Text Filtering by Boosting Naive Bayes Classifiers

Yu-Hwan Kim Shang-Yoon Hahn Byoung-Tak Zhang

Artificial Intelligence Lab (SCAI)
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
{yhkim, syhan, btzhang}@scai.snu.ac.kr

Abstract

Several machine learning algorithms have recently been used for text categorization and filtering. In particular, boosting methods such as AdaBoost have shown good performance applied to real text data. However, most of existing boosting algorithms are based on classifiers that use binary-valued features. Thus, they do not fully make use of the weight information provided by standard term weighting methods. In this paper, we present a boosting-based learning method for text filtering that uses naive Bayes classifiers as a weak learner. The use of naive Bayes allows the boosting algorithm to utilize term frequency information while maintaining probabilistically accurate confidence ratio. Applied to TREC-7 and TREC-8 filtering track documents, the proposed method obtained a significant improvement in LF1, LF2, F1 and F3 measures compared to the best results submitted by other TREC entries.

1 Introduction

Information filtering involves the delivery of information to people who need it [1]. With the explosion of electronic documents, information filtering has become one of the most important application areas in the information retrieval community. TREC conferences also introduced the filtering track and attracted many researchers around the world [18].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGIR 2000 7/00 Athens, Greece
© 2000 ACM 1-58113-226-3/00/0007...$5.00

An information filtering system learns the interest of users in terms of a profile which are applied to an incoming stream of new documents. Usually, it is assumed that once the user profile is constructed, it does not change any more. In addition, documents are not ranked when they are sent to users. This is exactly the same setting as assumed in the TREC batch filtering track [6].

Information filtering is closely related to text classification algorithms. To improve the performance of filtering systems, we need to improve classification algorithms. One of the most successful methods for text classification is the AdaBoost algorithm and its variants [5] [13]. In AdaBoost, each weak learner (i.e., a classifier performing just slightly better than random guessing) iteratively learns documents re-weighted by the previous weak learner. The combined classifier is composed by weighted voting of weak learners. Several researchers have studied and evaluated AdaBoost in text classification [4] [12] [14]. Most commonly used weak learners include decision stumps and C4.5. But most of them do not pay much attention to how to fully utilize the information obtained from the document representation. In other words, since these methods classify documents according to the existence or absence of words, they do not fully utilize other information, such as term weights, obtained from preprocessing documents.

In this paper, we propose to use naive Bayes classifiers as a weak learner for boosting, and show that it can be successfully used with AdaBoost in a text filtering context. One of the main advantages of this method, called BayesBoost, is that naive Bayes utilizes term weights such as term frequency naturally. Moreover, this is a probabilistic model, and thus provides a natural measure for calculating confidence ratios. This method has been applied to TREC-7 and TREC-8 filtering track datasets.

Compared to the best results of the TREC entries of the last two years, the BayesBoost filtering system achieved significant improvement in the LF1, LF2, F1, and F3 measures. We also investigate the reasons for performance improvement by further experimental analysis.

The paper is organized as follows. In Section 2, we briefly discuss related work on using machine learning techniques for text filtering. Section 3 describes the AdaBoost and naive Bayes learners and introduce the BayesBoost algorithm. Section 4, we report experimental results on TREC-7 and TREC-8 filtering datasets. Finally, Section 5 concludes this study.

2 Related Work

Information filtering has been studied from two different perspectives. One is from the machine learning point of view, and the other from the information retrieval aspect. Most research in the information retrieval community is closely related to the Rocchio algorithm. This method can be formulated as

$$Q' = \alpha Q + \beta \left(\frac{1}{R} \sum_{D_i \in \text{Rel}} D_i \right) - \gamma \left(\frac{1}{N} \sum_{D_i \notin \text{Rel}} D_i \right),$$

where Q is the old query vector, Q' is the new query vector, N is the number of relevant documents, and R is the number of non-relevant documents. Based on this algorithm, several researchers proposed and evaluated more powerful and robust algorithms. These include dynamic feedback optimization [3], query zoning [16], and pivoted document length normalization [17].

In the machine learning community, various methods have been used for document filtering. These include neural networks, k-nearest neighbor methods, support vector machines, and naive Bayes classifiers. Yang [19] compares these methods and others using various statistical tools.

Recently, there has been great interest in learning algorithms that achieve high accuracy by combining the predictions of several classifiers. They include committee machines, boosting, and bagging [2]. Several researchers have reported significant improvement using voting methods [4] [12]. Schapire et al. have compared AdaBoost with Rocchio using REUTERS-21578, TREC-3, and AP and concluded that both are competitive and are quite effective [15], except TREC-3 experiments where AdaBoost showed worse performance than Rocchio due to lack of data. MultiBoost is an algorithm in which AdaBoost and wagging is combined. Here, wagging is a variant of bagging; bagging uses resampling to get the datasets for training and producing a weak hypothesis, whereas wagging uses reweighting for each training example, pursuing the effect of bagging in a different way.

Lewis et al. [9] proposed to use two machine learning algorithms, the Widrow-Hoff and exponentiated-gradient (EG) algorithms. Both are linear classifiers and Lewis et al. showed that they are more effective than the Rocchio algorithm empirically.

Naive Bayes is a well-known probabilistic algorithm. It has been used with many algorithms which need probabilistic interpretations. For instance, McCallum et al. [11] proposed a new algorithm that is based on naive Bayes and used expectation-maximization (EM) to train the labeled and unlabeled documents regarding the label of the unlabeled documents to be a hidden variable to be determined in each EM step. They achieved good performance in REUTERS-21578 documents.

3 Filtering by BayesBoost

3.1 AdaBoost

AdaBoost is a learning algorithm that iteratively generates classifiers and then combines them to build the ultimate classifier. It was first proposed by Shapire et al. and studied and evaluated by several researchers. Shapire also shows that it is well fit to the problem of text classification as well [14] [15]. However, it is restricted that each hypothesis or weak learner can produce an output -1 or 1.

A more sophisticated version of the AdaBoost algorithm is AdaBoost with confidence ratio [13]. It interprets the sign of the weak learner $\theta_t(x)$ as the predicted label (-1 or 1) to be assigned to instance x, and the magnitude $|\theta_t(x)|$ as the "confidence" in this prediction. Thus if $\theta_t(x)$ is close to or far from zero, it is interpreted as a low or high confidence prediction. In this paper, the range of θ_t is [-1,+1].

AdaBoost builds a classifier by combining the hypotheses:

$$f(x) = \sum_{t=1}^{T} \alpha_t \theta_t(x),$$

where T is the number of iterations and α_t is the weight for hypothesis θ_t. If we restrict the range of $\theta_t(x)$ to [-1,1], we can calculate α_t by reusing the equation of the original AdaBoost.

Several researchers have used weak learners such as C4.5 (decision tree learner) [4] [12] and decision stumps [15]. All of them showed good performance in text classification. However, as mentioned previously,

these weak learners do not take into account weight information and basically classify documents according to the existence or absence of words. Therefore, it has room for further improvement. Another problem is that these weak learners are far from probabilistic, thus is not suitable for calculating confidence ratios. In order to solve these two problems, we use naive Bayes classifiers as a weak learner.

3.2 Naive Bayes

Naive Bayes is a well-known classical method that has been widely used in text categorization [7] [11]. The learning task for a naive Bayes classifier is to use a set of training documents to estimate parameters, then use the estimated model to classify new documents. Document d_i is composed of a sequence of words, which is $w_{i1}, w_{i2}, \cdots, w_{i|d_i|}$. We assume that the words in a document is mutually independent and the probability of a word is independent of its position within the document, which is not true in real situations. In spite of this disparity between the reality and the model, naive Bayes classifiers showed rather good performance in text classification [7] [11].

Due to the independence assumption the probability that a document d_i is generated from the class c_j can be expressed as

$$P(d_i|c_j;\theta) = P(|d_i|)\prod_{k=1}^{|d_i|} P(w_{d_{ik}}|c_j;\theta)^{N(w_{d_{ik}},d_i)},$$

where $w_{d_{ik}}$ denotes the k-th word in document d_i, $N(w_{d_{ik}}, d_i)$ denotes the weight of word $w_{d_{ik}}$ occurring in document d_i, and $|d_i|$ is the number of words in the document. The parameter of each class is $\theta_{w_k|c_j} = P(w_k|c_j;\theta)$, where $\sum_{k=1}^{|V|} P(w_k|c_j;\theta) = 1$.

$$\hat{\theta}_{w_k|c_j} = \frac{1+\sum_{i=1}^{|D|} N(w_k,d_i)P(c_j|d_i)}{|V|+\sum_{k=1}^{|V|}\sum_{i=1}^{|D|} N(w_k,d_i)P(c_j|d_i)} \quad (1)$$

The class prior probability is given as $\theta_{c_j} = P(c_j|\theta)$. This can be estimated as

$$\hat{\theta}_{c_j} = \frac{\sum_{i=1}^{|D|} P(c_j|d_i)}{|D|}, \quad (2)$$

where $|D|$ is the total number of documents. With these parameters, assuming $P(|d_i|)$ is uniform, we can compute the probability that a particular class generated a given document:

$$P(c_j|d_i;\hat{\theta}) = \frac{P(c_j|\hat{\theta})P(d_i|c_j;\hat{\theta})}{P(d_i|\hat{\theta})} \quad (3)$$

$$= \frac{P(c_j|\hat{\theta})\prod_{k=1}^{|d_i|} P(w_{d_{ik}}|c_j;\hat{\theta})^{N(w_{d_{ik}},d_i)}}{\sum_{r=1}^{|C|} P(c_r|\hat{\theta})\prod_{k=1}^{d_i} P(w_{d_{ik}}|c_r;\hat{\theta})^{N(w_{d_{ik}},d_i)}}$$

We must choose $\arg\max_j P(c_j|d_i;\theta)$ as the best class of the document.

Because naive Bayes is a probabilistic model, it will not achieve good performance if we do not have enough positive examples. In our experiments, we obtained poor results when we applied only naive Bayes classifiers to the TREC-8 filtering datasets. Naive Bayes also chooses the probability that is better of the two probabilities, without exploiting the difference of the two probabilities further. As we argued before, if we use this feature to calculate the confidence ratio, a better performance can be achieved.

3.3 BayesBoost: AdaBoost with Naive Bayes

Naive Bayes is a probabilistic model, and thus it is well fit to calculating confidence ratio. It also takes into account weight information and thus helpful to more accurate classification than using only decision stumps. This consideration leads to the BayesBoost algorithm as summerized in Figure 1.

Let $P(c_i = 1|d_i;\hat{\theta})$ denote the probability of document d_i being evaluated as relevant and $P(c_i = -1|d_i;\hat{\theta})$ as irrelevant. We can compute $h_t(d_i;\hat{\theta})$ by

$$h_t(d_i;\hat{\theta}) = P(c_i = 1|d_i;\hat{\theta}) - P(c_i = -1|d_i;\hat{\theta}) \quad (4)$$

which means the difference between the two probabilities of the two classes and have a value with range [-1,1]. In order to reflect the effect of reweighting we resampled training data from the weighted distribution by roulette wheel selection, which is frequently used in genetic algorithms. Roulette wheel selection is a method to choose data proportional to the density (in this context, D_i).

We can improve the performance of BayesBoost even more by choosing better starting points. It has the effect of giving more weights to the documents which can gain more score than other documents [15]. To do this, we initialize the distribution of the documents as follows:

$$D_1(i) = \begin{cases} \frac{a-c}{Z_0} & \text{if } y_i = +1 \\ \frac{d-b}{Z_0} & \text{if } y_i = -1, \end{cases} \quad (5)$$

where Z_0 is a normalization factor and a, b, c, and d are defined in Equation (6) below.

Figure 1: The BayesBoost algorithm.

1. Given: $(d_1, c_1), ..., (d_m, c_m)$, where $d_i \in X, c_i \in \{-1, +1\}$.
2. Initialize a distribution using Equation (5).
3. For $t = 1, \cdots, T$:
   - Select new training documents using roulette wheel selection.
   - Train a naive Bayes classifier and get a weak hypothesis h_t by Equation (4), where $\hat{\theta} = (\hat{\theta}_{w_k|c_j}, \hat{\theta}_{c_j})$ is computed by Equations (1) and (2)
   - Choose $\alpha_t = \frac{1}{2}\log(\frac{1+r}{1-r})$, where
   $$r = \sum_{i=1}^{|D|} D_t(i) c_i h_t(d_i; \theta).$$
   - Update the distribution:
   $$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t c_i h_t(d_i; \hat{\theta}))}{Z_t},$$
   where Z_t is a normalization factor.
4. Output the final hypothesis:
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(d_i; \hat{\theta})\right).$$

4 Experiments and Results

4.1 Performance Measures

Past studies on text filtering have used various measures to evaluate system performance. One of these is the break-even point [10]. But as discussed by Singhal et al., break-even points are not very suitable for measuring the performance of filtering algorithms [15]. Recently, TREC conferences have been moving toward the use of *utility* as the measure of choice for evaluating text filtering [6]. Thus, in this study we use utility, for example LF1, instead of the traditional break-even point.

Let R_+ be the number of relevant and retrieved documents, N_+ be the number of non-relevant but retrieved documents, R_- be the number of relevant but non-retrieved documents, and N_- be the number of non-relevant and non-retrieved documents. Then, linear utility is defined as follows:

$$\text{Linear utility} = aR_+ + bN_+ + cR_- + dN_- \quad (6)$$

where a, b, c, and d are constant coefficients.

In TREC-8, the LF1 and LF2 measures are used:

$$\text{LF1} = \text{F1} = 3R_+ - 2N_+ \quad (7)$$
$$\text{LF2} = 3R_+ - N_+. \quad (8)$$

In TREC-7, F1 and F3 are used. F1 is the same as LF1, and F3 is defined as

$$\text{F3} = 4R_+ - N_+. \quad (9)$$

However, it is meaningless to average through the runs, because a few topics will have dominant effects on the results. Hence, we also used the scaled linear utility to show the whole performance effectively [6].

$$\text{Scaled linear utility} = \frac{\max\{u(S,T), U(s)\} - U(s)}{MaxU(T) - U(s)},$$

where $u(S,T)$ is the linear utility of system S for topic T, $MaxU(T)$ is the maximum possible utility score for topic T, $U(s)$ is the utility of retrieving s non-relevant documents.

4.2 Datasets: TREC-7 & TREC-8

The datasets for the TREC-7 filtering track consist of the articles from AP News for the years 1988-1990, Topics 1-50. The 1988 AP documents were used as the training set, and the documents for 1989-1990 as the test set. For TREC-8 datasets, which consist of the FT articles from 1992-1994 and Topics 351-400, we used the 1992 FT documents as a training set and the documents for 1993-1994 as a test set. This is exactly the same setting as the TREC-7 and TREC-8 batch filtering tracks suggested.

We did not use any information from the topic, such as 'description.' We stemmed the documents using the Porter's algorithm, and removed words from the stop-list and common short words. Finally, we removed documents which had fewer than 40 terms. Standard $tf \cdot idf$ is used for term weighting. No thesaurus or other datasets were used. Though our term indexing method generated more than 160,000 terms, we extracted 5251 terms for TREC-7 and 4071 for TREC-8 according to the increasing document frequency in AP88 and FT92.

Three runs were made for each topic and the results are reported in average performance. In running the boosting algorithms, we continued training until 15 hypotheses are generated or the training error goes to zero. For comparison purposes, we also

Figure 2: Comparison of BayesBoost and the three best runs that were submitted to TREC-7. Two different performance measures were used: scaled F1 (first row) and scaled F3 (second row). A positive value for the topic number means BayesBoost outperforms the corresponding entry to the TREC-7 Filtering track.

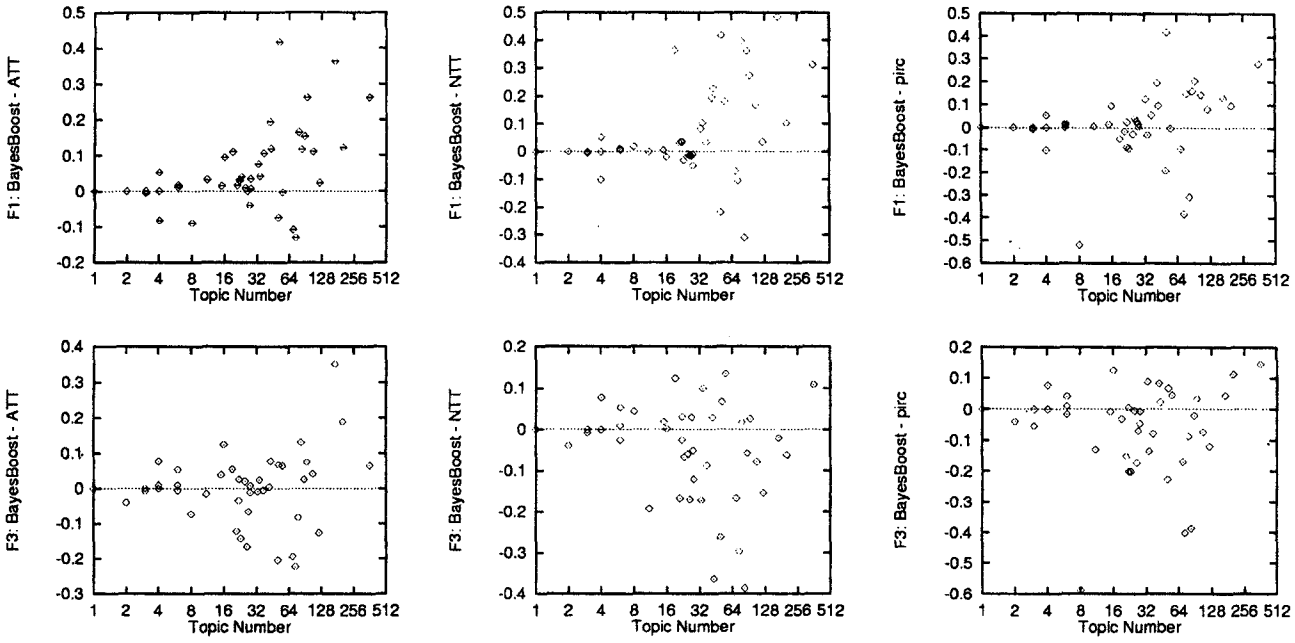


Figure 3: Comparison of BayesBoost and the three best runs that was submitted to TREC-8. Two different performance measures were used: scaled LF1 (first row) and scaled LF2 (second row). A positive value for the topic number means BayesBoost outperforms the corresponding entry to the TREC-8 Filtering track.

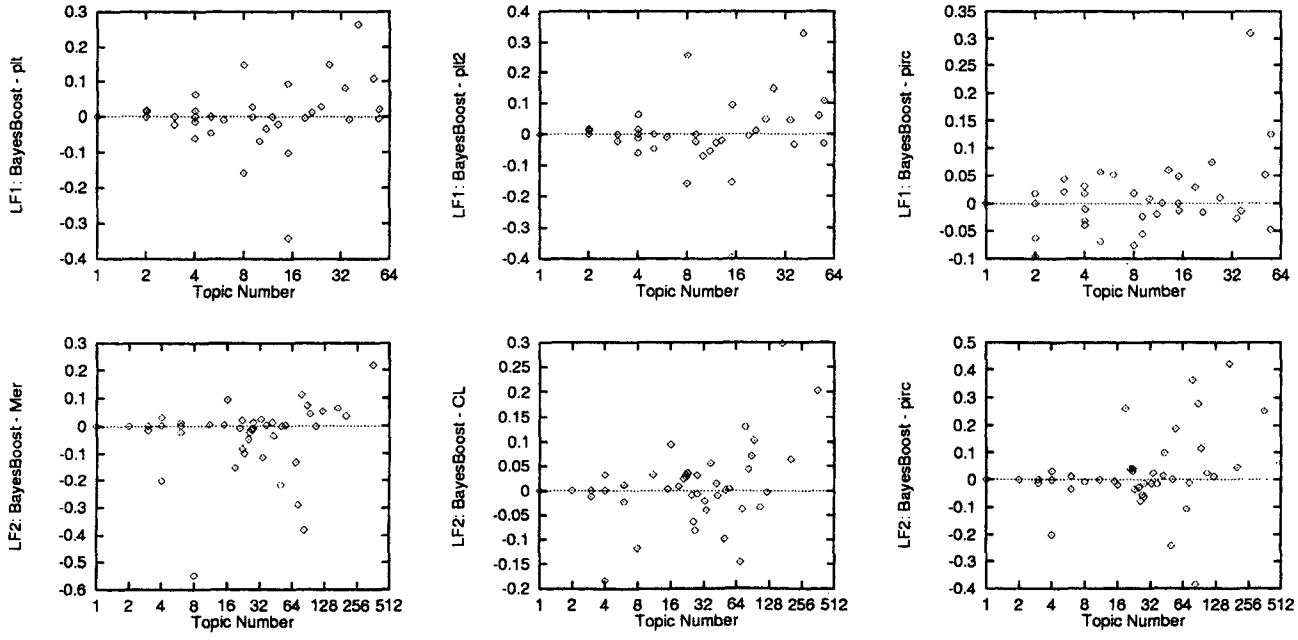


Table 1: Detailed results of BayesBoost (SCAI) on TREC-7 and TREC-8 datasets. Also shown are the results of the three best teams each year. These data are excerpted from TREC-7 and TREC-8. (The 'asl' at the bottom line stands for the averaged scaled linear utility when $U(s)$ is set to 100. and The values in the parentheses in 'SCAI' are the scores when we also take the unjudged documents into account.

	LF1				LF2			
	SCAI	ATT	NTT	pirc	SCAI	ATT	NTT	pirc
1	214	87	-91	101	311	386	288	391
2	77	-8	5	11	131	-31	193	37
3	93	136	220	204	139	316	359	333
4	9	-1	14	4	16	24	63	36
5	47	47	50	38	73	79	80	81
6	117	193	166	183	207	385	360	364
7	101	25	-14	3	166	130	235	231
8	0	-2	2	6	2	1	21	7
9	5	0	3	0	11	0	8	19
10	164	89	363	361	255	152	576	576
11	153	130	117	72	252	421	457	413
12	605	86	-90	419	871	213	910	788
13	6	36	0	180	18	48	0	260
14	23	-11	12	6	45	47	79	76
15	-11	-38	-143	1	-3	-19	-39	6
16	4	1	26	2	15	11	43	18
17	188	110	5	108	268	247	300	276
18	-1	0	-56	0	1	-21	-47	-14
19	69	0	83	0	103	4	119	3
20	35	17	-10	49	59	46	5	133
21	4	2	0	6	3	15	21	19
22	1475	786	655	745	2022	1807	1655	1528
23	131	207	192	353	218	391	446	523
24	253	78	70	117	360	311	363	346
25	4	-5	11	25	9	42	22	58
26	3	0	0	3	4	0	0	8
27	0	0	0	0	0	0	0	0
28	3	0	0	0	4	0	0	4
29	-11	-2	0	0	-14	-1	0	0
30	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0
37	0	0	0	-4	0	0	0	-10
38	111	50	44	13	152	196	363	95
39	0	0	0	-4	0	0	0	-7
40	73	3	-61	17	109	33	363	74
41	15	0	0	0	28	0	0	0
42	-31	-116	-139	-129	-17	-33	-44	-68
43	-2	0	0	1	-2	0	0	19
44	56	75	63	43	107	146	89	148
45	-57	-175	-167	-178	-93	-143	-123	-136
46	7	0	-2	28	8	18	-1	67
47	12	3	2	2	16	7	25	14
48	-1	-4	-2	-4	-6	-5	-1	-8
49	9	2	9	8	17	21	67	51
50	0	0	0	0	0	0	0	0
asl	0.512 (0.484)	0.461	0.452	0.500	0.462 (0.435)	0.460	0.505	0.509

	LF1				LF2			
	SCAI	plt1	plt2	pirc	SCAI	CL	pirc	Mer
351	17	22	25	25	20	24	28	-136
352	96	82	115	125	119	56	238	185
353	13	32	42	28	15	5	44	41
354	-6	-9	-9	-7	-3	-1	6	0
355	0	0	0	0	0	0	0	0
356	-16	7	7	4	-8	-1	6	-6
357	55	47	41	59	52	50	69	56
358	0	0	0	0	0	0	0	0
359	-2	-5	-5	0	-3	0	-1	0
360	0	0	0	0	0	0	3	3
361	0	0	0	0	0	-3	0	0
362	0	0	0	0	0	2	0	0
363	0	0	0	-4	0	-1	-2	-2
364	0	0	0	0	0	0	0	-1
365	0	-16	-28	4	0	-1	5	0
366	-2	0	0	3	-2	6	6	23
367	0	0	5	4	0	0	13	9
368	0	0	0	0	0	0	0	0
369	0	0	0	0	0	0	0	0
370	-2	-20	-20	-13	-2	-1	-2	-9
371	0	0	0	0	0	0	0	0
372	0	3	3	3	3	-3	8	0
373	0	0	0	0	0	14	0	0
374	4	5	5	-19	6	-3	6	42
375	-2	-2	1	3	3	2	6	5
376	0	3	3	0	0	-3	0	0
377	0	6	6	2	0	11	6	3
378	0	-25	-14	-12	4	-6	-12	-19
379	0	0	0	0	0	-4	0	0
380	0	0	0	0	0	0	0	0
381	0	0	0	0	0	-3	0	0
382	0	-1	-2	-2	0	8	2	0
383	-6	97	113	72	-2	0	103	59
384	0	0	0	0	0	0	0	0
385	14	9	14	28	16	26	35	31
386	0	0	0	-2	0	0	-1	0
387	-2	-10	-10	-1	-1	-3	0	3
388	0	0	0	0	0	0	1	0
389	273	145	113	76	208	213	203	32
390	0	-4	-4	0	0	0	0	3
391	34	36	-18	-35	55	59	39	68
392	-3	-32	-32	2	3	2	23	15
393	0	7	7	1	0	1	5	3
394	0	0	0	0	0	-4	0	0
395	-10	-8	-1	-50	-7	-3	16	44
396	0	1	1	-1	0	-3	2	0
397	0	-2	-2	0	0	0	0	-3
398	0	9	9	0	-1	2	4	5
399	0	0	0	3	0	4	5	0
400	-1	-13	-8	-9	1	5	11	9
asl	0.715 (0.700)	0.712	0.713	0.714	0.719 (0.706)	0.721	0.734	0.720

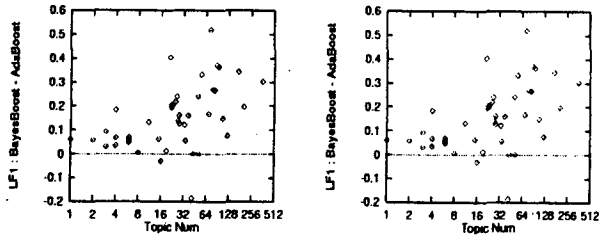
show in Figures 2 and 3 the results of the three best-ranked runs submitted to TREC-7 and TREC-8, respectively. Unfortunately, our results will have a disadvantage compared with other runs because our new runs did not enter TREC pools [18], so we ignored unjudged documents in the test documents. In this setting, we might have a slight advantage over other runs [6]. The x-axis is sorted by the number of positive documents in the training documents set. More detailed figures are shown in Table 1. These results are discussed in Section 4.3 below.

We also made experiments in the same setting using AdaBoost with decision stumps. The results are shown in Figure 4(left) for TREC-7 and Figure 4(right) for TREC-8. In general, BayesBoost significantly outperforms AdaBoost. The degree of improvement is especially big for TREC-8 datasets. It is also interesting to note that the performance of BayesBoost gets increased, compared to that of AdaBoost, as the number of positive documents for the topics increases.

4.3 Analysis

To summarize the comparative results shown in Figures 2 and 3 and Table 1, BayesBoost is competitive to other entries in almost all the runs both on TREC-7 and TREC-8 data, but it achieved moderate performance compared in the LF2 measure. This result is somewhat related to that of Schapire et al. [15]. They showed that though AdaBoost is very competitive in LF1 measure, it does not show good results when it comes to the LF2 measure compared with the Rocchio method. This might be due to the fact that boosting algorithms including BayesBoost do not optimize for the linear utility actively, except that they optimize the initial distribution for linear utility measure. However, as naive Bayes is a probabilistic model, if we calibrate the threshold given a

Figure 4: Comparison of BayesBoost and AdaBoost on TREC-7 (left) and TREC-8 (right). Positive values mean that BayesBoost performs better than AdaBoost in the scaled LF1 measure. The relative performance of BayesBoost increases as the number of positive documents for the topics increased.



probability that a document is a positive example, we could improve the results further.

In contrast to BayesBoost, 'pirc' prepares six forms of functions and uses genetic algorithms in determining various weights of their linear threshold functions to optimize for suitable utility measure [8]. Finally, 'pirc' votes with combined functions.

As we mentioned before our results might have disadvantages compared to other runs if we take all the unjudged but retrieved documents into consideration. Despite this disadvantage, our result is still competitive to the result of ATT that ranked the 2nd (0.461 in the LF1 measure) at TREC-7.

As Figures 2 and 3 show, BayesBoost outperforms other runs when given more positive training examples. But it shows a slightly worse performance than other runs when positive example are sparse. It is related to the fact that naive Bayes is a probabilistic model, so it is very unlikely to make accurate predictions given a small number of examples.

5 Conclusion and Future Work

Our results on TREC-7 and TREC-8 filtering experiments show that BayesBoost does significantly well in comparison to the best results submitted by other TREC entries. However, a moderate performance was observed when the text collection did not have many positive training examples.

Our experimental results also show that BayesBoost outperforms AdaBoost (with decision stumps) when applied to text filtering on TREC-7 and TREC-8 data. This seems due to the following features of BayesBoost. First, because it is a probabilistic model, it is suitable for calculating confidence ratios. Second, naive Bayes makes use of weight information as opposed to decision stumps or decision trees.

Possible areas of future research include sophisticated term selection. Automatic selection of the term size will also improve the accuracy of naive Bayes classifiers and BayesBoost.

Acknowledgements

This research was supported by Korea Ministry of Information and Telecommunication under Grant 98-199 through IITA, by Korea Science and Engineering Foundation through AITRC, and by the BK21-IT Program.

References

[1] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin?. *Communications of the ACM*, 35(12):29-38, 1992.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123-140, 1996.

[3] C. Buckley and G. Salton. Optimization of relevance feedback weights. In *Proc. SIGIR-95*, pp. 351-357, 1995.

[4] H. Drucker and C. Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems 8*, pp. 479-485, 1996.

[5] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th Int. Conf. on Machine Learning*, pp. 148-156, 1996.

[6] D. Hull. The TREC-8 filtering track: Description and analysis. In *Proc. 7th Text Retrieval Conf.* (TREC-7), pp. 33-56, 1998.

[7] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proc. Int. Conf. on Machine Learning* (ICML-97), pp. 143-151, 1997.

[8] K. L. Kwok, L. Grunfeld, M. Chan, N. Dinstl, and C. Cool. TREC-8 ad-hoc, query and filtering track experiments using PIRCS. In *Proc. Text Retrieval Conf.* (TREC-8), pp. 107-116, 1998.

[9] D. Lewis, R. E. Schapire, J. P.Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proc. SIGIR-96*, pp. 298-306, 1996.

[10] David Lewis. Evaluating and optimizing autonomous text classification systems. In *Proc. SIGIR-95*, pp. 246-255, 1995.

[11] A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proc. Int. Conf. on Machine Learning* (ICML-98), pp. 350-358, 1998.

[12] J. R. Quinlan. bagging, boosting and C4.5. In *Proc. AAAI-96*, pp. 725-730, 1996.

[13] R. E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297-336, 1999.

[14] R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651-1686, 1998.

[15] R. E. Schapire, Yoram Singer, and Amit Singhal. Boosting and Rocchio applied to text filtering. In *Proc. SIGIR-98*, pp. 251-223, 1998.

[16] A. Singhal, M. Mitra, and C. Buckley. Learning routing queries in a query zone. In *Proc. SIGIR-96*, pp. 21-29, 1996.

[17] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proc. SIGIR-96*, pp. 21-29, 1996.

[18] D. K. Harman. Overview of 8th Text Retrieval Conference (TREC-8). In *Proc. 8th Text Retrieval Conf.* (TREC-8), pp. 1-19, 1999.

[19] Y. Yang and X. Liu. A Re-examination of text categorization methods. In *Proc. SIGIR-99*, pp. 42-49. 1999.