshold. This resembles the tree structure proposed in [30], but in our case we deviate from using the default decision threshold as the splitting criterion and, as in [9], we choose the splitting threshold so as to satisfy certain FPR constraints. The intuition behind using FPR as the splitting criterion is to keep control of the misclassification cost. Generally, the goal of model induction is to minimize the expected loss, which with different misclassification costs for each class, in the context of spam filtering can be expressed as [31,32],

$$\min \left(\underbrace{N_{FP} * COST_{FP}}_{\texttt{misclassified good mails}} + \underbrace{N_{FN} * COST_{FN}}_{\texttt{misclassified spam}} \right) \tag{9}$$

However, given that the exact cost values are unknown, we consider the alternative criterion of maximizing the TP rate given a low target value of the FP rate. For each possible choice of the decision threshold of $C0$, this implies a joint optimization of thresholds of $C1$ and $C2$, so as to maximize the TP s.t. $FP \leq FP_u$. Note that locally tuning the thresholds of $C1$ and $C2$ does not necessarily guarantee a global optimum, which can be found efficiently, however, by solving the following linear programming problem:

$$\begin{aligned}
\texttt{Given} \quad & FP_u, \\
\texttt{Choose} \quad & \text{decision threshold of } C0 \\
\max \quad & TP_1 + TP_2 \\
\texttt{s.t.} \quad & FP_1 + FP_2 \leq FP_u \\
\texttt{where} \quad & TP_1 + FP_1 = P_1, TP_2 + FP_2 = P_2 \\
& P_1 + P_2 = P, FP_1 \geq 0, FP_2 \geq 0
\end{aligned} \tag{10}$$

Here, FP_1, TP_1 and FP_2, TP_2 denote the FP and negative counts of classifiers $C1$ and $C2$, respectively. Rather than considering all possible splitting thresholds of $C0$, we instead fix it heuristically such that $FP_0 = FP_u$. As a result, however, the training data received by $C1$ and $C2$ are likely to be class-imbalanced, when compared with the data seen by $C0$. For example, Figure 3 shows the imbalance ratio (no. of instances in minority class vs no. of instances in majority class) of the Hotmail training data after the splitting by the base classifier. All three data sets show a

DOI: 10.1002/spe

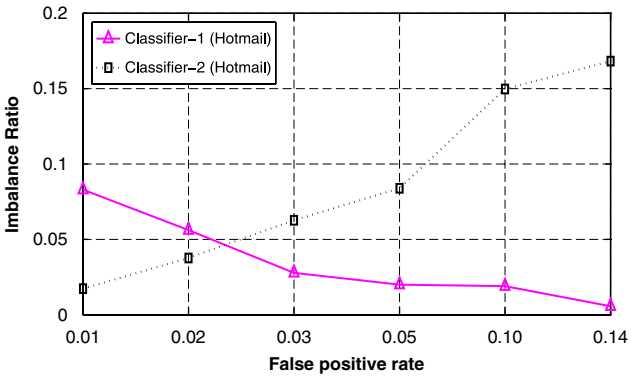

Figure 3. The imbalance ratio (no. of instances in the minority class vs no. of instances in the majority class) of the second-stage classifiers after splitting. Highly skewed distributions of two classes can be observed from the Hotmail data set (similar observation holds for the other two data sets).

Algorithm 2 Cascaded Binary Classifier

1. **Input:** training data $(x^{(d)}, y^{(d)})_1^n$, FP_u
2. **Begin Training Phase**:
3. Train a classifier $C0$ for the training data, get the score $(R^{(i)})_1^n$
4. Decide the cut point r based on FP_u
5. Separate the training data into two partitions:
6. 1. $R^{(i)} < r : T_1 = (x^{(j)}, y^{(j)})_1^{n_1}$
7. 2. $R^{(i)} \geq r : T_2 = (x^{(k)}, y^{(k)})_1^{n_2}$
8. Apply SMOTE sampling with $k = 5$ to T_1 and T_2
9. Train two classifiers $C1, C2$ for T_1, T_2
10. Decide the class boundary r_1, r_2 based on the criteria:
11. maximize $TP_1 + TP_2$,
12. subject to: $FP_1 + FP_2 \leq FP_u$
13. **End Training Phase**
14. **Begin Test Phase**:
15. for each test case $x^{(t)}$
16. 1. Apply $C0$ to get its score $r^{(t)}$
17. 2. if $r^{(t)} < r$ then send to $C1$, otherwise $C2$
18. 3. get its final label
19. **End Test Phase**

very high imbalance ratio which could lead to problems during parameter estimation. Previous work has shown that the class-imbalance problem usually leads to poor performance [33]. The minorities class(es) are usually underestimated or totally ignored, which may be the major reason for misclassifying unseen instances in the future. A common solution is to under or over sample the training data, e.g. via random majority under-sampling, random minority over-sampling, cluster-based over-sampling, synthetic minority over-sampling technique (SMOTE) [34]. In our case, we apply SMOTE as our sampling technique. SMOTE first finds the k nearest neighbors for each

DOI: 10.1002/spe

instance in the minority class, then generates artificial instances in the direction of the neighbors. We use the setting of $k=5$ as recommended by the original paper.

Details of this cascaded approach are shown in Algorithm 2.

5.1. All-in-one classifier

The three strategies we proposed, i.e. correlation-based term weighting, class-dependent CF, and cascaded classification, attempt to improve the result of classification from three different perspectives. It is thus natural to expect that combining them might lead to a stronger classifier. We propose the following framework that leverages these three techniques:

1. Use ACR (Equation (8)) as the term weighting function (assuming $L1$ document normalization).
2. Apply the CF technique (Algorithm 1) to predict missing terms for short training instances, with ACR used for scoring terms.
3. Train a cascaded model (Algorithm 2) on the augmented training data.
4. Use the resulting model for classifying new instances.

6. EMPIRICAL ANALYSIS

6.1. Data sets

In order to evaluate the techniques proposed in this paper, we chose two benchmark data sets as well as a real-world email data set from Hotmail. The two benchmark data sets come from the 2005 and 2006 TREC Spam Filtering Track [35,36]. The Hotmail data were collected from about 100 000 users, who labeled each message as either good or spam. Statistics of these three collections can be found in Table I. While the two benchmark data sets are relatively noise free, the Hotmail data suffers from approximately 3–5% class noise due to labeling errors. Only subject lines and message bodies of the emails were used during the feature extraction process. The messages were represented as bags of words and, following [17], in-document feature frequency was ignored since binary feature representation tends to be more beneficial in spam filtering [37].

6.2. Methodology

ROC curve [38] has been widely used to visualize classifier performance, with the area under the curve (AUC) providing a figure of merit capturing the quality of a classifier under all possible operating conditions. Since we are particularly interested in the results at very low FPRs, we focus on the FPR range of $[0, 0.14]$ and use the measurement $AUC_{0.1}$ as in [17], which corresponds to the area in the ROC plot

$$\{(FPR, TPR) : FPR \in [0, 0.1], TPR \in [0, 1]\} \tag{11}$$

For each data set three sets of experiments are performed:

1. We compared the ACR term weighting function (Equation (8)) with the best performer reported in [17], as well as with several other functions, including hybrids, all of which are listed in Table II. Following [17], the performance of NB was also compared with that of a linear

DOI: 10.1002/spe

Table II. Term weighting functions and their hybrids.

Function	Denoted by	Mathematical form
Inverse Doc Frequency	$IDF(x_k)$	$\log((2+\max DF)/(1+DF(x_k)))$
Information Gain	$IG(y,x_k)$	$-\sum_{c\in C,\bar{C}} P(c)\sum_{i=1}^{n} P(x_i\|c)\log P(x_i\|c)$
Correlation	$ACR(y,x_k)$	$\log(DF(x_k))\cdot\|\rho(y,x_k)\|$
Absolute Log-Odds	$ALO(x_k)$	$\left\|\log\frac{P(x_k\|c)}{P(x_k\|\bar{c})}\right\|$
Hybrid-1	$ALO*IDF$	$\frac{\|ALO(x_k)*IDF(x_k)\|}{\sum_j\|ALO(x_j)*IDF(x_j)\|}$
Hybrid-2	$IDF*ACR$	$\frac{\|IDF(x_k)*ACR(y,x_k)\|}{\sum_j\|IDF(x_j)*ACR(y,x_j)\|}$
Hybrid-3	$IDF*IG*ALO*ACR$	$\frac{\|IDF(x_k)*IG(x_k,y_i)*ALO(x_k)*ACR(y,x_k)\|}{\sum_j\|IDF(x_j)*IG(x_j,y)*ALO(x_j)*ACR(y,x_j)\|}$

SVM [39] using binary features. The optimal parameters of SVM were chosen by performing a 5-fold cross validation on the training data set.
2. The proposed CC technique was applied to the training set prior to NB model induction and its impact on the test set performance was measured.
3. Finally, the two-stage cascaded model was created and compared with the other models as well as to the cascaded model proposed by Yih *et al.* [9].

In all our experiments, $L1$ document length normalization was applied to both training and test data [15,17].

6.3. ACR vs ALO

Figure 4 presents the results of applying different term weighting functions for the three data sets considered. ACR clearly outperforms IDF-based methods for all cases (AUC). For the Hotmail data, NB with both ACR and IDF achieved better results than the SVM.

It is worth noting that NB with ACR performs particularly well in the very low-FP region of (0, 0.02). It either clearly outperforms SVM (for Trec-2006 and Hotmail data), or has very close results to SVM (for Trec-2005 data).

6.4. ACR vs Hybrids

Figure 5 presents the results of ACR as well as five other term weighting functions listed in Table II. ACR always outperforms others, followed by Hybrid 1 (ACR+IDF). The combination of all term weighting functions (Hybrid 4), however, does not show clear improvement over the simple ACR approach. The IG method, on the other hand, always behaves the worst.

The results can be interpreted as follows. First of all, the proposed hybrid methods generally perform better than a purely supervised term weighting function, i.e. IG, which shows the effectiveness of the combination of two or more functions. Second, ACR outperforms other hybrids in most of the scenarios. We believe that this can be explained by the orthogonal mixture of two functions, i.e. the unsupervised DF and the supervised correlation method, the first of which counts

DOI: 10.1002/spe

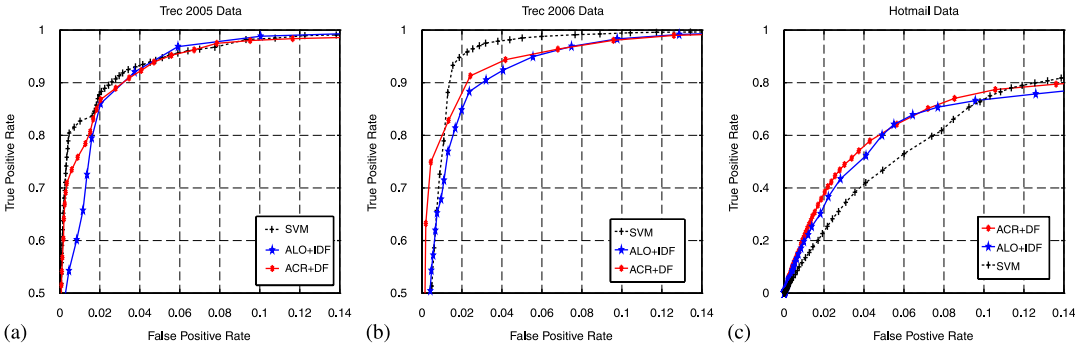

Figure 4. Comparison of two term weighting methods, ACR and ALO+IDF, used for naive Bayes. The latter showed the best performance in previous work using $L1$ normalization. SVM results are also included for reference: (a) Trec-2005; (b) Trec-2006; and (c) Hotmail.

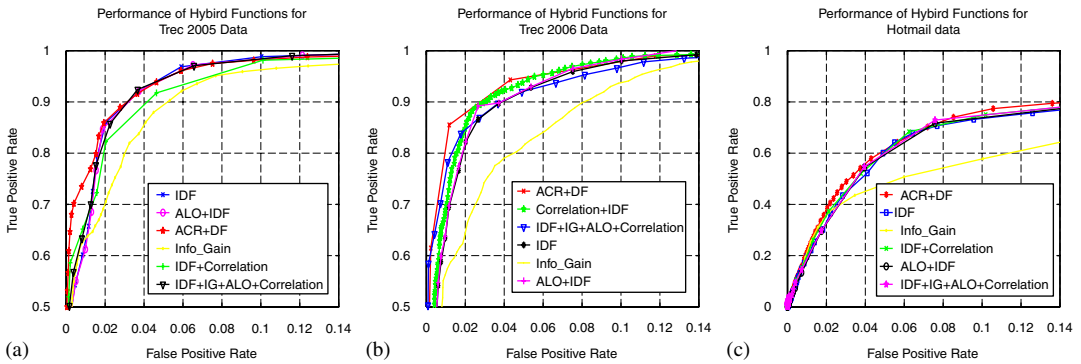

Figure 5. Comparison of different hybrid methods. Our correlation-based method shows very competitive results and outperforms others especially in very low-FP regions. (a) Trec-2005; (b) Trec-2006; and (c) Hotmail.

only to the occurrences of features, while the other one considers labels from the training set. This result indicates that exhaustively combining term weighting functions may not lead to optimal performance in practice.

6.5. The impact of CF

In this section we evaluate the impact of using CF. Based on the results thus far, we use ACR (Equation (8)) as the term weighting function. Figure 6 shows the results with and without applying collaborative filtering to the training data. It is clear that after 'recovering' the missing terms for short instances, the ROC curves improve for all three data sets. We also tuned parameters L_0, L_c, and m over a range of values. Specifically, we chose $L_0 \in [5, 30]$, $L_c \in [20, 100]$, and $m \in [1, 10]$. Empirically, we found that setting $L_0 = 10$, $L_c = 30$, and $k = 3$ (i.e. finding instances with less than 10 terms and augmenting them to 30 terms, where the top 3 ranked terms are used to calculate the weight matrix for finding similar instances) lead to the optimal results. However, other combinations of values did not decrease the performance significantly.

DOI: 10.1002/spe

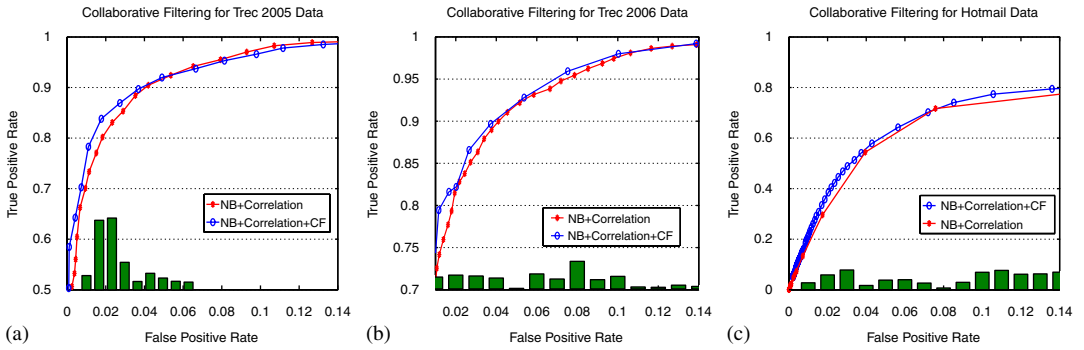

Figure 6. Collaborative filtering results on the: (a) Trec-2005; (b) Trec-2006; and (c) Hotmail data sets. The optimal parameters are $L_0 = 10$, $L_c = 30$ and $k = 3$. The bottom bar charts demonstrate the relative improvement after applying collaborative filtering.

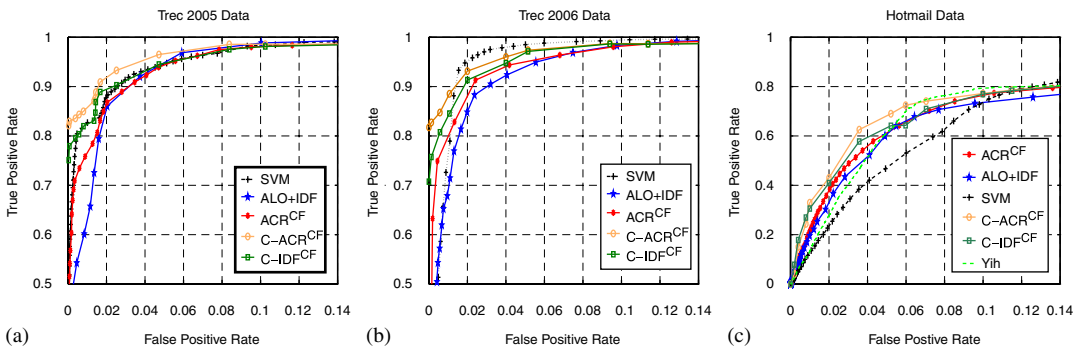

Figure 7. Comparison of cascaded models. The cascaded model from Yih *et al.* [9] is also compared in the Hotmail data. C-ACR^{CF} indicates the best performance among all in most cases: (a) Trec-2005; (b) Trec-2006; and (c) Hotmail.

6.6. Comparison between cascaded models

The ROC curves of the cascade models considered are shown in Figure 7, while Table III lists their corresponding $AUC_{0.1}$ scores (with the score of 1 indicating the optimum). We found that Yih *et al.*'s model [9] does not improve obviously over a simple $L1$ normalized NB method on either Trec-2005 or Trec-2006 data, and was therefore excluded from the comparison.

Overall, it can be seen that C-ACR^{CF} outperforms other models in most scenarios. It gains a 13 and 1% improvement in $AUC_{0.1}$ over SVM for Trec-2005 and Hotmail, respectively. While it can be observed that the two-stage cascaded model does show an advantage over the simple naive Bayes, NB with either ACR or IDF as the term weight function beats SVM in most cases. SVM shows better results for Trec-2006 data, but only in the FP region of [0.01, 0.09], and its $AUC_{0.1}$ score (0.9738) is almost the same as the two cascaded models (0.9704 and 0.9711). For Hotmail, the model of Yih *et al.* performs the best among all in the region of [0.07,0.12]. However, its true positive rate drops dramatically when the FPR falls below 0.05. And its $AUC_{0.1}$ is similar to a single ACR^{CF} approach without applying the cascaded model.

DOI: 10.1002/spe

Table III. $AUC_{0.1}$ scores within false positive region [0,0.1].

	IDF	IG	Hybrid-1	Hybrid-2	Hybrid-3	ACR
Hotmail	0.5740	0.4977	0.5769	0.5772	0.5770	0.5865
Trec-2005	0.9312	0.7977	0.9450	0.9453	0.9452	0.9468
Trec-2006	0.9517	0.7643	0.9516	0.9528	0.9526	0.9531
	ACR^{CF}	C-IDF^{CF}	C-ACR^{CF}	Yih	SVM	
Hotmail	0.5912	0.6013	0.6129	0.5864	0.4833	
Trec-2005	0.9494	0.9500	0.9561	—	0.9482	
Trec-2006	0.9535	0.9704	0.9711	—	0.9738	

Higher score indicates better performance, where 1 is optimal. Hybrid-1 corresponds to ALO+IDF. Hybrid-2 corresponds to IDF+ACR. Hybrid-3 combines all functions, i.e. IDF+IG+ALO+ACR. ACR^{CF} is the result after applying CF. C-IDF^{CF} and C-ACR^{CF} indicate the cascaded models, with IDF and ACR the weight functions, and used CF to augment the feature space.

Noticeably, our cascaded model performs extremely well in the very low FP region. It can be seen that when the FPR drops to nearly 0 for the Trec-2005 and Trec-2006 data sets, our two cascaded models C-IDF^{CF} and C-ACR^{CF} are still able to achieve fairly good results (i.e. $TPR > 70\%$).

Interestingly the results achieved by our best model combination, i.e. C-ACR^{CF}, are comparable to the best reported results for the TREC data sets, namely those achieved by an online SVM classifier ROSVM [40], while our results required significantly less training time‖.

6.7. Statistical significance tests

Finally, we rigorously conduct significance tests between all classifiers to report whether they perform equally. We are also interested to see whether our best proposed method C-ACR^{CF} performs significantly better than the others. The most appropriate framework for multiple-classifier multiple-data set significance testing was proposed in [42]. In our setting, we train 9 classifiers for each of the three data sets. We compare the difference of true positive rates achieved by the 9 classifiers in the FP region of [0,0.14]. We follow the Friedman test in [42] which states that all classifiers perform equivalently as the null hypothesis. The statistic is calculated as

$$\chi_F^2 = \frac{12N}{c(c+1)}\left(\sum_j R_j^2 - \frac{c(c+1)^2}{4}\right), \quad R_j = \frac{1}{N}\sum_i r_i^j \tag{12}$$

with N denoting the number of data sets, c the number of compared classifiers, and r_i^j the rank of the jth classifier on the ith data set. An alternative test was proposed by Iman and Davenport [43] to overcome the conservativeness of the Friedman test. Our results reject both null hypotheses with p-values less than 0.05 and we therefore concluded that the 9 classifiers do not behave equally. Consequently, we compare the performance of C-ACR^{CF} with every other classifier by using

‖For Trec-2005 and Trec-2006 data, our algorithm runs 76 and 43 s, respectively, on a single PC with 3 GB memory and 2.66 GHz CPU, comparing with the reported 24 720 and 18 541 s in [40]. For Hotmail, a linear SVM takes more than 14 h to train, while our model only requires less than 5 min. It should be noted though that using one of the new fast implementations of linear SVM [41] would have reduced the SVM time quite a bit.

DOI: 10.1002/spe

Table IV. Ranks of 9 classifiers on the three test data sets.

	Trec-2005	Trec-2006	Hotmail	Aver rank
IG	9	8	9	8.67
IDF	8	9	8	8.33
Hybrid-1	7	6	7	6.67
Hybrid-2	5	7	6	6
Hybrid-3	6	5	5	5.33
ACR	4	4	4	4
ACR^{CF}	3	3	3	3
C-IDF^{CF}	2	2	2	2
C-ACR^{CF}	1	1	1	1

Table V. p-values for significance tests of the classifiers. C-ACR^{CF} significantly outperforms IG, IDF and Hybrid-1. p-values that are significant (i.e. $p < 0.05$) are shown in bold fonts.

	IG	IDF	Hybrid-1	Hybrid-2
$\text{C-ACR}^{CF} >$	$\mathbf{p = 0.002}$	$\mathbf{p = 0.009}$	$\mathbf{p = 0.013}$	$p = 0.120$
	Hybrid-3	ACR	ACR^{CF}	C-IDF^{CF}
	$p = 0.152$	$p = 0.200$	$p = 0.255$	$p = 0.550$

the Bonferroni-Dunn test [44], which states that the performance is significantly different if the difference between the average ranks of two classifiers is greater than the critical difference (CD):

$$CD = q_\alpha \sqrt{\frac{c(c+1)}{6N}} \tag{13}$$

where q_α is calculated by dividing the Student's t statistic with $\sqrt{2}$. In our case, the value of the CD is 6.09 for 2-tailed test and 5.59 for 1-tailed test, at the 95% significance level.

Table V presents the p-values for all pairwise comparisons between C-ACR^{CF} and the other algorithms. It can be read as '*classifier C-ACR*^{CF} *outperforms*', with each column corresponding to the classifier that C-ACR^{CF} compares with. It can be observed that C-ACR^{CF} is significantly better than IG, IDF, and Hybrid-1. Meanwhile, Table IV lists the ranks of the 9 classifiers. Notice the consistency of the ranks of our proposed methods. By applying CF, our methods top the others for all three data sets. The three hybrid methods, however, do not exhibit small enough p-values that could also be used to conclude which one is the best among three.

7. RELATED WORK IN CASCADED MODELS FOR NB

Combining multiple classifiers for performance boosting traces back to the mid 1990s. Among these methods, Bayesian model averaging [25], bagging [26], boosting [27] and ensemble methods [28,45] have shown substantial performance improvements. Although ensembles of non-linear models (e.g. AdaBoost) usually outperform others [28], the high computational demand has yet prevented them from being used in large-scale real-world applications. Alternatively, ensembles of linear models and linear cascaded models [9] have earned much attention due to their efficiency and flexibility.

DOI: 10.1002/spe

Linear cascaded models work by sequentially combining a set of classifiers, each of which classifies a subset of the original data at different steps [46–50]. Particularly for NB, the Tree Augmented NB [51], which uses a simple Bayesian network structure, outperforms NB and C4.5 decision tree [52] on a large number of benchmark data sets. In the following, we present the related work on two recent improvements for the NB classifier and compare them with our model.

Wu's refinement model: A tree-based framework was proposed by Wu *et al.* [29] to handle the issue of model misfit in TC. Their model first trains a base classifier for the training set and classifies the data as either positive or negative. Their method then retrains a classifier using training examples from each of the predicted classes. By recursively repeating this procedure for a number of times, their model is capable of building a refinement tree to better fit the training data. Their procedure stops once the F-score of the training data stops improving. The resulting model may potentially include a substantial number of classifiers, whose performances are determined by the default threshold of the base classifier.

Yih's cascaded model: Rather than trying to combine several parallel classifiers to improve the performance [28], Yih *et al.* [9] advocated a two-stage cascaded model specifically for classifier improvements at a very low FPR. The first-stage classifier is used to remove instances that are either very evidently positive or very hard-to-classify negative. Past the first stage, only moderately hard and easy positives (spam), and moderately hard and very hard negatives (legitimate mails) are left, and the second-stage classifier is trained specifically to deal with these instances. To improve the performance, artificial examples are generated by sampling with replacement from the minority class. Two different learning algorithms were applied for this approach: logistic regression and NB. In comparison with the baseline classifiers, the proposed method gained 20 and 40% performance improvement, respectively.

Our proposed cascaded model differs from Liu, Wu and Yih's models in the following senses:

- Wu's model focuses on optimizing the F-score of the training data, which has no guarantee to work well for problems with highly asymmetric cost like spam filtering. Their method does not choose an optimized threshold for the base classifier, so that the model could be very complex with a large number of nodes grown from the root. Furthermore, their model may only achieve sub-optimal results since the sub-nodes could suffer from the class-imbalance problem with the growth of the tree, which usually leads to poor performance. Our model explicitly addresses these three issues.
- In Yih's model, the first classifier chooses a threshold that results in a low FPR but the instances above that threshold are left alone. While the base classifier in our model can choose any threshold for splitting the data, the two second-stage classifiers will automatically optimize the objective function based on this threshold.

Table VI summarizes the differences between these three models.

Table VI. Comparison of cascaded models.

	No. of classifiers	Splitting criteria	Optimization	Scalable?	Extensible?
Wu's model	Varies	Classifier-based	F-score	No	No
Yih's model	2	FP rate	None	Yes	Yes
Our model	3	FP rate	TP rate	Yes	Yes

Our model is similar to Yih's model in terms of the number of classifiers used and the splitting criteria. Both our model and Wu's use a optimization method that Yih's does not.

DOI: 10.1002/spe

8. CONCLUSIONS

In this paper we proposed several improvements to the NB classifier that make it well suited to applications requiring high precision, such as spam filtering. We introduced a term-weighting function based on the correlation measure, which was demonstrated to perform very well both on its own and as an alternative to the typical multiplicative aggregation of several term weighting functions. To address the problem of feature sparsity for short documents a class-dependent CF technique was proposed to expand their attribute vectors. This was shown to improve the classifier performance in the low-FP region. Finally, a novel two-stage NB cascade was introduced, which combines the ability to tackle the potential non-linearity of the decision boundary with an algorithm that jointly optimizes the decision thresholds of the terminal components of the cascade so as to achieve the best performance at a specified FP rate. Although the proposed techniques have been shown to be quite effective for NB, they are also applicable to other learners. Further investigation of their utility will be the subject of future work.

APPENDIX

Pseudo-code of training a cascaded Naive Bayes Model.

```
1
2  public class Hierarchical_NB {
3          public static int _traininglength = 0;
4          public static int _testlength = 0;
5          public static double _breakpoint = Double.MAX_VALUE;
6          public static double _breakpoint1 = 0;
7      public static double _breakpoint2 = 0;
8
9
10
11     procedure maximizeTP(array1, array2, totalFP){
12         length1 = array1.length;
13         length2 = array2.length;
14         int idx1 = -1;
15         int idx2 = -1;
16         int maxTP = -1;
17
18         for(i = 0; i < length1; i++){
19                 for(j = length2 - 1; j >= 0; j--){
20                         if(array1[i][1] + array2[j][1] > totalFP)
21                                 break;
22                         else if (array1[i][1] + array2[j][1] <= totalFP){
23                                 if(array1[i][2] + array2[j][2] > maxTP){
24                                         idx1 = i;
25                                         idx2 = j;
26                                         maxTP = (int)(array1[i][2] + array2[j][2]);
27                                 }
28                         }
29                         else{throw exceptions;}
30                 }
31         }
32         _breakpoint1 = array1[idx1][0];
33         _breakpoint2 = array2[idx2][0];
34
35     }
36
37         procedure calculateROC(double threshold){
38         double TP = 0, FP = 0, FN = 0, TN = 0;
39                 for( int i = 0; i < _length; i++){
40                     if(truelabel[i] == 1 && predictlabel[i] >= threshold){ // true positive
41                         TP ++;
```


DOI: 10.1002/spe

```
42                      }
43                      else if(truelabel[i] == -1 && predictlabel[i] >= threshold){ // false positive
44                          FP ++;
45                      }
46                      else if(truelabel[i] == -1 && predictlabel[i] < threshold){ // true negative
47                          TN ++;
48                      }
49                      else{
50                          FN ++;
51                      }
52                  }
53
54                  _TP = (int)TP;
55                  _FP = (int)FP;
56                  //System.out.println("[TP]\t[FP]\t[TN]\t[FN]");
57                  //System.out.println(TP+ "\t" + FP + "\t" + TN + "\t" + FN);
58                  TP = TP / (double)(TP + FN);
59                  FP = FP / (double)(FP + TN);
60                  //System.out.println(threshold + "\t" + FP + "\t" + TP);
61                  _threshold = threshold;
62                  _TPR = TP;
63                  _FPR = FP;
64      }
65
66      procedure base_classifier(training_file) {
67            String trainingfile   = mb.BootrapFile("2006/trec.trn.svm");
68          double[][] odds                 = nb.Training(trainingfile);
69          double[] samplerank     = nb.TestResult(odds,trainingfile);
70          _traininglength                 = nb._testlength;
71
72          double MAXAllowed = 0.03; // the maximum allowed FPR
73          int totalFP = 0;
74
75          System.out.println(samplerank.length+"_"+_traininglength);
76          roc.readFile(labelfile, predictfile);
77          myprint("total_POS:"+roc._POSITIVEFILES+"\tNEG:"+roc._NEGATIVEFILE);
78          for(double i = -4; i < 4; i+= 0.01){
79                  calculateROC(i);
80                  if(roc._FPR < MAXAllowed){
81                                  totalFP = roc._FP; // the total number of FP files classified by
                                    the base classifier
82                                  break;
83                          }
84          }
85                  //// end of finding the break even point
86
87
88
89
90
91                  ///// start separating training samples
92                  int[] trainingarray1 = new int[_traininglength];
93                  int[] trainingarray2 = new int[_traininglength];
94                  Arrays.fill(trainingarray1, -1);
95                  Arrays.fill(trainingarray2, -1);
96                  int indexfirst = 0, indexsecond = 0;
97                  for(int i = 0; i < _traininglength; i++){
98                          if(samplerank[i] > _breakpoint){
99                                  trainingarray1[indexfirst++] = i; // training array 1 contains
                                    instances with higher rank
100                         }
101                         else{
102                                  trainingarray2[indexsecond++] = i;
103                         }
104                 }
105
106                 //// end of separating training samples
107
108                 String firstfile                = separateFiles(trainingarray1, trainingfile);
    /// train a NB for the first file
109                 double[][] oddsfirstNB  = nb.Training(firstfile);
110                 double[] firstrank              = nb.TestResult(oddsfirstNB,firstfile);
    /// get the rank of training file for the first array
111
112
113
```


DOI: 10.1002/spe

```
111
112
113
114                 String secondfile               = separateFiles(trainingarray2, trainingfile); ///
115                         //double[][] oddssecondNB = nb.ConvertSVMtoMatrix(secondfile);
116                 double[][] oddssecondNB = nb.Training(secondfile);
117                 double[] secondrank     = nb.TestResult(oddssecondNB, secondfile);
    /// get the rank of training file for the second array
118
119
120                 ////////////////////////////////////////////////////////////////////
121                 ////////////////////// start set the threshold for each classifier
122                 ////////////////////// The objective is FP1 + FP2 <= FP,
123                 ////////////////////// and maximize TP1 + TP2 (> TP)
124                 ////////////////////////////////////////////////////////////////////
125
126                 roc = new ROC();
127
128                 double[][] ROC1 = new double[10000][3];  /// contains threshold, FP, TP,
129                 int ROC1length = 0;
130                 for(double i = bp1; i < 10; i+= 0.01){
131                         roc.calculateROC(i);
132
133                         if(roc._TP < 1){ // no need to go further
134                                 //myprint(roc._threshold+"\t"+roc._FPR+"\t"+roc._TPR);
135                                 break;
136                         }
137                         ROC1[ROC1length][0] = roc._threshold;
138                         ROC1[ROC1length][1] = roc._FP;
139                         ROC1[ROC1length][2] = roc._TP;
140                         ROC1length++;
141                         //myprint(roc._threshold+"\t"+roc._FP+"\t"+roc._TP);
142                 }
143
144                 myprint("length_of_ROC1_is:" + ROC1length);
145
146
147                 ////////////////////////////////////////////////
148                 /// the second classifier
149                 ////////////////////////////////////////////////
150
151                 double[][] ROC2 = new double[10000][3];  /// contains threshold, FP, TP
152                 int ROC2length = 0;
153                 for(double i = bp2; i < 10; i+= 0.01){
154                         roc.calculateROC(i);
155                         if(roc._TP < 1){ /// no need to go further
156                                 //myprint(roc._threshold+"\t"+roc._FPR+"\t"+roc._TPR);
157                                 break;
158                         }
159                         ROC2[ROC2length][0] = roc._threshold;
160                         ROC2[ROC2length][1] = roc._FP;
161                         ROC2[ROC2length][2] = roc._TP;
162                         ROC2length++;
163                         //myprint(roc._threshold+"\t"+roc._FP+"\t"+roc._TP);
164                 }
165
166                 myprint("length_of_ROC2_is:" + ROC2length);
167
168                 maximizeTP(ROC1, ROC1length, ROC2, ROC2length, totalFP);
169 }
```

REFERENCES

1. Lewis DD. Naive (Bayes) at forty: The independence assumption in information retrieval. *Proceedings of ECML-98, 10th European Conference on Machine Learning*, Chemnitz, DE, vol. 1398, Nédellec C, Rouveirol C (eds.). Springer: Heidelberg, DE, 1998; 4–15.
2. Eyheramendy S, Lewis D, Madigan D. *On the Naive Bayes Model for Text Categorization*, 2003; 332–339.

DOI: 10.1002/spe

3. Kołcz A. Local sparsity control for naive Bayes with extreme misclassification costs. *KDD '05*: *Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM Press: New York, NY, U.S.A., 2005; 128–137.
4. Cormack GV, Bratko A. Batch and online spam filter comparison. *Conference on Email and Anti-spam*, *CEAS 2006*, Mountain View, CA, July 2006.
5. Kołcz A, Alspector J. SVM-based filtering of e-mail spam with content-specific misclassification costs. *Proceedings of the Workshop on Text Mining* (*TextDM'2001*), San Jose, CA, U.S.A., 2001.
6. Metsis V, Androutsopoulos I, Paliouras G. Spam filtering with naive Bayes—Which naive Bayes. *Proceedings of the 3rd Conference on Email and Anti-spam* (*CEAS-2006*), Mountain View, CA, 2006.
7. Robinson G. A statistical approach to the spam problem. *Linux Journal* 2003; **2003**(107):3.
8. Ntoulas A, Najork M, Manasse M, Fetterly D. Detecting spam web pages through content analysis. *WWW '06*: *Proceedings of the 15th International Conference on World Wide Web*. ACM Press: New York, NY, U.S.A., 2006; 83–92.
9. Yih W, Goodman J, Hulten G. Learning at low false positive rates. *Proceedings of the 3rd Conference on Email and Anti-spam* (*CEAS-2006*), Mountain View, CA, 2006.
10. Domingos P, Pazzani M. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 1997; **29**(2–3):103–130.
11. McCallum A, Nigam K. *A Comparison of Event Models for Naive Bayes Text Classification*, 1998; 41–48.
12. Manning CD, Schütze H. *Foundations of Statistical Natural Language Processing*. The MIT Press: Cambridge, MA, 1999.
13. Chen SF, Goodman J. An empirical study of smoothing techniques for language modeling. *Proceedings of the Thirty-fourth Annual Meeting of the Association for Computational Linguistics*, Joshi A, Palmer M (eds.). Morgan Kaufmann Publishers: San Francisco, 1996; 310–318.
14. Rennie J. *Improving Multi-class Text Classification with Naive Bayes*, 2001.
15. Rennie J, Shih L, Teevan J, Karger D. Tackling the poor assumptions of naive Bayes text classifiers. *Proceedings of the Twentieth International Conference on Machine Learning* (*ICML-2003*). AAAI Press: Menlo Park, CA, 2003; 616–623.
16. Kim S-B, Han K-S, Rim HC, Myaeng SH. Some effective techniques for naive Bayes text classification. *IEEE Transactions on Knowledge and Data Engineering* 2006; **18**(11):1457–1466.
17. Kołcz A, Yih WT. Raising the baseline for high-precision text classifiers. *KDD '07*: *Proceeding of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM Press: New York, NY, U.S.A., 2007; 400–409.
18. Salton G, McGill MJ. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc.: New York, NY, U.S.A., 1986.
19. Debole F, Sebastiani F. Supervised term weighting for automated text categorization. *SAC '03*: *Proceedings of the 2003 ACM Symposium on Applied Computing*. ACM Press: New York, NY, U.S.A., 2003; 784–788.
20. Hall MA. Correlation-based feature selection for discrete and numeric class machine learning. *ICML'00*: *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, U.S.A., 2000; 359–366.
21. Breese JS, Heckerman D, Kadie C. Empirical analysis of predictive algorithms for collaborative filtering. *Uncertainty in Artificial Intelligence. Proceedings of the Fourteenth Conference* (*1998*). Morgan Kaufmann: Los Altos, CA, 1998; 43–52.
22. Hoashi K, Matsumoto K, Inoue N, Hashimoto K. Query expansion based on predictive algorithms for collaborative filtering. *SIGIR '01*: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press: New York, NY, U.S.A., 2001; 414–415.
23. Song Y, Zhou D, Huang J, Councill IG, Zha H, Giles CL. Boosting the feature space: Text classification for unstructured data on the web. *ICDM '06*: *Proceedings of the Sixth International Conference on Data Mining*. IEEE Computer Society: Washington, DC, U.S.A., 2006; 1064–1069.
24. Trefethen LN, Bau D. *Numerical Linear Algebra*. SIAM: Philadelphia, PA, June 1997.
25. Raftery AE, Madigan D, Hoeting JA. Bayesian model averaging for linear regression models. *Journal of the American Statistical Association* 1997; **92**(437):179–191.
26. Breiman L. Bagging predictors. *Machine Learning* 1996; **24**(2):123–140.
27. Schapire RE, Freund Y, Bartlett P, Lee WS. Boosting the margin: A new explanation for the effectiveness of voting methods. *Proceedings of the 14th International Conference on Machine Learning*. Morgan Kaufmann: Los Altos, CA, 1997; 322–330.
28. Dietterich TG. *Ensemble Methods in Machine Learning* (*Lecture Notes in Computer Science*, vol. 1857). Springer: Berlin, 2000; 1–15.
29. Wu H, Phang TH, Liu B, Li X. A refinement approach to handling model misfit in text categorization. *KDD '02*: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM: New York, NY, U.S.A., 2002; 207–216.
30. Liu B, Xia Y, Yu PS. Clustering through decision tree construction. *CIKM '00*: *Proceedings of the Ninth International Conference on Information and Knowledge Management*. ACM Press: New York, NY, U.S.A., 2000; 20–29.
31. Elkan C. The foundations of cost-sensitive learning. *IJCAI*, Seattle, Washington, U.S.A., 2001; 973–978.

DOI: 10.1002/spe

32. Bach FR, Heckerman D, Horvitz E. Considering cost asymmetry in learning classifiers. *Journal of Machine Learning Research* 2006; **7**:1713–1741.
33. Tang L, Liu H. Bias analysis in text classification for highly skewed data. *ICDM '05*: *Proceedings of the Fifth IEEE International Conference on Data Mining*. IEEE Computer Society: Washington, DC, U.S.A., 2005; 781–784.
34. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. *SMOTE*: *Synthetic Minority Over-sampling Technique*, 2002; 321–357.
35. Cormack G, Lynam T. TREC 2005 spam track overview. *Proceedings of TREC 2005—The Fourteenth Text REtrieval Conference*, 2005.
36. Cormack G. The TREC 2006 spam filter evaluation track. *Virus Bulletin*, vol. 1, 2006.
37. Blanzieri E. Instance-based spam filtering using svm nearest neighbor classifier. *Proceedings of FLAIRS 2007*, Key West, FL, 2007; 441–442.
38. Fawcett T. An introduction to ROC analysis. *Pattern Recognition Letters* 2006; **27**(8):861–874.
39. Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. *COLT '92*: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. ACM Press: New York, NY, U.S.A., 1992; 144–152.
40. Sculley D, Wachman GM. Relaxed online SVMs for spam filtering. *SIGIR '07*: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press: New York, NY, U.S.A., 2007; 415–422.
41. Joachims T. Training linear svms in linear time. *KDD '06*: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM: New York, NY, U.S.A., 2006; 217–226.
42. Demšar J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 2006; **7**:1–30.
43. Iman R, Davenport J. Approximations of the critical region of the Friedman statistic. *Communications in Statistics—Theory and Methods* 1980; **A9**(6):571–595.
44. Dunn OJ. Multiple comparisons among means. *Journal of the American Statistical Association* 1961; **56**:52–64.
45. Opitz D, Maclin R. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research* 1999; **11**:169–198.
46. Gama J, Brazdil P. Cascade generalization. *Machine Learning* 2000; **41**(3):315–343.
47. Garcia-Pedrajas N, Ortiz-Boyer D, del Castillo-Gomariz R, Hervas-Martinez C. Cascade ensembles. *Computational Intelligence and Bioinspired Systems*. Springer: Berlin/Heidelberg, 2005; 598–603.
48. Zhang P, Bui TD, Suen CY. A novel cascade ensemble classifier system with a high recognition performance on handwritten digits. *Pattern Recognition* 2007; **40**(12):3415–3429.
49. Heisele B, Serre T, Prentice S, Poggio T. Hierarchical classification and feature reduction for fast face detection with support vector machines. *Pattern Recognition* 2003; **36**(9):2007–2017.
50. Giusti N, Masulli F, Sperduti A. Theoretical and experimental analysis of a two-stage system for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2002; **24**(7):893–904.
51. Friedman N, Geiger D, Goldszmidt M. Bayesian network classifiers. *Machine Learning* 1997; **29**(2–3):131–163.
52. Quinlan JR. *C4.5*: *Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, U.S.A., 1993.

DOI: 10.1002/spe