DESIGN AND IMPLEMENT COST-SENSITIVE EMAIL FILTERING ALGORITHMS

Wenbin Li[1,2], Chunnian Liu[1] and Yiying Chen[2]
1. Beijing University of Technology, Beijing 100022,China
2. ShiJiaZhuang University of Economics,ShiJiaZhuang 050031,China

Abstract: The growing problem of unsolicited bulk e-mail, also known as "spam", has generated a need for reliable anti-spam e-mail filters. We introduce seven filtering algorithms: Naive Bayesian (NB), Decision Tree (DT), AdaBoost, ANN, SVM, VSM and KNN. Design considerations and implementation issues of these filters are discussed, such as how to get cost-sensitive NB, SVM, VSM, KNN. Using two relatively large amounts of real personal E-mail data, a comprehensive comparative study based on a cost-sensitive measure we approved was conducted using above seven filters. The study includes the effect of feature subset size, training-corpus distribution, issues that have not been explored in previous experiments. The comparative results show that cost-sensitive filters such as NB, SVM, VSM and KNN have fewer count of misclassifying legitimate when relative parameters, feature subset size and training dataset's distribution are reasonable.

Key words: Spam E-mail, E-mail Filter, Cost-Sensitive

1. INTRODUCTION

It's known that spam Emails has generated a need for reliable anti-spam email filters. Filtering spam-email is a problem of text categorization (TC), it's said that it's the problem of automatically assigning predefined categories {spam, legitimate} to new emails [11]. So many researches addressed email filtering by casting it in the general framework of TC. However, email filtering is different from TC. Firstly, mistakenly treating a legitimate message as spam can be a more severe error than treating a spam message as legitimate, so filter should take this unbalanced misclassification cost into account. Secondly, the main barrier to seeing automated mail filtering becoming commonplace is the resolution issues regarding the

implementation of mail filters and their integration into e-mail clients [4, 8]. Some of the important issues regarding mail filters include speed efficiency, database size and the collection of supervised training data. Time-consuming training or classification can degrade the interface. A large database to store the classification model may limit the user base. Also, a mail filter is only likely to be used if no additional effort is required to reap the benefits.

This paper is a comprehensive comparative study of email filtering algorithms. The focus is on cost-sensitive evaluation and effecting issues of filters. Seven methods were evaluated, including Naive Bayesian (NB),Decision Tree (DT), AdaBoost, Artificial Neural Network (ANN), Support Vector Machine (SVM), Vector Space Model (VSM) and K-Nearest Neighbor (KNN). In fact, some researches have done such works before [2, 9], but their studies are not full-scale on the one hand, on another hand, they didn't discuss specified problems in email filter.

The remainder of the paper is organized as follows. Section 2 discusses benchmark corpus and cost-sensitive evaluating measures. Section 3 presents our design and implementation of above seven filters. The experiments and results are presented in Section 4. Section 5 concludes the paper.

2. CORPUS, COST-SENSITIVE EVALUATING METHOD

Research on text categorization has benefited significantly from the existence of publicly available, manually categorized document collections, like the Reuters corpus, that have been used as standard benchmarks. Producing similar corpora for anti-spam filtering is more complicated, because of privacy issues. Publicizing spam messages does not pose a problem, since spam messages are distributed blindly to very large numbers of recipients, and, hence, they are effectively already publicly available. Legitimate messages, however, in general cannot be released without violating the privacy of their recipients and senders.

One way to bypass privacy problems is to experiment with legitimate messages collected from freely accessible newsgroups, or mailing lists with public archives. Ling-Spam, the earliest of our benchmark corpora, follows this approach. It consists of 481 spam messages received by the first author, and 2412 legitimate messages retrieved from the archives of the Linguist list, a moderated and, hence, spam-free list about linguistics. The SpamAssassin public mail corpus is similar to Ling-Spam, in that its legitimate messages are publicly available. It contains 1897 spam and 4150 legitimate messages, the latter collected from public fora or donated by users with the understanding that they may be made public. An alternative solution,

adopted in pu1, is to release benchmark corpora that each consists of messages received by a particular user, after replacing each token by a unique number throughout the corpus.The mapping between tokens and numbers is not released, making it extremely difficult to recover the original text. PU1 contains 480 spam and 610 legitimate messages.

Table 1. Cost Matrix

	Actual Legitimate	Actual Spam
Predict Legitimate	c_{00}	c_{01}
Predict Spam	c_{10}	c_{11}

Table 2. Contingency Table

	Actual Legitimate	Actual Spam
Predict Legitimate	f_{00}	f_{01}
Predict Spam	f_{10}	f_{11}

This paper used Ling-Spam, PU1 as experimental data set. (downloaded from http://www.mlnet.org/cgi-bin/mlnetois.pl/?File=datasets.html).

Becaue email filtering is a cost-sensitive classification decisions with two-class, so PRECISION and RECALL can't reflect efficiency a filter h well and truly, because between high precision and high recall exists a trade-off. So we define the "Spam ->Legitimate Error Rate" SLER(h) and "Legitimate -> Spam Error Rate" LSER(h) of a filter h as follows:

$$SLER(h) = c_{01} * f_{01} / (c_{01} * f_{01} + f_{11}) \quad (1)$$

$$LSER(h) = c_{10} * f_{10} / (c_{10} * f_{10} + f_{00}) \quad (2)$$

"Error Rate" ER(h) and "Correct Rate" CR(h) are:

$$ER(h)=SLER(h)+LSER(h) \quad (3)$$

$$CR(h)=2-ER(h) \quad (4)$$

Here, c_{01}, c_{10} is give in cost matrix shown as Table 1. Parameters f_{01}, f_{01}, f_{00}, f_{11} is given in Table 2 which provides a convenient display of the prediction behavior for Email filtering. Each cell of table 2 represents of the four possible outcomes of a prediction for an example. The diagonal cells count how often the prediction was correct. The off-diagonal entries show the frequency of prediction errors. The sum of all cells equals the total number of predictions.

3. DESIGN AND IMPLEMENT COST-SENSITIVE FILTERS

The following lists the basic design decisions for the experiments in this paper.

- Words are chosen as the basic representational units. Words are defined as non-

whitespace strings enclosed by whitespace characters.

- Only those words occurring in the training data are considered as potential features. And feature selection method is Information Gain(IG)[10].
- Stemming, stopwords is not used.
- NB, DT, AdaBoost, ANN, and SVM use the binary vector representation, while KNN and VSM adopt TF-IDF [6] vector representation.

3.1 NB

Given a new (unseen) document d, C= {c_0 = "legitimate", c_1 = "spam"}, classification of d is performed by computing the posterior probability of each class, given d, by applying Bayes' rule:

$$P(c_j \mid d;\theta) = P(c_j \mid \theta)P(d \mid c_j;\theta) / P(d \mid \theta) \tag{5}$$

The classifier simply selects the class with the highest posterior probability. Note that P(d|b_) is the same for all classes, thus d can be classified by computing:

$$c_d = \arg\max_{c_j \in C} P(c_j \mid \theta)P(d \mid c_j;\theta) \tag{6}$$

By making the Naive Bayes assumption, we can compute the probability of a document given a class from the probabilities of the words given the class [1] (V is feature set, $B_i \in \{0,1\}$ indicates whether word w_i occurs at least once in d):

$$P(d \mid c_j;\theta) = \prod_{i=1}^{|V|}(B_i P(w_i \mid c_j;\theta) + (1 - B_i)(1 - P(w_i \mid c_j;\theta))) \tag{7}$$

Here, $P(w_i c_j;\theta)$ can be estimated as the fraction of training documents in c_j that contain w_i($D=\{d_1, ..., d_m\}$ is the set of training emails):

$$P(w_i \mid c_j;\theta) = \prod_{i=1}^{|V|} \frac{\sum_{K=1}^{|D|} B_{ik} P(c_j \mid d_k)}{\sum_{k=1}^{|D|} P(c_j \mid d_k)} \tag{8}$$

And $P(c_j \mid \theta)$ are estimated as the fraction of training documents in c_j:

$$P(c_j \mid \theta) = \frac{\sum_{K=1}^{|D|} P(c_j \mid d_k)}{|D|} \tag{9}$$

NOTE: In order to get cost-sensitive NB, we think d is a spam if and only if $P(c_1 \mid d;\theta) / P(c_1 \mid d;\theta) + P(c_0 \mid d;\theta) > 0.99$, or else d is a legitimate mail.

3.2 DT and AdaBoost

Decision tree is a widely used data mining technique for classification and prediction. C4.5, a typical and effective method of building decision trees, was used in our work to build a Email filter. It's used with the default parameter settings and with rule post-pruning turned on. **NOTE:** DT algorithm is not easy to handle a large number of textual features. A straightforward way is to limit the number of textual features that are considered by the filter when a tree is built.

The purpose of boosting is to find a highly accurate classification rule by combining many weak hypotheses (assume it's h_t, 1<=t<=T), each of which may be only moderately accurate. It's assumed the existence of a separate procedure called the"Weak Learner" for acquiring the weak hypotheses. The boosting algorithm finds a set of weak hypotheses by calling the "Weak Learner" repeatedly in a series of T rounds. These weak hypotheses are then linearly combined into a single rule called the combined hypothesis (assume it's H). AdaBoost takes as input a training set $(x_1,c_1),...,(x_n,c_n)$ where $x_n \in X$, and $c_i \in C=\{c_0=$"legitimate",$c_1=$"spam"$\}$, its output is:

$$H(x) = sign(\sum_{t=1}^{T} a_t(h_t(x)) \quad (10)$$

H is a weight majority vote of the T weak hypotheses where α_t is the weight assigned to h_t, [5] discussed many methods for choosing α_t. **NOTE:** The "Weaker Learner" in our implementation is DT algorithm. The maximum depth of the DT is limited to one ten of the total number of frequent words.

3.3 ANN and SVM

This paper implements the back propagation ANN algorithm [7].The following lists main implemented details:

- Network has 3 layers: 1) Input layer: node number is ||V||+1 (V is feature subset); 2) Hidden layer: node number is half of input nodes; 3)Output Layer: only one node.
- We use sigmoid function as the activation function both in hidden layer and output layer.

Support vector machines are well founded from Statistical Learning Theory. They minimize the structural risk, instead of empirical risk as in most learning methods. SVM makes no assumptions of the distribution of the data. Support vector machines have been proven to be a very powerful classification method for high dimensional data set, e.g. in hand-written digits recognition, face recognition, object recognition and so on. In its simplest linearly separable case, a support vector machine is a hyperplane that separates a set of positive examples and negative examples correctly. Support vector machine can be generalized to linearly no

separable case via introducing the concept of "soft-margin"; and constructs nonlinear classifiers via kernel functions. Kernel functions map the training examples into a higher dimensional feature space and SVM constructs a optimal separating hyper plane in the higher dimensional feature space that is corresponding to a nonlinear classifier in the input space. Examples of kernel functions are polynomials, Gaussian, and Radial Basis Functions. In experiments of this paper, we choose polynomials function.

In our application, we would like to penalize errors on legitimate examples stronger than errors on spam examples. This can be achieved using cost factors c_{10} and c_{01} to adjust the cost of false legitimate mails vs. false spam mails. Such cost factors can be directly incorporated into the SVM [3]. **NOTE:** In our experiments, we adopt $c_{10} = 1.5$ and $c_{01} = 0.5$.

3.4 KNN and VSM

Let χ be a set. We define:

DEFINE 0. For every email $x \in \chi$, a vector representation $\vec{x} = <w_1', w_2', \ldots w_{\|V\|}'>$, where w_i' is the value of feature f_i, it's computed as follows (tf_i is the number of times feature f_i occurs in x, N is the total number of emails in corpus, N_i is total number of emails f_i appears in):

$$w_i' = tf_i * \log(N / N_i + \alpha) \tag{11}$$

We normalize $\vec{x} = <w_1', w_2', \ldots w_{\|V\|}'>$ as follow:

$$< w_1, \ldots, w_{\|V\|} > = < w_1', \ldots, w_{\|V\|}' > / (\max(w_1', \ldots, w_{\|V\|}') * \sum_{j=1}^{\|V\|} w_j'^2) \tag{12}$$

For $x, y \in \chi$, assume $\vec{x} = <w_{x1}', w_{x2}', \ldots w_{x\|V\|}'>$ and $\vec{y} = <w_{y1}, w_{y2}, \ldots, w_{y\|V\|}>$, distance between x and y is:

$$D(x, y) = \sum_{i=1}^{\|V\|} (w_{xi} * w_{yi}) / \sqrt{\sum_{i=1}^{\|V\|} w_{xi}^2 \sum_{i=1}^{\|V\|} w_{yi}^2} \tag{13}$$

DEFINE 2. $D(\bullet,\bullet) : \chi \times \chi -> IR$ is a distance if, $\forall X, Z \in \chi$: $D(X,Z) \geq 0$, $D(X, Z) = D(Z,X)$ and $D(X,Z) \leq D(X,V) + D(V,Z)$.

DEFINE 3. A metric space is a pair $(\chi, D(\bullet,\bullet))$, where χ is a distance on χ.

Let $\Gamma = (x_1, c_1), \ldots, (x_n, c_n)$ be a training set, $x_i \in X$, $X \in$ separable metric space χ, $c_i \in C = \{c_0 =$"spam", $c_1 =$"legitimate"$\}$. Assume x is a sample to be labeled, let $(x_1', c_1'), \ldots, (x_n', c_n'))$ be the training samples arranged in increasing order of distance from X, we assign class c_j' to x, where

$$j' = \arg\max_j \sum_{i=1}^{k} 1_{c_i'=j'} \quad (14)$$

NOTE: In order to get cost-sensitive KNN, we can do that: In the k nearest neighbors of x, there are a spam mails and b legitimate mails, if and only if a/(a+b)> λ (λ = 0.99), we think x is a spam.

The VSM algorithm is very simple as well as effective. Let $X_0 = \{ \vec{x}_{01},..., \vec{x}_{0m} \}$ be vector set of the legitimate training data, $X_1 = \{ \vec{x}_{10},..., \vec{x}_{1m} \}$ be vector set of the spam training data. We can get centric vector $\vec{c}_{x_0}$ of X_0:

$$\vec{c}_{x_0} = \sum_{i=1}^{m} \vec{x}_{0i} / m \quad (15)$$

Similarly, we can get centric vector $\vec{c}_{x_1}$ of X_1. Let $\vec{x} = \{w_1,..., w_{\|V\|}\}$ is a sample to be classified, we compute $D(\vec{x}, \vec{c}_{x_1})$ and $D(\vec{x}, \vec{c}_{x_1})$, if $D(\vec{x}, \vec{c}_{x_1})$ > $D(\vec{x}, \vec{c}_{x_1})$, we think $\vec{x}$ is "spam", else $\vec{x}$ is "legitimate". **NOTE:** In fact, in order to get cost-sensitive VSM algorithm, we can classify $\vec{x}$ into "spam" if and only if $D(\vec{x}, \vec{c}_{x_1})$ - $D(\vec{x}, \vec{c}_{x_1}) > \lambda$. However, it's difficult to specify the value of λ, so we classify $\vec{x}$ into "spam" if and only if $D(\vec{x}, \vec{c}_{x_1})$ / ($D(\vec{x}, \vec{c}_{x_1})$ + $D(\vec{x}, \vec{c}_{x_1})$) > λ ($\lambda = 0.6$)

*Table 3. Effect of varying the size of feature subset. Results1 for PU1 and Ling-Spam. *: Out of memory. @: Training time is too long (above 1 hours).x ratio of feature selection, y: size of feature subset. In each cell, a:b:c is result of a filter h, a is error count for loeitimate, b is error count for spam, c is value of ER(h).*

PU1					
Filter x(y)	0.05%(10)	0.25%(52)	0.5%(105)	1%(211)	2%(423)
NB	1:46:0.131	1:40:0.11	1:36:0.106	2:27:0.098	1:34:0.102
DT	83:2:0.496	33:13:0.33	14:9:0.170	11:12:0.15	*
AdaBoost	85:1:0.495	21:19:0.27	15:18:0.21	17:12:0.21	*
ANN	10:20:0.17	12:16:0.18	4:13:0.082	5:6:0.07	3:9:0.06
SVM	5:28:0.14	1:37:0.109	0:38:0.10	0:39:0.10	0:50:0.121
KNN(K=120	180:0:1	180:0:1	180:0:1	4:40:0.162	0:90:0.2
VSM	24:9:0.252	12:10:0.16	8:20:0.150	6:18:0.121	2:27:0.098
Ling-Spam					
Filter x(y)	0.05%(10)	0.25%(52)	0.5%(105)	1%(211)	2%(423)
NB	0:49:0.170	0:50:0.174	0:52:0.179	0:53:0.182	0:54:0.184
DT	22:23:0.35	2:15:0.087	*	*	*
AdaBoost	28:10:0.32	3:25:0.140	*	*	*
ANN	1:28:0.121	0:7:0.029	@	@	@
SVM	0:45:0.159	0:46:0.162	0:53:0.182	0:52:0.17	0:52:0.179
KNN(K=120	119:0:1	18:10:0.24	7:10:0.127	0:84:0.261	0:81:0.254
VSM	10:7:0.145	4:9:0.087	1:13:0.066	0:18:0.07	0:15:0.059

4. EXPERIMENTAL RESULTS

Table 3 shows the effect of varying the size of feature subset. We do experiments on PU1 and Ling-Spam dataset respectively. For PU1, we use 346 mails of spam and 430 examples of legitimate for training; testing mails of spam contain 134, non-spam testing mails contain 180. For Ling-Spam, count of training mails of spam is 357, count of legitimate mails for training is 2293; testing mails of spam and legitimate is 124 and 119.

Table 4. Effect of varying the count of training examples. Results for PU1 and Ling-Spam.: Out of memory. @: Training time is too long (above 1 hour). x0: training emails count of spam, y0: training emails count of legitimate, x1: testing emails count of spam, y1: testing emails count of legitimate, z: size of feature subset. In each cell, a:b:c is result of a filter h, a is error count for legitimate, b is error count for spam, c is value of ER(h). x0:y0:x1:z1(z) in each case is as follows: case 1: 10:430:134:180 (146) ,case 2: 100:430:134:180 (168); case 3: 200:430:134:180 (190); case 4: 346:430:134:180 (211); case 5: 346:10:134:180 (119); case 6:346:100:134:180 (144); case 7: 20:2293:124:119 (531); case 8: 100:2293:124:119 (542); case 9: 357:2293:124:119 (283); case 10: 357:357:124:119 (219); case 11: 357:10:124:119 (96); case 12: 357:100:124:119 (138)*

	PU1					
Filter	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
NB	0:106:0.23	0:47:0.115	0:36:0.091	2:27:0.098	101:0:0.53	6:23:0.137
DT	2:90:0.266	5:35:0.161	13:28:0.23	11:12:0.15	134:2:0.63	32:7:0.297
AdaBoost	0:90:0.2	5:35:0.161	10:21:0.18	17:12:0.21	100:0:0.53	30:5:0.275
ANN	0:134:1	1:28:0.086	3:19:0.08	5:6:0.072	176:0:0.66	23:4:0.222
SVM	0:134:1	0:83:0.187	0:57:0.13	0:39:0.098	176:0:0.66	10:14:0.15
KNN	26:91:0.70	0:128:0.26	0:128:0.26	4:40:0.162	180:0:1	8:22:0.157
	K=10	K=60	K=120	K=120	K=10	K=120
VSM	14:23:0.22	6:20:0.128	5:20:0.116	6:18:0.121	7:14:0.119	9:8:0.120
	Ling-Spam					
Filter	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12
NB	9:8:0.120	0:54:0.185	0:53:0.182	0:39:0.141	11:26:0.25	0:42:0.15
DT	*	*	*	2:12:0.075	95:2:0.579	9:80.139
AdaBoost	*	*	*	3:5:0.058	79:3:0.531	14:6:0.179
ANN	@	@	@	1:2:0.021	119:0:1	3:2:0.044
SVM	0:123:0.34	0:116:0.33	0:52:0.179	0:21:0.08	119:0:1	1:8:0.046
KNN	0:123:0.34	0:123:0.34	7:10:0.127	0:124:1	34:2:0.306	2:9:0.062
	K=10	K=60	K=1	K=120	K=06	K=60
VSM	1:92:0.325	0:44:0.156	1:13:0.066	0:0:0	6:8:0.106	3:7:0.066

From table 3, we can see that: (1) Most filters's performance increase with an increasing size of feature subset; (2) When increase feature subset's size, cost-sensitive filters (SVM, NB, KNN, VSM) decrease the count of misclassifying legitimate (but increase the count of misclassifying spam sometimes), while DT, AdaBoost and ANN didn't have this characteristic; (3)DT, AdaBoost is not fit to

large feature subset; (4)Although ANN has good performance, it's not available for email filtering because of its long training time. (5)Design of SLER(h), LSER(h) and ER(h) is reasonable, it reflect a filter's performance well and truly.

Table 4 shows the effect of varying the count of training examples. For PU1 and Ling-Spam dataset, we all select 1% features from original feature space. These results show us: (1)Filter h always errs in class _ with fewer training examples, because h know little information about $\mathcal{E}$; (2)Most filters are very sensitive to unbalance dataset, while VSM is not so depend on dataset's distribution.

In addition, we investigated training time and filtering time of all filters. According to our all experiments, we sort these filters un-strictly: (1)Training time: {KNN} $\prec$ {NB, VSM} $\prec$ {SVM} $\prec$ {DT,AdaBoost}_{ANN}; (2)Testing time: {NB, SVM, VSM, ANN} $\prec$ {DT,AdaBoost} $\prec$ {KNN}.

5. CONCLUSION

Experiments suggest that our cost-sensitive evaluation method is good. The comparative results show that cost-sensitive filters we implemented such as NB, SVM, VSM and KNN have fewer count of misclassifying legitimate when relative parameters, feature subset size and training dataset's distribution are reasonable. It's noticeable that a filter with a low error rate on legitimate has high error rate on spam, so we will design a new method with low SLER(h) as well as low LSER(h) in the future.

ACKNOWLEDGEMENTS

(Partially) supported by the NSFC major research program: "Basic Theory and Core Techniques of Non-Canonical Knowledge" (60496322) .

REFERENCES

[1] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In M. Sahami, editor, Learning for Text Categorization: Proceedings of the 1998 AAAI/ICML Workshop, Madison, WI, 1998. AAAI Press, (1998) 41-48

[2] Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Spyropoulos, C. D. An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages. Nicholas J. Belkin, Peter Ingwersen, Mun-Kew Leong.In Proc. of the 23 rd Annual International ACM SIGR Conference on Research and Development in Information Retrieval, Athens, Greece, 2000.ACM, (2000) 160-167

[3] Harris Drucker, Vladimir N. Vapnik, IEEE TRANSACTION ON NETWORK, VOL. 10.

NO. 5, SEPTEMBER 1999, Support Vector Machines for Spam Categorization
[4] Jason D. M. Rennie. ifile: An Application of Machine Learning to E-Mail Filtering. M. Grobelnik, D. Mladeni c, and N. Milic-Frayling.In Proc. KDD-2000 Text Mining Workshop, Boston, MA, USA,2000. University of Alberta,2000.
[5] Robert E. Schapire. Drifting games. Machine Learning. (June 2001) 43(3):265-291
[6] Thorsten Joachims. Learning to Classify Text Using Support Vector Machines.Kluwer Academic Publishers, (2001) 21-22
[7] T. M. Mitchell. Machine Learning. McGraw-Hill, 1997
[8] Wenbin Li, Ning zhong and Chunian Liu. Design and Implementation of an Email Classifier. 2nd International Conference on Active Media Technology, May 29-31, 2003. ACTIVE MEDAIA TECHNOLOGY, World Science, (2003) 423-430
[9] Y. Diao, H. Lu, and D. Wu. A Comparative Study of Classification Based Personal Email Filtering. Takao Terano, Huan Liu, Arbee L. P. Chen.In Proc. PAKDD-2000, Kyoto,Japan,2000.Springer, (2000) 408-419
[10] Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In Proc. 14th International Conference on Machine Learning (ICML-97), 412C420
[11] Zhong, N., Matsunaga, T., Liu, C. A Text Mining Agents Based Architecture for Personal E-mail Filtering and Management, Proc. Third International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2002), LNCS 2412, Springer, 337-346.