Using Bayesian Priors to Combine Classifiers for Adaptive Filtering

Yi Zhang
Language Technology Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15232, USA
yiz@cs.cmu.edu

ABSTRACT

An adaptive information filtering system monitors a document stream to identify the documents that match information needs specified by user profiles. As the system filters, it also refines its knowledge about the user's information needs based on long-term observations of the document stream and periodic feedback(training data) from the user. Low variance profile learning algorithms, such as Rocchio, work well at the early stage of filtering when the system has very few training data. Low bias profile learning algorithms, such as Logistic Regression, work well at the later stage of filtering when the system has accumulated enough training data.

However, an empirical system needs to works well consistently at all stages of filtering process. This paper addresses this problem by proposing a new technique to combine different text classification algorithms via a constrained maximum likelihood Bayesian prior. This technique provides a trade off between bias and variance, and the combined classifier may achieve a consistent good performance at different stages of filtering. We implemented the proposed technique to combine two complementary classification algorithms: Rocchio and logistic regression. The new algorithm is shown to compare favorably with Rocchio, Logistic Regression, and the best methods in the TREC-9 and TREC-11 adaptive filtering tracks.

1. INTRODUCTION

An information filtering system monitors a document stream to find the documents that satisfy users' information needs. The potentially relevant documents must be delivered immediately, thus the system has no time to accumulate and rank a set of documents as a traditional retrieval system does. Instead, a filtering system usually makes a binary decision to accept or reject a newly arrived document for each user. While filtering, *an adaptive information filtering* system receives periodic feedback from the user about whether a delivered document is good, which provides training data for online learning.

The filtering task is similar to the text classification task and the ad-hoc retrieval relevance feedback task. Algorithms used for text classification tasks or the ad-hoc retrieval relevant feedback task, such as SVM, logistic regression, Naive Bayes, decision tree, language modelling, or Rocchio, can be used for profile updating[1]. Although a large literature about these algorithms exists, there is no conclusion about which algorithm works best. Because the answer depends on the data set. In order to understand which algorithm works better under what kind of situations, we can decompose the generalization error of a learning algorithm into 3 parts:

- Bias: Measure of how closely the learning algorithm is able to approximate the best solution.
- Variance: Measure of how sensitive the learning algorithm is to the training sample.
- Noise: Measure of the inherent irreducible uncertainty of the problem. For example, for a given x there are more than one possible y.

A high variance means the learning algorithm converges to its asymptotical classifier very slow. A high bias means the asymptotical classifier is far from the optimal classifier. For problems with many training samples, the bias can be the dominant contributor to the generalization error, while for problems with very few training samples, the variance may be a dominant contributor. The purpose of profile learning is to find a classifier with the least generalization error using the training data available. One may want to use a learning algorithm with both low bias and low variance. However, there is a natural "bias-variance trade-off" for any learning algorithm[6].

Bias-Variance Dilemma As the complexity of the learning algorithm increases, the bias goes down, but the variance goes up.

Because of this dilemma, some complex learning algorithms, such as SVM or logistic regression, work well when the number of training data is big, but some simple algorithms, such as naive Bayesian or Rocchio, work better when the number of training data is very small.

[1]If using ad-hoc retrieval relevance feedback algorithms such as Rocchio, a thresholding module is needed for a filtering system to make online binary judgments.

SIGIR'04, July 25–29, 2004, Sheffield, South Yorkshire, UK.

At the early stage of filtering, we have very few training data, thus a low variance learning algorithm could be a better choice. When we have enough training data later based on long term interaction with the user, a low bias learning algorithm may work better. However, an adaptive filtering system needs to work consistently well in the whole filtering process.

This paper proposes to solve this problem using a Bayesian Prior to combine two different models. The combined algorithm may get a good bias variance trade off based on the size of the training set, thus achieve a consistent good performance at different stages of filtering. We applied the proposed technique to combine Rocchio with logistic regression. An adaptive filtering system using the new algorithm behaves similar to Rocchio at the early stage of filtering, and becomes more like logistic regression as more user feedback are available. The experimental results show that the proposed algorithm is significantly better than both Rocchio and logistic regression, and compare favorably with the best methods in the TREC-9 and TREC-11 adaptive filtering tracks.

This paper is organized as follows. Section 2 introduces the definition of Rocchio algorithm, logistic regression algorithm. A new algorithm Logistic_Rocchio that combines Rocchio and logistic regression is introduced in Section 3. Section 4 and Section 5 describes our experimental methodology and results. Further discussion and some related works are provided in Section 6. Section 7 concludes.

2. ROCCHIO AND LOGISTIC REGRESSION FOR FILTERING

At a certain point in the adaptive filtering process, suppose we have t training documents with user feedback $D_t = (X,Y)_t = [(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_t}, y_t)]$, where $\mathbf{x_i}$ $(i = 1$ to $t)$[2] is a vector that represents the document d_i in a K dimensional space indexed by K keywords[3]. y=1 if the document $\mathbf{x}$ is relevant, otherwise y=-1. The core problem in relevancy filtering is estimating the posterior probability of relevancy of document $\mathbf{x}$ based on the training data: $P(y = 1|\mathbf{x}, D_t)$.

As mentioned before, many algorithms can be used for profile learning. This section introduces two very representative algorithms: Rocchio algorithm and logistic regression.

2.1 Rocchio algorithm

A widely used profile updating methods in the information retrieval community are different variations of the incremental Rocchio algorithm [1][5][23][2], which can be generalized as:

$$\mathbf{Q}' = \alpha \cdot \mathbf{Q} + \beta \frac{\sum_{\mathbf{x_i} \in R} \mathbf{x_i}}{|R|} - \gamma \frac{\sum_{\mathbf{x_i} \in NR} \mathbf{x_i}}{|NR|} \quad (1)$$

where $\mathbf{Q}$ is the initial profile vector, $\mathbf{Q}' = (w_{r1}, .., w_{rK})$ is the new profile vector, R is the set of relevant documents, and NR is the set of non-relevant documents.

When a document arrives, the Rocchio algorithm only provides a score indicating how well the document matches each user profile[4]. The score is calculated by measuring the distance between the document vector and the user profile Q'. Because an adaptive system needs to make a binary decision for each incoming document (e.g., choose from action "deliver" or "not deliver"), so researchers usually have to use another module to learn the dissemination thresholds [24][27]. The filtering system will deliver document $\mathbf{x}$ to the user if and only if its score is above the dissemination threshold. The corresponding decision rule is:

$$\text{Deliver iff } (w_{r1}, .., w_{rK})^T \mathbf{x} >= threshold \quad (2)$$

Let $w_{r0} = -threshold$, $\mathbf{w_R^T} = (w_{r0}, w_{r1}, .., w_{rK})$. If we make $\mathbf{x}$ a K+1 dimension vector with the first dimension corresponds to a pseudo feature which is always equal to 1, the above equation can be rewritten as[5]

$$\text{Deliver iff } \mathbf{w_R^T x} >= 0 \quad (3)$$

The Rocchio algorithm is popular for two reasons. First, it is computationally efficient for online learning [1]. Second, compared to many other algorithms, it works well empirically [2][27]. However, the Rocchio algorithm is not based on a strong probabilistic framework and there is no guarantee about whether it will provide the optimal decision boundary in the high dimensional document space asymptotically, with infinite training data. In other words, Rocchio algorithm is a simple algorithm with high bias and low variance.

2.2 Logistic regression

Logistic regression is one widely used statistical algorithm that can provide a comparatively good estimation of the posterior probability $P(y|\mathbf{x})$ of an unobserved variable y given an observed variable $\mathbf{x}$. It has been widely used in the statistical and machine learning community.

A logistic regression model estimates the posterior probability of y via a log linear function in observed document $\mathbf{x}$.

$$P(y = \pm 1|\mathbf{x}, \mathbf{w}) = \sigma(y\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp(-y\mathbf{w}^T\mathbf{x})}$$

where $\mathbf{w}$ is the K dimensional logistic regression model parameter learned from the training data.

Before filtering, the system usually begins with a certain prior belief $p(\mathbf{w})$ about the distribution of the logistic regression parameter $\mathbf{w}$. Prior is a very important notion in Bayesian theory. Using a prior benefits us in two aspects: 1) it provides a tool for introducing human knowledge into model building; and 2) it acts as a regularizer to control the model complexity and avoid over fitting.

A Gaussian distribution $p(\mathbf{w}) = N(\mathbf{w}; \mathbf{m_w}, v_w)$ is often used as the prior distribution for logistic regression weights, where $\mathbf{m_w}$ is the mean of the Gaussian distribution in the K dimensional parameter space and v_w^{-1} is a $(K+1) \cdot (K+1)$ covariance matrix of the Gaussian distribution.

If there is no obvious correlation between the different dimensions of $\mathbf{w}$, or if we can not tell what the correlations are, the off diagonal values in the matrix v_w^{-1} are usually set to zero. If we are not very confident about whether the

[2]Through out this paper, we use a symbol in bold font to represent a vector.

[3]Any weighting schema, such as $TF \cdot IDF$, can be used to convert a document d_i into its vector representation $\mathbf{x_i}$.

[4]The same is true for many other statistical retrieval algorithms that were originally designed for ad-hoc retrieval, where a ranking of documents is sufficient.

[5]For convenience, depends on the context, we will use $\mathbf{x}$ to represent K dimensional vector or K+1 dimensional vector in the paper.

true value of $\mathbf{w}$ is $\mathbf{m_w}$, the diagonal variables of the matrix v_w^{-1} are usually set to a small number. If all items in matrix v_w^{-1} are zero, $p(\mathbf{w})$ is a non-informative prior: all values of $\mathbf{w}$ have the same probability. Non-informative prior may seems *objective* as it represents the idea of *letting the data speak for themselves*. However, classifier learned using a non-informative prior usually over fits the data. So, v_w^{-1} is usually set to a diagonal matrix with a small non zero positive number on the diagonal to act as a regularizer. The effect is the same as smoothing techniques used while building language models [26]. [26] is smoothing with priors for generative models, while we are regularizing with priors for discriminative models.

At a certain point in the adaptive filtering process, we can update our belief about the logistic regression parameter $\mathbf{w}$ conditional on the training data D_t using Bayesian theorem.

$$\begin{aligned} P(\mathbf{w}|D_t) &= \frac{P(D_t|\mathbf{w})P(\mathbf{w})}{\int_{\mathbf{w}} P(D_t|\mathbf{w})P(\mathbf{w})} \\ &= \frac{\prod_{i=1}^{t} P(y_i|\mathbf{w},\mathbf{x_i})P(\mathbf{w})}{\int_{\mathbf{w}} \prod_{i=1}^{t} P(y_i|\mathbf{w},\mathbf{x_i})P(\mathbf{w})} \end{aligned}$$

With a Gaussian prior $N(\mathbf{w};\mathbf{m_w},v_w))$, the max a posterior estimation (MAP) of $\mathbf{w}$ is:

$$\begin{aligned} \mathbf{w}_{MAP_t} &= \underset{\mathbf{w}}{argmax} P(\mathbf{w}|D_t) \\ &= \underset{\mathbf{w}}{argmax} \prod_{i=1}^{t} P(y_i|\mathbf{w},\mathbf{x_i})P(\mathbf{w}) \\ &= \underset{\mathbf{w}}{argmax} \sum_{i=1}^{t} log(1+\exp(-y_i\mathbf{w}^T\mathbf{x_i})) - v_w(\mathbf{w}-\mathbf{m_w})^2 \end{aligned}$$

There is no close form solution for $\mathbf{w}_{MAP_t}$, so greedy search algorithms such as Conjugate gradient descent, are often used to find it. In most of the cases where we have no prior knowledge about what the logistic regression parameter is, we usually set $\mathbf{w} = (0,...,0)$ and v_w^{-1} to a very small value. Then, we get a norm-2 regularized logistic regression model:

$$\mathbf{w}_{MAP_t} = \underset{\mathbf{w}}{argmax} \sum_{i}^{t} log(1+\exp(-y_i\mathbf{w}^T\mathbf{x_i})) - v_w\mathbf{w}^2 \quad (4)$$

When a new document $\mathbf{x}$ arrives, we can estimate the probability of relevancy for this document:

$$P(y=1|\mathbf{x}) = P(y=1|\mathbf{x},\mathbf{w}_{MAP_t}) \quad (5)$$

3. LOGISTIC_ROCCHIO: COMBINE ROCCHIO WITH LOGISTIC REGRESSION USING A BAYESIAN PRIOR

At the early stage of filtering when the system has very few training data, a simple profile learning algorithm with low variance, such as Rocchio, may be a good choice. At the later stage of filtering when the system has enough training data, a complex profile learning algorithm with low bias, such as logistic regression, may be better. How do we get an algorithm that works well consistently over the whole filtering process? One possible approach is to use Rocchio first, then switch to logistic regression later when the number of training data is enough. Another approach is to combine Rocchio and logistic regression together to get a natural and smooth trade off between variance and bias. The second approach, which will be explored further in this section, is the motivation for the new algorithm, a combination of Rocchio and logistic regression based on a Bayesian prior.

As mentioned before, a Bayesian prior is a widely used tool for introducing human knowledge into model building; and it is also a regularizing tools for controlling the model complexity and avoiding over fitting. An adaptive filtering task begins with a very small amount of training data, thus a stable prior that works well with little training data is very important for building a good filtering system. Since the Rocchio algorithm is designed by researchers in IR community and works well empirically in the adaptive filtering task, using it to set a Bayesian prior of the logistic regression model parameter may help us to achieve stable performance at the early stage of filtering.

Usually, a logistic regression prior is a Gaussian distribution $N(\mathbf{m_w},v_w)$. How to find the prior mean $\mathbf{m_w}$ from Rocchio is not straight forward. A new technique is proposed here to solve this problem.

Let $\mathbf{w_R} = (w_{r0},w_{r1},w_{r2},w_{r3}...,w_{rK})$ be the profile vector calculated based on the Rocchio algorithm (Equation 3).

For logistic regression, we use the same representation as Rocchio for documents: the same set of keywords with the same weighting schema, plus a pseudo dimension, which is always 1, as features. The probability of relevancy of a given document $\mathbf{x}$ based on logistic regression model w is:

$$P(y=1|\mathbf{x},\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$$

If the goal is to minimize classification error, a filtering system using the logistic regression model will:

$$\text{Deliver iff } \mathbf{w}^T\mathbf{x} >= 0 \quad (6)$$

Equation 3 and Equation 6 looks very similar. If the threshold of Rocchio learner is set to minimize classification error too, both algorithms are trying to find a linear decision boundary in the high dimensional space indexed by the set of keywords for the same objective. The Rocchio algorithm tends to be more stable at the early stage of filtering, and the decision boundary find by Rocchio is likely to be better than the one find by logistic regression. A Gaussian prior $N(\mathbf{w};\mathbf{m_w},v_w)$ for logistic regression encodes the belief that the true decision boundary is around the one defined by $\mathbf{m_w}$. If we set $\mathbf{m_w}$ so that the decision boundary of it is the same as the one find by Rocchio, $N(\mathbf{w};\mathbf{m_w},v_w)$ may be better than the commonly used non-informative prior or zero mean Gaussian prior.

A prior $\mathbf{m_w}$ that encodes Rocchio's suggestion about decision boundary can be learned via constrained maximum likelihood estimation:

$$\mathbf{m_w} = \underset{\mathbf{w}}{argmax} \sum_{i=1}^{t} log(\frac{1}{1+\exp(-y_i\mathbf{w^T x_i})}) \quad (7)$$

$$\text{under the constraint: } \cos(\mathbf{w},\mathbf{w_R}) = 0 \quad (8)$$

The resulting logistic regression parameter $\mathbf{m_w}$ maximizes the likelihood of the data (Equation 7) under the constraint that it corresponds to the same decision boundary as the Rocchio algorithm (Equation 8). This is an optimization problem, and the solution is in a simple form that can be calculated efficiently:

$$\mathbf{m_w} = \alpha^* \cdot \mathbf{w_R} \quad (9)$$

where α is a scaler:

$$\alpha^* = \underset{\alpha}{argmax} \sum_{i=1}^{t} \frac{1}{1+\exp(-y_i \alpha \mathbf{w_R^T x_i})}$$

This is a one dimension optimization problem, and the solution can be find quickly using gradient descent algorithms.

At the early stage of filtering when the number of training data is small, the prior is very important, thus the influence of the Rocchio algorithm on the logistic regression model is strong. When the system gets more and more training data, the prior is less important, and logistic regression dominates. This technique automatically manages the trade off between bias and variance based on the number of training data available. With the Bayesian prior estimated using Rocchio, the logistic regression parameter estimated has a higher bias but lower variance than without a prior. As more and more training data are available, the bias decreases. Asymptotically, the learned classifier converges to the optimal linear decision boundary in the high dimensional document space.

4. EXPERIMENTAL METHODOLOGY

Some experiments were carried out to understand the proposed new algorithm and compared it with Rocchio algorithm (Equation 1), Norm 2 regularized logistic regression algorithm 4, logistic regression with Rocchio weights $\mathbf{w_r}$ a mean, and the best methods in TREC-9 and TREC-11 adaptive filtering tracks . We set the variance of the prior according to our confidence about the prior distribution. For example, we feel more confident about the prior mean set by the scaled Rocchio weight $\alpha^* w_R$, thus we set the variance of the prior mean to a comparatively smaller number, 10I. While using other prior means, such as zero used in the Norm 2 regularized logistic regression model, we are less confident about whether they are close to the optimal or not, thus we set the variance of the prior mean to a bigger number, 100I. The Rocchio weights $(\alpha, \gamma, \lambda)$ is set to $(1, 3.5, 2)$ as [27].

The experiments follow the requirements specified by TREC adaptive filtering track[18][19]. The whole task models the text filtering process from the moment user arrives with a small amount of identified relevant documents and a natural language description of the information need. Documents arrive before user profile construction, among which only 2 or 3 are labelled, can be used as training data, to learn word occurrence (e.g., idf) statistics, corpus statistics (e.g., average document length), and the initial profile representation. Remaining documents in the incoming documents stream are testing data. As soon as a new document arrives, the system makes a binary decision about whether or not deliver it to the user. If it is delivered, relevance judgment for that document is released to the system and added to the training data set for profile updating.

The system treats the initial user profile description, which includes the title and description fields of the corresponding topic provided by NIST, as a relevant document while training logistic regression models. Because TREC adaptive filtering runs begin with no non-relevant documents, our system randomly sampled a small number (<= 3) documents and uses them as non-relevant documents to train logistic regression and Logistic_Rocchio models. Documents are represented as vectors using a variation of INQUERY's $TF \cdot IDF$ weighting [1] [6]

[6]We set $x_{i,d} = tfbel_{i,d} \cdot idf_i$, where $tfbel_{i,d} = \frac{tf_{i,d}}{tf_{i,d}+0.5+1.5\frac{len_d}{avgDocLen}}$, $idf_i = \log(\frac{N+0.5}{df_i}/\log(N+1))$, $tf_{i,d}$ is the number of times term i occurs in document d, len_d is the length of document d, $avgDocLen$ is the average length of documents processed, N is the number of documents processed, and df_i is the number of documents that contain term i.

Table 1: The values assigned to relevant and non-relevant documents that the filtering system did and did not deliver.

	Relevant	*Non-Relevant*
Delivered	R^+, A_R	N^+, A_N
Not Delivered	R^-, B_R	N^-, B_N

4.1 Data sets

Two different text corpora were used to evaluated the proposed algorithms in our experiments: the OHSUMED data set used in the TREC-9 Filtering Track, and the Reuters 2001 data set used in the TREC-11 Filtering Track.

4.2 TREC9 OHSUMED Data

The OHSUMED data set is a collection from the US National Library of Medicine's bibliographic database [7]. It was used by the TREC-9 filtering track [18]. It consists of 348,566 articles from a subset of 270 journals covering the years 1987 to 1991. 63 OHSUMED queries were used to simulate user profiles. The relevance judgments for the OHSUMED queries were made by medical librarians and physicians based on the results of interactive searches. In the TREC-9 adaptive filtering track it is assumed that the user profile descriptions arrive at the beginning of 1988. The average numbers of relevant articles per profile in the testing data are 51 for the OHSUMED topics. However, only two relevant documents and zero non-relevant documents are given as labelled training data for a filtering to begin with.

4.3 TREC11 Reuters 2001 Data

The Reuters 2001 data(also called the RCV1 corpus) provided by Reuters is a collection of about 800,000 news stories. It covers a time period of 1996 to 1997. This corpus was used by TREC-11 filtering tracks [19].

50 topics created by TREC assessors were used to simulate user profiles. The relevance judgments for these queries were collected using extensive searches with multiple rounds of multiple retrieval systems after an initial definition of the topics. Two sets of relevance judgments on this data set are available. The first set is used by the official TREC runs. The second set is released after official TREC runs, and is a modified version of the first set. We will show experiments using both sets of relevance judgements

The overall number of relevant articles in the testing data is about 9 to 599 documents per profile. The documents from 20 August through 20 September 1996 are used as initial training data, however, only 3 documents that are relevant for the user topic are labeled, while other documents in the training set are unlabeled.

4.4 Evaluation measures

A linear utility function is usually used to model user satisfaction and evaluate a filtering system. A general form of

Table 2: A comparison of different algorithms on the TREC-9 OHSUMED data set.

	Rocchio	LR1	LR2	LR_Rocchio
Prior Mean	NA	0	$\mathbf{w}_R$	$\alpha_*\mathbf{w}_R$
T11SU	0.37	0.36	0.12	**0.39**
T9U	11.38	10.43	-24.54	**15.03**
Precision	0.37	0.30	0.10	0.37
Recall	0.19	0.21	0.20	0.23
Doc/Profile	20.10	24.67	62.87	23.78

the linear utility function used in the recent TREC Filtering track is shown below [18].

$$Utility = A_R \cdot R^+ + A_N \cdot N^+ + B_R \cdot R^- + B_N \cdot N^- \quad (10)$$

This model corresponds to assigning a positive or negative value to each element in the categories of Table 1, where R^-, R^+, N^-, and N^+ correspond to the number of documents that fall into the corresponding category, and A_R, A_N, B_R and B_N correspond to the credit/penalty for each element in the category.

In the TREC-9, TREC-10 and TREC-11 filtering tracks, the following utility function was used:

$$T9U = T10U = T11U = 2R^+ - N^+ \quad (11)$$

A normalized version of T11SU was also used in TREC-11:

$$T11SU = \frac{max(\frac{T11U}{MaxU}, MinNU) - MinNU}{1 - MinNU} \quad (12)$$

where $MaxU = 2 * (R^+ + R^-)$ is the maximum possible utility and $MinNU = -0.5$.

T11U measure (Equation 11) and T11SU measure (Equation 10) are used for evaluation in the experiments reported here. In order to optimize the utility defined by Equation 11, the filtering system delivers a document if and only if

$$P(rel|document) > 0.33$$

.

5. EXPERIMENTAL RESULTS

The experimental results on the OHSUMED data set are in Table 2. The Rocchio algorithm has a reasonable performance, among the best compared to other filtering systems that participated in the TREC-9 adaptive filtering track [18][7]. Logistic regression with a prior $N(\mathbf{w}; 0, 100I)$ is a little worse, but still good. Logistic regression using the prior $N(\mathbf{w}; \mathbf{w_R}, 10I)$ is bad. This indicates using w_R directly estimated by Rocchio algorithm as the prior mean is very misleading. However, after scaling using Equation 9 and set the prior to $N(\mathbf{w}; \alpha^*\mathbf{w}_R, 10I)$, we get a significant improvement. This confirmed our hypothesis that combining Rocchio with logistic regression will work well. This is not surprising since using a low variance classifier to set the prior for a low bias classifier may provide a nice trade off between variance and bias.

[7]The T9U values of the top 3 TREC participants on this data set are: 17.3, 10.7 and 10.1.

Table 3: A comparison of different algorithms on the TREC-11 Reuters data set.

	Rocchio	LR1	LR2	LR_Rocchio
Prior Mean	NA	0	$\mathbf{w}_R$	$\alpha_*\mathbf{w}_R$
T11SU	0.46	0.49	0.16	**0.52**
T11U	63.06	76.60	-25.86	**83.10**
Precision	0.53	0.48	0.20	0.56
Recall	0.33	0.41	0.46	0.41
Doc/Profile	69.36	88.28	223.32	84.12

Table 4: A comparison of different algorithms on the TREC-11 Reuters data set using the old relevance judgments.

	Rocchio	LR1	LR2	LR_Rocchio
Prior Mean	NA	0	$\mathbf{w}_R$	$\alpha_*\mathbf{w}_R$
T11SU	0.46	0.49	0.11	**0.54**
T11U	40.20	54.66	-32.38	**61.68**
Precision	0.51	0.46	0.18	0.54
Recall	0.35	0.49	0.50	0.50
Doc/Profile	47.22	69.96	165.04	66.18

The experimental results on Reuters data set are in Table 3 and Table 4[8]. The Rocchio algorithm gets reasonable performance. Logistic regression with the prior with zero mean is a little better. Logistic regression using the Gaussian prior with w_R as the mean performs poorly. Using scaled Rocchio weight (Equation 9) as the prior performs the best.

Figures 1 and Figure 2 compare the performance of the new algorithm with the Rocchio and logistic regression algorithms on each individual profile using T11U measure to get an idea of how significant the improvement is on a query by query basis. We can see that for most of the profiles, the new algorithm works better (above the horizontal line). A statistical test (sign test) indicates that the new algorithm is significant better than Rocchio algorithm and logistic algorithm.

Figure 3 compares the performance of our system with other systems participated in the TREC11 adaptive filtering track[9]. In TREC11, each participant submitted 1-4 runs to NIST, and the best run of each individual participant is reported here. A system that delivers nothing will get T11SU=0.33, and some TREC-11 participating systems are lower than that. The logistic regression and the Rocchio algorithm are among the best of TREC participants, while the proposed algorithm (Logistic_Rocchio) is much better than the best runs submitted by the TREC-11 adaptive filtering track participants.

We also compared the accumulated performance of different profile learning algorithms *over time* on the TREC-11 adaptive filtering task (Figure 4). Rocchio works better than logistic regression at the early stage of filtering, but worse at later stages. Logistic regression with a Rocchio prior is the best at any time. This is not surprising based on the

[8]The final relevance judgments used in Table 3 are a little better than the old relevance judgements used in Table 4 (Section 4.3.).

[9]Our results and the results of TREC-11 participants are not directly comparable, because we have had greater experience with the dataset. Figure 3 is provided only to give context for the results reported in this paper.

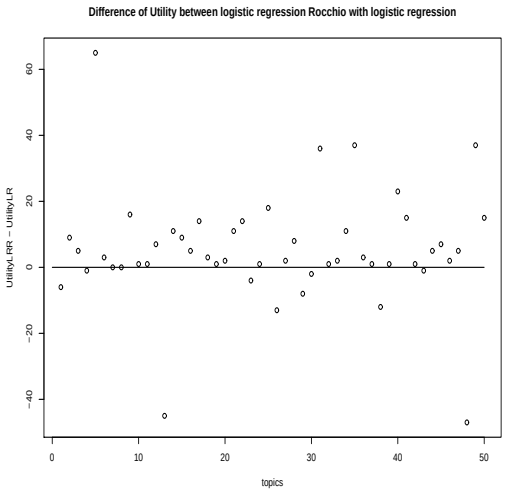

Figure 1: A comparison of the performance on different profiles: T11U of Logistic_Rocchio - T11U of LogisticRegression.

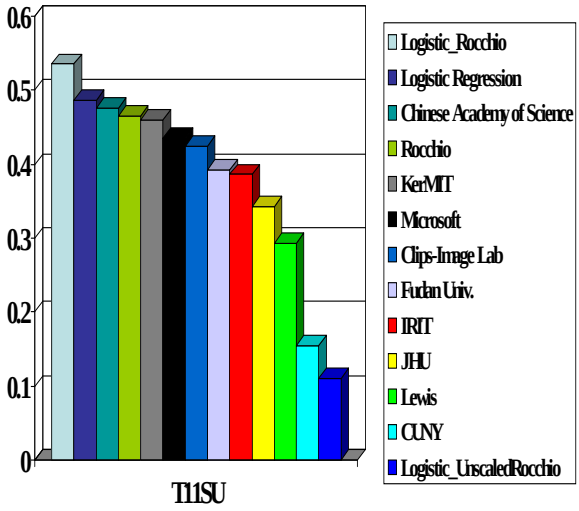

Figure 3: A comparison of our runs using T11SU evaluation measure on the TREC-11 adaptive filtering task with other TREC participants. Systems in the legend from top to bottom correspond to bars from left to right.

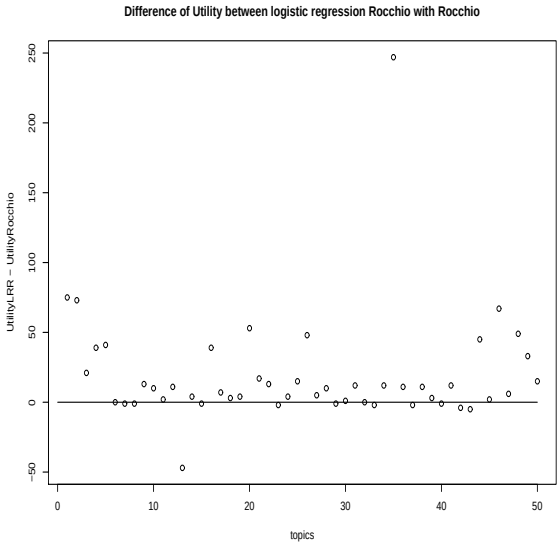

Figure 2: A comparison of the performance on different profiles: T11U of Logistic_Rocchio - T11U of Rocchio

bias variance analysis in Section 1. The new algorithm Logistic_Rocchio, is consistently much better than the other two.

6. RELATED WORK AND FURTHER DISCUSSION

Although our detailed analysis is focused on combining Rocchio and logistic regression, the proposed technique can also be applied to combining other learning algorithms. One may ask, how can we tell which algorithms we should combine? In other words, how can we tell which algorithm has a lower bias, and which algorithm has a lower variance?

We do not have a good answer to this question. The Bias-Variance Dilemma tells us that the complexity of the learning algorithm grows, the bias goes down, while the variance goes up. Unfortunately the complexity of classification algorithms are difficulty to measure, thus it is hard to use this dilemma empirically. In order to get a better answer to this question, we can compare the following two more general probabilistic approaches.

Discriminant approach: The posterior distribution $P(y|\mathbf{x})$ is modelled directly. For example, SVM [10] and logistic regression are popular discriminative models. In this approach, systems directly model the boundary between classes.

Generative approach: The distributions $P(\mathbf{x}|y)$ and $P(y)$ are modelled, and the posterior distribution $P(y|\mathbf{x})$ is derived using the Bayes rule:

$$P(y=1|\mathbf{x}) = \frac{P(y=1,\mathbf{x})}{P(\mathbf{x})} = \frac{P(y=1,\mathbf{x})}{\sum_y P(\mathbf{x}|y)}$$

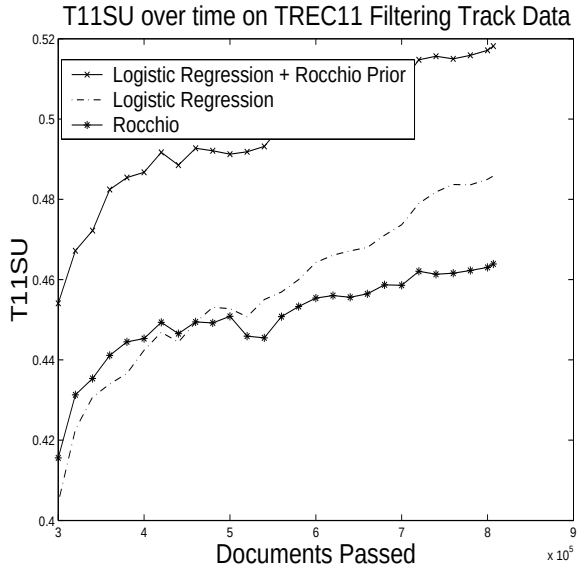

Figure 4: A comparison of the performance of different profile learning algorithms over time on the TREC-11 adaptive filtering task. This figure is the average of 50 user profiles.

For example, language models [13], Naive Bayes [15], and BIM [20] are popular generative models used in IR. In this approach, systems first model the characteristics of each classes, and then derive the boundary between classes. The non-probabilistic Rocchio algorithm has some generative model flavor in the sense that the centroid of relevant and non-relevant documents are estimated [9][21].

Some early work suggests that the generative models may have a low variance and work well with few training data, and comparable discriminative models may have a low bias and work better when the number of training data is big [16] [22]. This suggests combining comparable generative-discriminative models using the technique proposed in this paper may give a good performance in adaptive filtering system.

Because Rocchio has some generative flavor, and logistic regression is a comparable discriminative model, the comparison between the two more general approaches may be applied to Rocchio and logistic regression as well. This explains why Rocchio is likely to work well when training data is few, while logistic regression is likely to work well when the system has enough training data. Thus a filtering system using a Rocchio prior for logistic regression performs well.

The technique proposed in this paper has the flavor of a Bayesian analysis, but it is not a strict Bayesian approach, because the prior is estimated using the data. This is similar to Empirical Bayes methods[14].

There has been much research on combining different text classification algorithms or retrieval algorithms in the IR community[11] [25][8] [3] [12]. [11] picks up different classifiers for different categories using categorical features; [8] [25] [3] combine the output, or transformation of the output, of different classifiers using linear combinations; [4] combines out put of classifiers with probabilistic learning based on some reliability indicators.

In contrast to the previous research on text classification or retrieval task, our work is different in three aspects

- The focus of this paper is adaptive filtering task, where the number of training data changes over time.
- The previous work combines the output of various text classifiers, while the proposed techniques combines the classifiers based on better understanding of the variance, bias properties of the classifiers to be combined
- By using a low variance classifier to set the prior for a low bias classifier, the proposed technique automatically gets a nice trade off between variance and bias based on the size of training data set. The combined classifier has a low variance when the number of training data is small, and has a low bias when the number of training data is big.

We also noticed a recent independent work that combines generative/discriminative models [17] for bias variance trade off. However, they are focused on text classification instead of the adaptive filtering task. The combining technique and the effect of their algorithm are very different from ours. For example, their classifier does not converge to logistic regression's optimal estimation when the number of training data goes to infinity.

7. CONCLUSION

In this paper, we proposed a new technique to combine two classifiers, using constrained maximum Likelihood approach to learn a Bayesian prior for one classifier from another classifier. If the classifier used to learn the prior has a low variance, and if the classifier that uses the prior has a low bias, the proposed approach will provide a reasonable trade off between variance and bias based on the number of training data, so the combined algorithm will work well at different stages of filtering.

We developed a new algorithm to incorporate the Rocchio algorithm into logistic regression models using the proposed technique. We evaluated this new algorithm on the standard TREC adaptive filtering data sets. The performance is much better than both the Rocchio algorithm and logistic regression algorithm, comparable to the best in the TREC-9 adaptive filtering track, and much better than the best in the TREC-11 adaptive filtering track.

This work is important because it shows how to satisfy conflicting user requirements, i.e., learn the information need quickly, and continue to improve over time. Previous solutions required users to choose between rapid learning or long-term accuracy; now they can have both.

Although we are focusing on combining Rocchio with logistic regression, the proposed technique can also be applied to combine other classifiers. Early research work that compares generative models and discriminative models[16][22] suggests that generative models may have lower bias and higher variance than their discriminative counterparts. So using a generative model to set a Bayesian prior for its discriminative counterpart may get a good performance.

8. ACKNOWLEDGMENTS

We thank Jamie Callan, Wei Xu, Tom Minka, Yiming Yang, Stephen Robertson, Paul Ogilvie, Chengxiang Zhai, John Lafferty for valuable discussions about the work described in this paper. This work was sponsored by a Ph.D. Fellowship from IBM Corporation, and by the Advanced Research and Development Activity in Information Technology (ARDA) under its Statistical Language Modeling for Information Retrieval Research Program. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author, and do not necessarily reflect those of the sponsors.

9. REFERENCES

[1] J. Allan. Incremental relevance feedback for information filtering. In *SIGIR-96*, 1996.

[2] T. Ault and Y. Yang. knn, rocchio and metrics for information filtering at trec-10. In *TREC-10*. NIST, 2001.

[3] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In *Research and Development in Information Retrieval*, pages 173–181, 1994.

[4] P. N. Bennett, S. T. Dumais, and E. Horvitz. Probabilistic combination of text classifiers using reliability indicators: models and results. In *SIGIR-02*, 2002.

[5] J. Callan. Document filtering with inference networks. In *SIGIR-96*, 1996.

[6] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. pages 1–58. Neural Computation 4, 1992.

[7] W. Hersh, C. Buckley, T. J.Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *SIGIR-94*, pages 192–201, 1994.

[8] D. A. Hull, J. O. Pedersen, and H. Schütze. Method combination for document filtering. In *SIGIR-96*, 1996.

[9] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *ICML-97*, 1997.

[10] T. Joachims. Text categorization with support vector machine. In *ECML-98*, 1998.

[11] W. Lam and K.-Y. Lai. A meta-learning approach for text categorization. In *SIGIR-01*, 2001.

[12] L. S. Larkey and W. B. Croft. Combining classifiers in text categorization. In *SIGIR-96*, 1996.

[13] V. Lavrenko and W. B. Croft. Relevance-based language models. In *Research and Development in Information Retrieval*, pages 120–127, 2001.

[14] J. Maritz and T. Lwin. *Empirical Bayes Method*. Chapman and Hall, 2 edition, 1989.

[15] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[16] A. Y. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes. In *NIPS-02*, 2002.

[17] R. Raina, Y. Shen, A. Y. Ng, and A. McCallum. Classification with hybrid generative/discriminative models. In *(NIPS)*, 2003.

[18] S. Robertson and D. Hull. The TREC-9 Filtering track report. In *TREC-9*. NIST, 2001.

[19] S. Robertson and I. Soboroff. The (trec 2002) filtering track report. In *TREC-11*, 2002.

[20] S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. In *Journal of the American Society for Information Science*, volume 27, pages 129–146, 1976.

[21] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System–Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

[22] Y. D. Rubinstein and T. Hastie. Discriminative vs informative learning. In *KDD-97*, pages 49–53, 1997.

[23] R. Schapire, Y. Singer, and A. Singhal. Boosting and Rocchio applied to text filtering. In *SIGIR-98*, 1998.

[24] Y. Yang. A study on thresholding strategies for text categorization. In *SIGIR-01*, 2001.

[25] Y. Yang, T. Ault, and T. Pierce. Combining multiple learning strategies for effective cross-validation. In *ICML-00*, 2000.

[26] C. Zhai and J. Lafferty. A study of smoothing methods for langauge models applied to ad hoc information retrieval. In *SIGIR-01*, 2001.

[27] Y. Zhang and J. Callan. Maximum likelihood estimation for filtering thresholds. In *SIGIR-01*, 2001.