Maximum Likelihood Estimation for Filtering Thresholds

Yi Zhang Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15232, USA
{yiz, callan}@cs.cmu.edu

ABSTRACT

Information filtering systems based on statistical retrieval models usually compute a numeric score indicating how well each document matches each profile. Documents with scores above profile-specific *dissemination thresholds* are delivered.

An optimal dissemination threshold is one that maximizes a given utility function based on the distributions of the scores of relevant and non-relevant documents. The parameters of the distribution can be estimated using relevance information, but relevance information obtained while filtering is *biased*. This paper presents a new method of adjusting dissemination thresholds that explicitly models and compensates for this bias. The new algorithm, which is based on the Maximum Likelihood principle, jointly estimates the parameters of the density distributions for relevant and non-relevant documents and the ratio of the relevant document in the corpus. Experiments with TREC-8 and TREC-9 Filtering Track data demonstrate the effectiveness of the algorithm.

Keywords

Information Filtering, Dissemination Threshold, Maximum Likelihood Estimation.

1. INTRODUCTION

Information filtering systems monitor a document stream to find documents that match the information needs described by *profiles* consisting of queries and related context or history. Filtering systems based on statistical models (e.g., vector space, probabilistic, inference network) use a numeric score to indicate how well a document matches a profile, and only disseminate a document when its score is above some threshold (the *dissemination threshold*).

As the information stream is processed, the system may be provided with relevance judgments for some of the documents it delivers. An *adaptive* filtering system can learn from this feedback, so that it becomes more accurate over time.

SIGIR'01, September 9-12, 2001, New Orleans, Louisiana, USA.

Generally, an adaptive filtering system that can learn from user feedback has three important subtasks:

1. Learning corpus statistics, such as *idf* for each term;
2. Learning filtering profiles, for example, adding or deleting terms and adjusting term weights; and
3. Learning dissemination thresholds.

Most of the prior research on adaptive filtering has been focused on learning filtering profiles. A common approach to learning profiles is to use an incremental version of the Rocchio algorithm [1][2][3][4][5][6][20][21][22]:

$$Q' = \alpha Q + \beta \left(\frac{\sum_{D_i \in R} D_i}{|R|} \right) - \gamma \left(\frac{\sum_{D_i \in NR} D_i}{|NR|} \right)$$

where Q is the initial profile vector, Q' is the new profile vector, R is the set of relevant documents, NR is the set of non-relevant documents, D_i is a document vector, and α, β, and γ are constants indicating the relative value of each type of evidence. Machine learning algorithms for text classification have also been used for this task, for example, k-nearest neighbor, naive Bayes and boosting [6][7][8][18][19].

The problem of how to set filtering thresholds received little attention until recently, in part because many operational environments relied upon Boolean retrieval models and profiles that either matched or did not. Researchers working with statistical algorithms could delay dissemination decisions while scores were computed for each document in the stream, sort documents by their scores, and then disseminate the top N documents, as was done in the TREC Routing task. This approach is effective when it is not necessary to disseminate information quickly. When dissemination decisions cannot be delayed, it is necessary to find dissemination thresholds for each profile.

The problem of setting dissemination thresholds can be described as finding a threshold that maximizes a utility metric in which each relevant document is associated with some reward and each non-relevant document is associated with some penalty, as in recent TREC Filtering Track evaluations [6][14]. Much of the recent research on setting dissemination thresholds is based either on heuristics or regression methods [6][15][16][20][21][22]. For example, CLARITECH used a heuristic that allowed the threshold to vary between a lower bound utility value (zero) and an upper bound optimal value[20]. Microsoft Cambridge mapped a document score s_d to the a probability of relevance score p_d using the formula:

$$\log\left(\frac{P_d}{1-P_d}\right)=\beta\frac{s_d}{ast1}+\gamma$$

where γ is set according to another corpus, β is learned adaptively, and ast1 is the average score of the top 1% of retrieved documents [15]. KUN assumed a Gaussian distribution for the scores of relevant documents and an exponential distribution for the scores of non-relevant documents, and then found the threshold that maximized a given linear utility measure This approach was extremely effective, achieving the best result for utility oriented runs in the TREC9 Filtering Track evaluation [2].

One potential weakness with the KUN approach is that it assumes that the training data accurately represents the distribution of relevant and non-relevant document scores. This assumption is not true in an adaptive filtering environment, because relevance information is obtained only for documents that are actually disseminated. Relevance information is not available for documents that have scores below the threshold, so the training data is inherently biased.

This paper presents a new method for setting dissemination thresholds that, like the KUN method, assumes that relevant document scores are distributed normally and non-relevant document scores are distributed exponentially [2]. However, our algorithm explicitly models the sampling bias, and uses the maximum likelihood principle to find unbiased estimates of the parameters. It jointly estimates the parameters of the two density distributions and the ratio of the relevant documents in the corpus. Experiments indicate that this new method produces dissemination thresholds that are significantly more accurate than the baseline KUN method in some cases.

This following section describes the use of Maximum Likelihood Estimation (MLE) for modeling score distributions. Section 3 describes our experimental methodology and our new algorithm for setting dissemination thresholds based on these score distribution. Section 4 reports experimental results. Finally, Section 5 concludes.

2. MODELING SCORE DISTRIBUTIONS

2.1 A Mathematical Model of Document Score Distributions

Usually only a small proportion of documents is relevant. Given an accurate model of the distribution of document scores for the top ranking documents (relevant and non-relevant), the dissemination threshold can be set accurately.

Several researchers suggest modeling score distributions using a Gaussian distribution for the scores of relevant documents and an exponential distribution for the top ranking non-relevant documents [2][12]. These distributions are modeled as:

$$P(score \mid R=r)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(score-u)^2}{2\sigma^2}}$$

$$P(score \mid R=nr)=\lambda e^{-\lambda(x-c)}$$

where:

u : mean of the Gaussian distribution;

σ: variance of Gaussian distribution;

$1/\lambda$: variance of exponential distribution; and

c : minimum score a non-relevant document can get.

Analysis of experimental results on TREC9 Filtering Track data (OHSU topics, OHSUMED collection) support this approach. Figures 1 and 3 illustrate how the normal distribution fits relevant document scores for two TREC9 topics. Figures 2 and 4 illustrate how the exponential distribution fits the top-100 non-relevant document scores for the same topics.

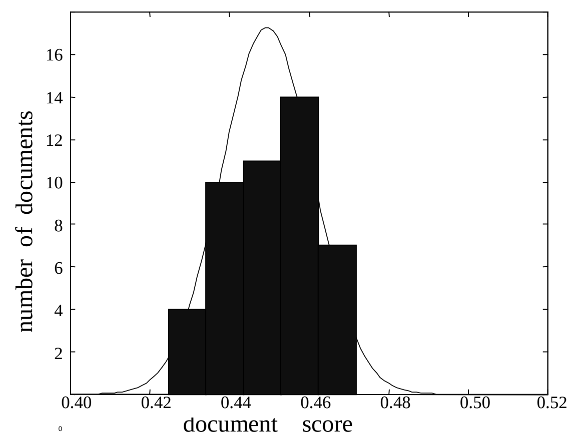

Figure 1. Density of relevant document scores: Topic5.

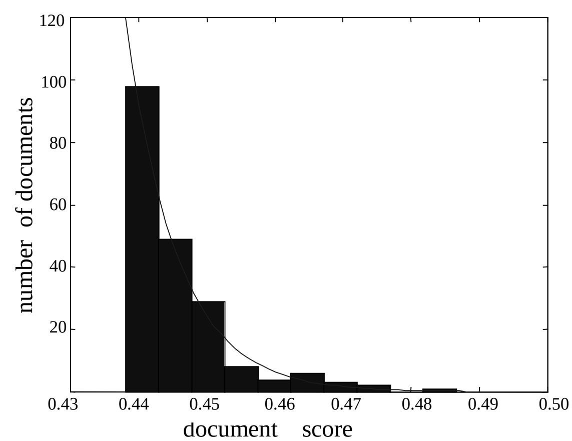

Figure 2. Density of non-relevant document scores: Topic5.

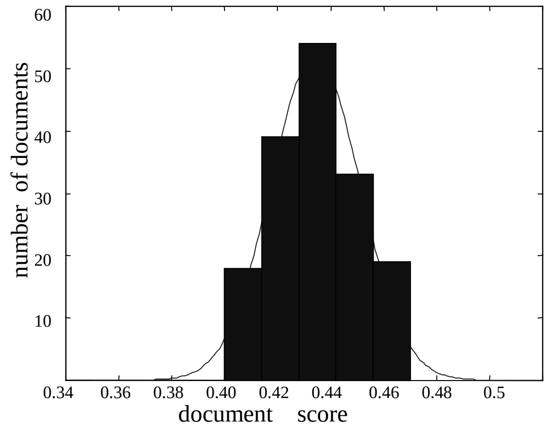

Figure 3. Density of relevant document scores: Topic3.

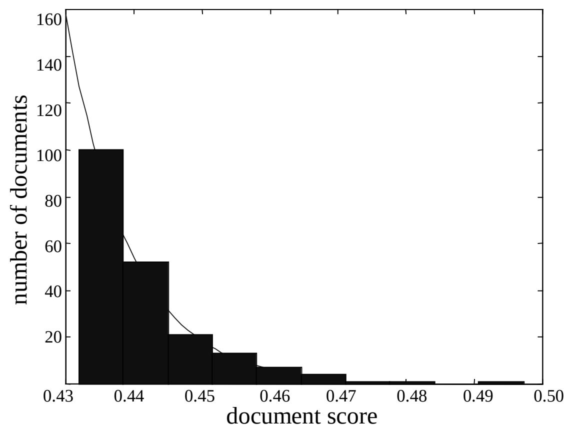

Figure 4. Density of non-relevant document scores: Topic3.

This model is not perfect, especially for low scoring documents. However, it can provide a relatively accurate estimate of the distribution of scores for higher-scoring documents, and we believe that in the information-filtering task the area of interest (i.e., the area in which thresholds are likely to be located) is among the higher-scoring documents.

2.2 Problem with KUN's Parameter Estimation Method

Given a set of scores, KUN's parameter estimation method for normal and exponential distributions is a simple calculation of the mean and variance over training data [2]. This approach to parameter estimation produces biased estimates, because in the information filtering task relevance information is available only for documents with scores above the dissemination threshold. When training data is restricted to scores above the threshold, the mean of the sample scores is likely to be higher than the real mean.

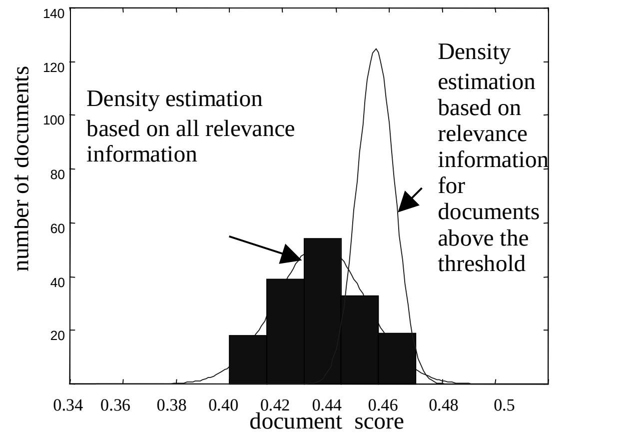

Figure 5. Estimation of parameters for relevant document scores: Topic 3.

For example, for the profile in Figure 3, the mean and variance of the Gaussian distribution (based on all relevant documents) are (0.4343, 0.0169). However, if training data is restricted to documents with scores above a fixed dissemination threshold of θ = 0.4435, the mean and variance calculated using KUN's method are (0.4551, 0007) (Figure 5).

In order to get an unbiased estimate of the distribution parameters, we must take into consideration the sampling constraint, i.e., the dissemination threshold. And in the real world of adaptive filtering, the threshold is changing over time, so the problem becomes more interesting.

2.3 Maximum Likelihood Estimation of Score Distribution Parameters

Maximum Likelihood Estimation, an unbiased parameter estimation method, can be used to solve this problem.

At a certain point in the filtering process, the filtering system has already delivered N documents to a user and relevance judgments of delivered documents are provided by the user. We can treat these documents as training data. For the i th delivered document with user feedback, let's represent it with a triple $(R_i, Score_i, \theta_i)$, where:

$$
\begin{aligned}
R_i \quad &= \begin{cases} r & \text{for relevant document} \\ nr & \text{for non-relevant document} \end{cases} \\
Score_i \quad &: \quad \text{The score of document } D_i\text{; and} \\
\theta_i \quad &: \quad \text{The threshold when } D_i \text{ was delivered.}
\end{aligned}
$$

In order to describe the density distribution of the scores, 4 parameters are necessary: (u, σ, λ, p). The meanings of u, σ, and λ are described in Section 2.1. p is the expected ratio of relevant documents in the corpus *according to the model*. p does not represent the ratio in the corpus as a whole, because the exponential model fits only the top non-relevant scores. The model is focused on, and is most accurate modeling, the scores of the top-ranking documents, where thresholds are typically set.

Given the observed training data, which are represented by a set of triples $D = \{(R_i, Score_i, \theta_i), i = 1\, to\, N\}$, according to Bayes theorem, the most probable value of H = (u, σ, λ, p) is:

$$\mathrm{H}^* = \arg\max_{\mathrm{H}} P(\mathrm{H} \mid D) = \arg\max_{h} \frac{P(D \mid \mathrm{H})P(\mathrm{H})}{P(D)} \quad (1)$$

For simplicity, we first assume that there is no prior knowledge of the distribution of *H* and treat the prior probability of *P(H)* as uniform. (We will revisit and remove the assumption in Section 3.3.) Because *P(D)* is a constant independent of *H*, it can be dropped. Thus the most probable *H* is the one that maximizes the likelihood of the training data.

$$\begin{aligned}(u^*, \sigma^*, \lambda^*, p^*) &= \arg\max_{(u,\sigma,\lambda,p)} P(D \mid (u,\sigma,\lambda,p)) \\ &= \arg\max_{(u,\sigma,\lambda,p)} \prod_{i=1}^{N} P(D_i \mid \mathrm{H}) \\ &= \arg\max_{(u,\sigma,\lambda,p)} \sum_{i=1}^{N} \log(P(D_i \mid \mathrm{H})) \\ &= \arg\max_{(u,\sigma,\lambda,p)} \sum_{i=1}^{N} \log(P(Score = Score_i, R_i \mid \mathrm{H}, Score > \theta_i))\end{aligned} \quad (2)$$

The second step of Equation 2 is due to the assumption that each document is independent; the third step is due to the fact that maximizing a function is equivalent to maximizing its logarithm; and the last step indicates the sampling constraints for training data. Notice that although the training data are not sampled randomly from the whole corpus, each individual training document is sampled randomly according to the conditional probability $P(Score = Score_i, R_i \mid H, Score > \theta_i)$. So parameters estimated based on Equation 2 are unbiased.

For each item inside the sum operation of Equation 2, we have:

$$\begin{aligned}&P(Score = Score_i, R_i \mid \mathrm{H}, Score > \theta_i) \\ &= \frac{P(Score = Score_i, Score > \theta_i, R_i \mid \mathrm{H})}{P(Score > \theta_i \mid \mathrm{H})} \\ &= \frac{P(Score = Score_i, Score > \theta_i \mid R_i, \mathrm{H})P(R_i \mid \mathrm{H})}{P(Score > \theta_i \mid \mathrm{H})} \\ &= \frac{P(Score = Score_i \mid R_i, \mathrm{H})P(R_i \mid \mathrm{H})}{P(Score > \theta_i \mid \mathrm{H})}\end{aligned} \quad (3)$$

The first step in Equation 3 is based on the definition of conditional probability; the second step is due to the chain rule of probability; and the last step is due to the fact that all training documents must have a score higher than the threshold.

For convenience, let $f_1(u,\sigma,\theta_i)$ be the probability of a document getting a score above threshold θ_i if it is relevant and $f_2(\lambda, \theta_i)$ be the probability of it getting a score above threshold if it is non-relevant. The probability of getting a score above θ_i can be calculated by integrating the density function from θ_i to positive infinity. That is:

$$\begin{aligned}&f_1(u,\sigma,\theta_i) = P(Score_i > \theta_i \mid R_i = r) \\ &= \int_{\theta_i}^{+\infty} P(Score = x \mid R_i = r)dx \\ &= \int_{\theta_i}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-u)^2}{2\sigma^2}} dx\end{aligned} \quad (4)$$

$$\begin{aligned}&f_2(\lambda,\theta_i) = P(Score_i > \theta_i \mid R_i = nr) \\ &= \int_{\theta_i}^{+\infty} P(Score = x \mid R_i = nr)dx \\ &= \int_{\theta_i}^{+\infty} \lambda e^{-\lambda*(x-c)} dx = e^{-\lambda(\theta_i - c)}\end{aligned} \quad (5)$$

If we use $g(u,\sigma,\lambda,p,\theta_i)$ to represent the probability of a document getting a score above threshold θ_i, we can get:

$$\begin{aligned}&g(u,\sigma,\lambda,p,\theta_i) = P(Score > \theta_i \mid \mathrm{H}) \\ &= P(R = r \mid \mathrm{H}) \cdot P(Score > \theta_i \mid R = r, \mathrm{H}) \\ &\quad + P(R = nr \mid \mathrm{H}) \cdot P(Score > \theta_i \mid R = nr, \mathrm{H}) \\ &= p \cdot f_1(u,\sigma,\theta_i) + (1-p) \cdot f_2(\lambda,\theta_i))\end{aligned} \quad (6)$$

An intuitive explanation of Equation 6 is illustrated in Figure 6. If we assume the sum of areas under the exponential and Gaussian curves is 1, then $p \cdot f_1(u,\sigma,\theta_i)$ corresponds to the area to the right of θ_i and below the normal distribution curve. $(1-p) \cdot f_2(\lambda,\theta_i)$ corresponds to the area to the right of θ_i and below the exponential distribution curve.

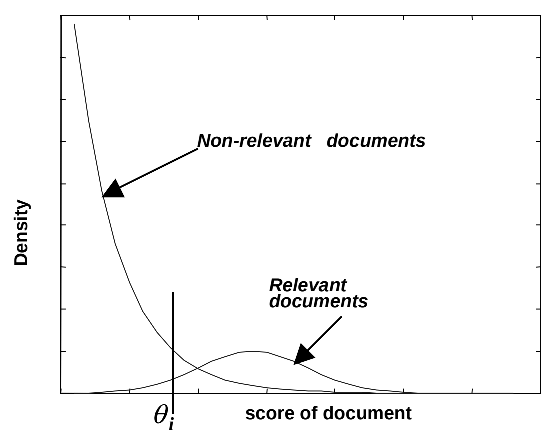

Figure 6. Distribution of scores of documents for a profile.

From Equations 2-6, we get:

$$(u^*, \sigma^*, \lambda^*, p^*) = \arg\max_{(u,\sigma,\lambda,p)} \sum_{i=1}^{N} LP_i \quad (7)$$

where for relevant documents:

$$LP_i = -\frac{(Score_i - u)^2}{2\sigma^2} + \ln(p/(\sigma \cdot g(u,\sigma,\lambda,p,\theta_i)))$$

and for non-relevant documents:

$$LP_i = -\lambda(Score_i - c) + \ln((1-p)\lambda / g(u,\sigma,\lambda,p,\theta_i))$$

If the training data are random samples of the whole corpus (not true for filtering), θ_i =-∞ and thus $g(u,\sigma,\lambda,p,\theta_i)$ = 1. If we replace $g(u,\sigma,\lambda,p,\theta_i)$ in Equation 7 with 1, the optimal parameters will be the same as those proposed by KUN. In other words, KUN's method is a maximum likelihood estimation of the parameters if the training data are a random sample of the dataset. However, $g(u,\sigma,\lambda,p,\theta_i)$ is not equal to 1 because the sample is biased during filtering. We must use Equation 7 to find the unbiased parameters.

2.4 Parameter Optimization Using Conjugate Gradient Descent

There is no closed form solution for Equation 7, so numerical methods must be used. One such method is conjugate gradient descent (abbreviated *CG*). CG is an iterative minimization procedure for a function $f(x)$. A sketch of the CG algorithm is shown in Figure 7. At each step, x is the approximate solution, q is the search direction, and x is improved by searching for a better solution along direction q. More detailed descriptions are available in [10][23].

The derivative of the right side of Equation 7 is used for calculating the search direction q. Taylor expansion is used to calculate the integration in Equation 4, and the Taylor expansion is guaranteed to converge.

```
x = initial guess for the minimum
q = negative of gradient at x (search direction)
do {
        x = the minimal point along direction h
        q = a linear combination of new gradient and old q
} until convergence
```

Figure 7. Sketch of the CG algorithm.

3. EXPERIMENTAL METHODOLOGY

The baseline and unbiased ML methods of setting dissemination thresholds were evaluated in a series of experiments with TREC data. The experimental methodology for the TREC Filtering Track changes from year to year. In our experiments we used the TREC-9 experimental methodology [14] in experiments with TREC-8 and TREC-9 Filtering data.

The advantage of this approach is that experimental results obtained with different datasets can be compared, because they were obtained with the same experimental methodology. However, a consequence is that our results with TREC-8 data are not directly comparable to results from systems participating in the TREC-8 Filtering Track.

The evaluation measures, datasets, and experimental methodology are described in more detail below.

3.1 Evaluation Measure

One commonly used evaluation measure for a filtering system is linear utility, which corresponds to assigning a positive value or negative cost to each element in the category [14].

	Relevant	**Non-relevant**
Delivered	R^+ / A	N^+ / B
Not Delivered	R^- / C	N^- / D

$$Utility = A \cdot R^+ + B \cdot N^+ + C \cdot R^- + D \cdot N^- \quad (8)$$

The variables R^+, N^+, R^-, and N^- are the number of documents in the corresponding category, and *A*, *B*, *C* and *D* are the benefit or cost associated with that category.

Filtering according to a linear utility function (Equation 8) is equivalent to setting the threshold at θ^*, where:

$$\frac{C-A}{B-D} \cdot \frac{p}{1-p} \cdot P(score = \theta^* \mid R_i = r) = P(score = \theta^* \mid R_i = nr)$$

The solution was solved in [2] by:

$$\theta^* = \begin{cases} (b - \sqrt{\Delta})/a & \text{if } \Delta \geq 0 \\ +\infty & \text{if } \Delta < 0 \end{cases} \quad (9)$$

where:

$$\Delta = b^2 - ad$$

$$a = \frac{1}{\sigma^2}$$

$$b = \frac{u}{\sigma^2} + \lambda$$

$$d = \frac{u^2}{\sigma^2} - 2\log(\lambda \cdot p \cdot \frac{1}{\lambda\sqrt{2\pi}\sigma}) + 2\lambda \cdot c$$

If we let (A, B, C D) = (2, 1, 0, 0), the utility becomes[1]:

$$T9U' = 2 \cdot R^+ - N^+ \quad (10)$$

[1] The TREC-9 T9U measure is *Max (2R⁺ - N+, MinU)*, where MinU=-100 for OHSU topics or –400 for Mesh topics. T9U sets a lower bound on profile utility, limiting the impact of poor profiles on overall filtering results. We have no need for the lower bound in our experiments, because all of the profiles get scores above MinU, so we opted for the simpler definition of utility

which is a slightly simplified variant of the T9U utility metric used in the TREC-9 Filtering track. This is the measure we used in our experiments. The corresponding delivery rule is:

$$\text{deliver if } P(R = r \mid score) > 0.33 \qquad (11)$$

3.2 Data

Two different text corpora were used to test our algorithm: the OHSUMED dataset used in the TREC-9 Filtering track and the FT dataset used in the TREC-8 Filtering track.

3.2.1 OHSUMED Data

The OHSUMED data is a collection from the US National Library of Medicine's bibliographic database [11]. It was previously used by the TREC9 Filtering Track [14]. It consists of 348,566 articles from the subset of 270 journals covering the years 1987 to 1991. 63 OHSUMED queries and 500 MeSH headings were used to simulate user profiles. The relevance judgments for the OHSUMED queries were made by some medical librarians and physicians on the basis of output of interactive search [11]. For the Mesh headings, assignment of a heading to a document by the National Library of Medicine is treated as equivalent to a positive relevance judgment.

In the TREC9 Filtering Track, it is assumed that the user profile descriptions arrive at the beginning of 1988, so the 54,709 articles from 1987 can be used to learn word occurrence (e.g., idf) statistics and corpus statistics (e.g., average document length). For each user, the system begins with a natural language description of the information need and 2 examples of relevant documents. The average numbers of relevant articles in the testing data are 51 for the OHSUMED topics and 249 for the Mesh headings.

3.2.2 FT Data

The FT data is a collection of 210,158 articles from the 1991 to 1994 Financial Times. It was previously used by the TREC8 Filtering Track [6]. TREC topics 351-400 were used to simulate user profiles. The relevance judgments were made by NIST on the basis of pooled output from several searches.

For each profile, the system begins with two identified relevant documents and a natural language description of the information need, which is the title and description field of the corresponding TREC topic. The average number of relevant articles in the testing data is 36.

3.3 Methodology

The YFilter information filtering system [22] was used for our experiments. It processes documents by first removing unusual symbols (such as punctuation and special characters), excluding the 418 highly frequent terms listed in the default INQUERY stop words list [3], and then stemming using the Porter stemmer [9]. Processed documents are compared to each profile using the BM25 tf.idf formula [17] to measure the similarity of the document and user profile.

For each topic, the filtering system created initial filtering profiles using terms from the TREC topic Title and Description fields, and set the initial threshold to allow the highest-scoring $\delta = 3$ documents in the training dataset to pass. For simplicity, the filtering profile term weights are not updated while filtering.

Because the first two relevant documents given to the system were not sampled under the constraint that their scores must exceed the dissemination threshold, their probabilities are simply $P(d_i \mid R_i = r)$, and the corresponding element of Equation 7 was changed to:

$$LP_i = -\frac{(Score_i - u)^2}{2\sigma^2} - \ln(\sigma)$$

In Section 2.3, for simplicity we set the prior probability of $P(\mathrm{H})$ to be uniform. However, in the real filtering task, especially during the early stage of filtering where there are only a small number of samples, this may cause problems. For example, if only non-relevant documents are delivered, the estimate of p will be 0 without a prior. Or, if all the relevant documents have the same score, the variance will be 0. Smoothing using a prior of parameters can solve these problems. The prior needn't be very accurate, because as the amount of sample data increases, the influence of the prior decreases. In our experiments, we set the prior of p as a beta distribution[2]: $p^{\varepsilon_1} \cdot (1-p)^{\varepsilon_2}$, which is equal to adding ε_1 relevant documents and ε_2 non-relevant documents sampled randomly for smoothing. The prior of σ is set to be $\exp(-v^2/(2\sigma^2))$, which is equal to adding v^2 to the sum of the square of the variance of relevant documents [3]. We set $\varepsilon_1 = 0.001, \varepsilon_2 = 0.001, v = 0.005$ in our experiment, which is equivalent to adding 0.001 relevant documents and 0.001 non-relevant documents, and adding 0.005 to the sum of the square of the variance of Guassian distribution.

For each query set, 4 runs were carried out. The first run (the *baseline run*) used the parameter estimation method described in [2]. The third run used our maximum likelihood estimation. Both runs stopped delivering documents when Δ is negative (Equation 12). This is especially common at the beginning of filtering, so a minimum delivery ratio was introduced to avoid this problem. If a profile has not achieved the minimum delivery ratio, its threshold will be decreased automatically. This corresponds to the second and fourth runs. In our experiments, the minimum delivery ratio was set to let approximately 10 documents be disseminated to each profile.

4. EXPERIMENTAL RESULTS

All of the experiments reported in this paper attempt to optimize the T9U′ Utility metric, which is consistent with recent TREC Filtering Track evaluation methodologies. Optimizing Utility can lead to higher Precision or Recall as a side-effect, but is not

---

[2] Because the beta distribution is a conjugate prior for binomial distributions, we use it as a prior to simplify calculations.

[3] This is a special case of inverse gamma distribution, which is the conjugate prior for the variance of the normal distribution.

guaranteed to do so. In some cases the best way to increase Utility is to increase Precision (higher threshold); in other cases the best way to increase Utility is to increase Recall (lower threshold). We report all three metrics to provide greater insight into the experimental results, but we consider only Utility, the metric being optimized, when evaluating the results.

The experimental results for the OHSUMED dataset indicate that neither algorithm works well without using a minimum delivery ratio (Table 1, columns 1 and 3). Especially at the early stage of filtering, with only about 2 documents delivered, it is very hard to estimate the score distribution correctly. When $\Delta < 0$ in Equation 9, the threshold is set too high to let any future documents be delivered, hence no learning takes place. Introducing a minimum delivery ratio (i.e., forcing the system to deliver at least a certain percentage of documents) guarantees enough training data to enable the system to recover automatically.

On the OHSUMED dataset, for both OHSU topics and MESH topics, using maximum likelihood estimation plus a minimum delivery ratio achieved the best result. Although profile updating was disabled while filtering, all of the runs on this dataset received a positive average utility. The results for Run 4 on OHSU topics are above average compared with other filtering systems in TREC9 adaptive filtering track. This indicates how effective the threshold-setting algorithm is, because the other filtering systems learned improved profiles while filtering, whereas all of our experiments used static filtering profiles.

Table 1: Utility of each filtering run: OHSUMED dataset.

		Run 1	**Run 2**	**Run 3**	**Run 4**
		Method in [2]	Method in [2] + min. delivery ratio	ML	ML + min. delivery ratio
OHSU topics	T9U′ utility	1.84	3.25	2.7	**8.17**
	Avg. docs. delivered per profile	3.83	9.65	5.73	**18.40**
	Precision	**0.37**	0.29	0.36	0.32
	Recall	0.036	0.080	0.052	**0.137**
MESH topics	T9U′ utility	1.89	4.28	2.44	**13.10**
	Avg. docs. delivered per profile	3.51	11.82	6.22	**27.91**
	Precision	**0.42**	0.39	0.40	0.34
	Recall	0.018	0.046	0.025	**0.068**

Table 2: Utility of each filtering run: FT dataset.

		Run 1	**Run 2**	**Run 3**	**Run 4**
		Method in [2]	Method in [2] + min. delivery Ratio	ML	ML + min. delivery ratio
Topics 351-400	T9U′ utility	**1.44**	-0.209	0.65	0.84
	Avg. docs. delivered per profile	9.58	10.44	9.05	**12.27**
	Precision	0.20	0.17	0.22	**0.26**
	Recall	0.161	0.167	0.15	**0.193**
	Number of Profiles with zero Precision	24	22	24	**10**

All four algorithms appear to perform about equally on FT data (Table 2). One difference between the FT dataset and the OHSUMED dataset is the average number of relevant documents per profile in the testing set (Section 3.2). Most of the FT filtering profiles are not good profiles, which means it is almost impossible to find a threshold that achieves a positive utility without profile updating. Indeed, the baseline method (Runs2) sets thresholds so high for some profiles that no relevant documents are delivered, thus getting zero Precision and Recall on those profiles. Our algorithm (Run 4) does not increase the threshold that much, which is why the baseline and unbiased ML methods appear similar according to the utility metric yet different according to the Precision and Recall metrics.

When we compared the utilities of each OHSU topic on Run 4 and Run 2 (Table 1), we found that our ML method did especially well on several good profiles where the score distribution of relevant documents and non-relevant documents are good from the system point of view (Figure 8). For other user profiles, the difference between our method and the baseline algorithm is not large. Comparison at the topic level and comparison between the OHSUMED and FT corpora suggests that our method works much better for good profiles, while its performance on other profiles are similar to the baseline method.

The baseline method delivered fewer OHSUMED documents, on average, than the unbiased ML method (Figure 9, Table 1, columns 2 and 4). The average number of documents delivered by the baseline method was close to the minimum delivery ratio, which is empirical justification for our previous analysis illustrated in Figure 5: the Gaussian mean estimated by the baseline method was higher than the actual mean, thus the threshold was set too high. The final results of the baseline method appear to be highly influenced by the minimum delivery ratio. This is not a problem for the maximum likelihood method, because thresholds based on unbiased estimates of score distributions are more accurate.

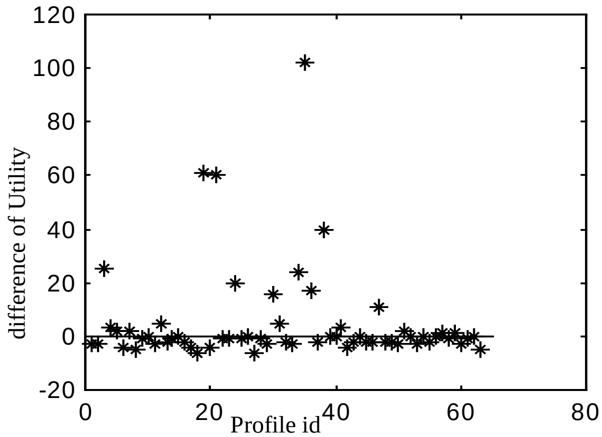

Figure 8. Utility differences on OHSU topics: Run4 - Run2. For some topics, Run 4 (ML method) does especially better than Run 2 (baseline method) (indicated by points high above the horizontal line), while they work similar on other topics.

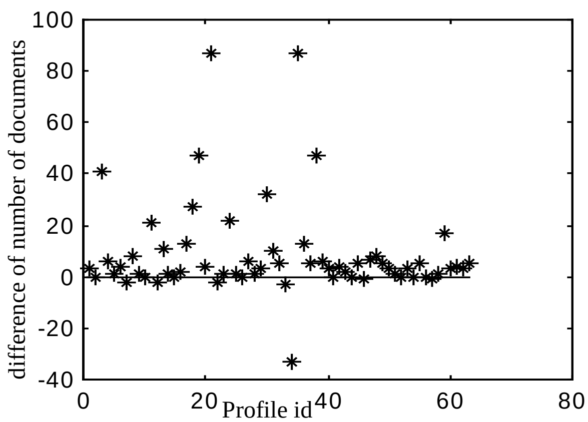

Figure 9. Difference in number of documents delivered for OHSU topics: Run4 - Run 2. For most of the topics, Run 4 (ML) delivered more documents (indicated by points above the horizontal line) than Run 2 (baseline method).

We have not focused on computational efficiency in this paper, but computational efficiency is important in environments where the filtering system must keep pace with a high-speed document stream. Although the mathematics may look complex, the unbiased ML algorithm is computationally efficient. It takes about 21 minutes ("wall-clock" time) to filter 4 years of OHSUMED data for 64 OHSU profiles (Table 1, Run 4) on a 500 MHz Intel Pentium III processor with 256 MB of memory. This time includes all aspects of document filtering, including the setting and updating of dissemination thresholds while filtering.

5. CONCLUSION AND FUTURE WORK

We have developed an effective algorithm for setting dissemination thresholds while filtering documents. Using a Gaussian distribution for the scores of relevant documents and an exponential distribution for the scores of non-relevant documents, a threshold that maximizes a given utility measure can be derived based on the parameters of the distribution.

Due to the fact that only relevance judgments of delivered documents are available, the method of estimating score density parameters proposed by [2] is biased and results in a threshold higher than optimal. We proposed an unbiased algorithm based on the maximum likelihood principle to jointly estimate (i) the parameters of the two density distributions, and (ii) the ratios of relevant and non-relevant documents. The new algorithm explicitly models the inherent bias in the filtering environment, where training data consists only of documents with scores above a threshold that changes constantly. We believe this is the first paper that solves the sample bias problem for information filtering. Our new method obtained significant improvements on the TREC-9 Filtering task.

In our experiments, filtering profiles were not updated while filtering. Although the effectiveness of the system is already very competitive, we believe combining threshold updating with profile updating will achieve better performance. One difficulty of combining our algorithm with profile updating is adjusting training data $D=\{(R_i, Score_i, \theta_i), i=1\cdots N\}$, especially θ_i, based on new profile terms and weights. Future research can be focused on this problem.

6. ACKNOWLEDGMENTS

We thank Wei Xu for discussions about some of the ideas in this paper. We also thank Paul Ogilvie for reviewing an earlier draft of this paper.

This material is based on work supported by Air Force Research Laboratory contract F30602-98-C-0110. Any opinions, findings, conclusions or recommendations expressed in this paper are the authors', and do not necessarily reflect those of the sponsors.

7. REFERENCES

[1] J. Allan. 1996. Incremental relevance feedback for information filtering. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 270-278. ACM.

[2] A. Arampatzis, J. Beney, C.H.A. Koster, and T.P. van der Weide. 2001. KUN on the TREC-9 Filtering Track: Incrementality, decay, and threshold optimization for adaptive filtering systems. In *Proceeding of Ninth Text REtrieval Conference (TREC-9).* National Institute of Standards and Technology, Special Publication.

[3] J. Broglio, J.P. Callan, W.B. Croft, and D.W. Nachbar. 1995. Document retrieval and routing using the INQUERY system. In *Proceeding of Third Text REtrieval Conference (TREC-3)*, pp. 29-38. National Institute of Standards and Technology, Special Publication 500-225.

[4] J. Callan. 1996. Document filtering with inference networks. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 262-269. ACM.

[5] J. Callan. 1998. Learning while filtering. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 224-231. ACM Press.

[6] D A. Hull, and S. E. Robertson. 1999. The TREC-8 Filtering Track final report. In *Proceeding of the Eighth Text REtrieval Conference (TREC-8)*, pp. 35-56. National Institute of Standards and Technology, Special Publication 500-246.

[7] Y.H. Kim, S.Y. Hahn, and B.T. Zhang. 2000. Text filtering by boosting Naive Bayes classifiers. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 168-175. ACM Press.

[8] R.D. Lyer, D.D. Lewis, R.E. Schapire, Y. Singer, and A. Singhal. 2000. Boosting for document routing. In *Proceedings of the Ninth International Conference on Information Knowledge Management (CIKM 2000)*, pp. 70-77. ACM Press.

[9] M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14 (3), pp. 130-137.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing*, pp. 420-425. Cambridge University Press.

[11] W. Hersh, C. Buckley, T. J. Leone and D. Hickam 1994. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 192-201. ACM Press

[12] R. Manmatha, T. Rath, and F. Feng. 2001. Modeling score distributions for combining the outputs of search engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* , New Orleans, LA. ACM.

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

[14] S. E. Robertson, and D. A. Hull. 2001. Guidelines for the TREC-9 Filtering Track. *Proceeding of Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, Special Publication.

[15] S.E. Robertson, and S. Walker. 2001. Microsoft Cambridge at TREC-9: Filtering track. In *Proceeding of Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, Special Publication.

[16] S.E. Robertson. In press. Threshold setting in adaptive filtering. *Journal of Documentation.*

[17] S. E. Robertson, S. Walker, M. M. Beaulieu, and M. Gatford, A. Payne. 1995. Okapi at TREC-4. In *Proceeding of Fourth Text REtrieval Conference (TREC-4)*. National Institute of Standards and Technology, Special Publication 500-236.

[18] R.E. Schapire, Y. Singer, and A. Singhal. 1998. Boosting and Rocchio applied to text filtering. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 215-223. ACM.

[19] T.Ault, and Y. Yang. 2001. kNN at TREC-9: A failure analysis. In *Proceeding of Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, Special Publication.

[20] C. Zhai, P. Jansen, and E. Stoica. 1998. Threshold calibration in CLARIT adaptive filtering. In *Proceeding of Seventh Text REtrieval Conference (TREC-7)*, pp. 149-156. National Institute of Standards and Technology, Special Publication 500-242.

[21] C. Zhai, P. Jansen, N. Roma, E. Stoica, and D.A. Evans. 1999. Optimization in CLARIT adaptive filtering. In *Proceeding of the Eighth Text REtrieval Conference (TREC-8)*, pp. 253-258. National Institute of Standards and Technology, Special Publication 500-246.

[22] Y. Zhang, and J. Callan. YFilter at TREC9. 2001. In *Proceeding of Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, Special Publication.

[23] http://wol.ra.phy.cam.ac.uk/mackay/c/macopt.html

[24] http://www.ccr.buffalo.edu/class-notes/hpc2-00/odes/node4.html