The Bias Problem and Language Models in Adaptive Filtering

Yi Zhang Jamie Callan

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15232, USA

{yiz, callan}@cs.cmu.edu

ABSTRACT

We used the YFILTER filtering system for experiments on updating profiles and setting thresholds. We developed a new method of using language models for updating profiles that is more focused on picking informative/discriminative words for query. The new method was compared with the well-known Rocchio algorithm. Dissemination thresholds were set based on maximum likelihood estimation that models and compensates for the sampling bias inherent in adaptive filtering. Our experimental results suggest that using what kind of distribution to model the scores of relevant and non- relevant documents is corpus dependant. The experimental results also show the sampling bias problem of training data while filtering makes the final profile learned biased.

1. INTRODUCTION

Given an initial description of a profile, a filtering system must sift through a stream of information and deliver the most relevant documents to the profile. Filtering is more like an online classification problem than a traditional search problem, because it is a binary decision process. Text classification algorithms, such as SVM, Rocchio, Boosting and Naive Bayes, can be applied to filtering, especially for batch filtering and routing. A common approach to learning profiles is to use an incremental version of the Rocchio algorithm [7]

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

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.

Language modeling has been applied to filtering track in TREC8 [4]. EM algorithm is used to find optimal parameters to maximize the likelihood of joint probability of relevant document and query. Our work introduces another way of using language modeling for the profile learning, which is also usin g EM but maximizing the likelihood of training data. We compares it with Rocchio in TREC10. Our result also shows the sampling bias problem (user only provide feedback for documents delivered) on learned profile terms and term weights/

2. SYSTEM DESCRIPTION

The YFILTER information filtering system we used is architecturally similar to InRoute [3]. It supports both structured Boolean and natural language descriptions of initial profiles. For natural language profiles, it can automatically update the profile according to user relevance feedback. YFILTER provides an option to use the INQUERY stopwords list and the Porter word-stemming algorithm [6].

3. PROFILE LEARNING

3.1 Initial Profile and Scoring Method

Each profile begins with topic words (usually 1-4 words) given by NIST, together with the training documents (usually 2). We used the BM25 tf.idf formula for scoring documents. Idf is initialized based on training data and updated over time as documents are filtered.

3.2 Profile Updating

At the very beginning when the number of training data is small, YFILTER has profile-specific anytime updating. That is, it updates a profile (threshold and scoring function) immediately whenever feedback, positive or negative, is available for that profile (Figure 1). After getting enough positive training examples (30 by default), the system begins to decrease the update frequency, updating the threshold only if:

- Current number of relevant documents delivered is 2*(number of relevant documents delivered at the last update), or
- Current number of non-relevant documents is 2*(number of non-relevant documents delivered at the last update), or
- Recent dangerous performance, which we define as 9 non-relevant documents delivered in a row.

3.2.1 Threshold Updating

We used the algorithm described in our SIGIR01 paper for threshold updating [9]. We provided a solution to optimize for F-beta measure based on our model. In case the system failed to find the optimal model, some heuristic were used to set the threshold.

3.2.2 Updating Terms and Term Weights

We tried the following two-term selection and term weight updating algorithms and compared their performance.

3.2.2.1 Language Model

Probabilistic language models, which are used widely in speech recognition and have shown promise for ad-hoc information retrieval. The strong theoretical foundation of language models enables us to build a variety of new capabilities. Current research on using language models for information retrieval tasks is focused on developing techniques similar to those used in speech recognition. However the differing requirements of speech recognition and information retrieval suggest that major adaptation of traditional approaches to language modeling is necessary to develop algorithms that will be highly accurate in the real world.

One research work in this direction is [4]. In their work, EM algorithm is used to find optimal relevance weights of each word that maximize the likelihood of joint probability of relevant document and query:

$$\prod_j P(d_j, q) = \prod_j P(d_j) P(q \mid d_j) \qquad (1)$$

In our work, we tried a different way of using language model. We propose a mixture of generative language models, which is more appropriate for the task of query expansion in information filtering and traditional ad-hoc retrieval task. As shown in Figure 1, we assume

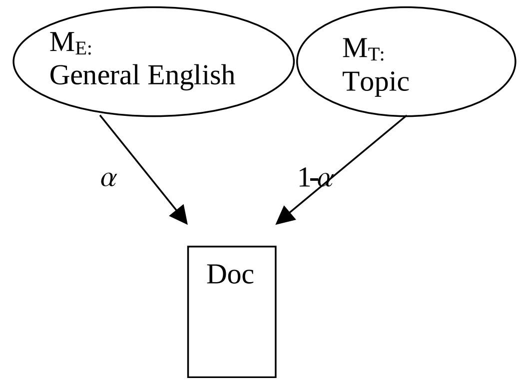

Figure 1 A mixture model to generate on topic documents

each relevant document is generated by a combination of two language models: A General English Model M_E, and a user-specific Topic Model M_T. For example, if the user is interested in “Star Wars”, in a relevant document we expect words such as “is” and “the” will come from the general English model, and words such as “star” and “wars” will come from the topic model.

When doing user profile updating for query expansion, the filtering system will focus on learning M_T to find words that are very informative for deciding whether the document is on topic or not. Given a fixed value of α (usually a very high value, such as 0.95), we can train M_E to maximize the likelihood of all documents processed, and train M_T using the EM algorithm to maximize the likelihood of all the relevant documents processed. A sketch of the training algorithm was given in Figure 2.

This new mixture model will pick words where P(w| relevant) / P(w | general English) is very high. Because the task of profile updating is to provide a profile that can separate relevant documents from non-relevant documents, we believe such words are more discriminative, and thus are good candidates for being added to user profiles. Similar techniques are developed for ad-hoc retrieval independently [10].

Although both algorithms are called “Language Model” approach, our work and [4] are very different in that 1) Our optimization goal is to maximize the likelihood of training data (which is a widely used method when using MLE), while they are maximizing something else (Equation 1) 2) The parameter our algorithm estimates is the Topic model, which is a multinomial distribution, while the parameters [4] are estimating is relevance weights.

(1) Calculate the General English language model:

For each word w_i

$$P(w_i \mid M_E) = \frac{tf\_i}{\sum_{j:\text{all words in corpus}} tf\_j}$$

(2) Calculate the Topic language model using the EM algorithm to maximize the likelihood of all relevant documents

a. Initialize Topic language model as uniform

$$P(w_i \mid M_T) = 1/n$$

where n is the number of unique words that appear in the relevant docuements

b. EM step:

Iterate the following steps until changes on $P(w_i \mid M_T)$ are small enough for that iteration:

For each words in relevant documents:

$$T\_tf_i = rtf_i * \frac{(1-\alpha)*P(w_i \mid M_T)}{(1-\alpha)*P(w_i \mid M_T)+\alpha*P(w_i \mid M_E)}$$

$$P(w_i \mid M_T) = \frac{T\_tf_i}{\sum_{i:\text{all words in relevant documents}} T\_tf_i}$$

Figure2 Training Algorithm for Language Model

3.2.2.2 Rocchio

The Rocchio algorithm used this year is very similar to the one we used last year in TREC9. Each time a positive relevance feedback arrives (including those in the training data), all words in that document are added to the profile's candidate list of terms. Then the weight of each word in the candidate list is calculated according to the incremental Rocchio formula:

$$Rocchio = \alpha \cdot w_q + \beta \cdot w_{rel} - \gamma \cdot w_{non-rel} \qquad (1)$$

where

$w_q(t)$: max(term frequency of word t in original topics, 0.5)

$$w_{rel}(t) = \frac{1}{|rel\_set(t)|+1} \sum_{d \in rel\_set(t)} tf\_bel_{t,d} * idf_{t,b} \qquad (2)$$

$$idf_{t,d} = \log((C_d + 0.5)/df_{t,d})/\log(C_d + 1.0) \qquad (3)$$

$$tf\_bel_{t,d} = tf_{t,d}/(tf_{t,d} + 0.5 + 1.5 \cdot (dl_d / avg\_dl_d)) \qquad (4)$$

The meanings of the above parameters are:

$tf_{t,d}$: Number of times term t occurs in document d

dl_d : Length of document d

C_d : Number of documents that arrived before document d

adv_dl_d : Average length of documents that arrived before document d

$rel\_set(t)$: Relevant documents after word t is added to the candidate list of the profile

α: 1; β: 3.5; γ: 2

In order to learn faster, β is set bigger than usual in the relevance feedback formula to emphasize the importance of relevant documents.

3.3 Hierarchical Category Structure

The filtering profiles correspond to Reuters categories, which are organized hierarchically. We assume that if a document belongs to a child category, it should also belong to the parent category. We used the following rules to take advantage of the hierarchical relationship between profiles when making the decision whether to deliver a document: If C_i is a child category of Cj, then:

- When a document d_t comes, we first consider whether it should be delivered to Cj, and if so, *then* consider whether it should be delivered to Ci.
- If the document d_t was delivered to Cj and the system received negative feedback, do not deliver d_t to Ci
- If d_t was not delivered to Cj previously, but was delivered to Ci and the system received a positive feedback, deliver d_t to Cj

By doing this, we get more training data for some of the profiles. For example, profiles 17, 34, and 45 get 8 instead of 2 relevant training documents to begin with, which is very helpful at the early stage.

The Reuters category assignments (i.e., the training data) are not consistent. Some documents are judged as relevant to a child category but non-relevant to the parent category. For example, documents 135639, 24269 26015 belong to R18 (DOMESTIC MARKETS) but do not belong to R17 (MARKETS/MARKETING). After reading those documents, we believe that they are in fact relevant to R17. It is well-known that human category assignments are not perfectly consistent, and any algorithm that uses them must compensate for noise in the training data. Using hierarchical structure of the categories helps to solve this problem to some extend.

4. ANALYSIS OF RESULTS

	Run 1	Run2	Run3	Run4
Profile Updating	Roc.	Roc.	LM	LM
Threshold Updating	ML	ML	ML	ML
Optimized for	T10SU	T10F	T10SU	T10F
T10SU	0.144	0.143	0.081	0.080
T10F	0.273	0.275	0.158	0.163
Profile>=Mean	55	41	32	21

Table 1: Submitted runs in TREC-9

We submitted 4 runs for the adaptive filtering task using Rocchio ("Roc") or language models ("LM") for profile updating, and our Maximum Likelihood Estimation for setting dissemination thresholds. The results are reported in Table 1. Compared with other groups, the results are not satisfying. Implementation errors were one cause, but we defer a discussion of their impact.

Figures 2 and 3 show the system performance over time for run 1. Precision and recall improve over time (Figure 3), but Utility decreases (Figure 4). This means the profiles (terms and term weighting) were improving as we got more training data while filtering, but unfortunately the threshold was set too high and got worse over time.

Despite the bugs and problems with the threshold, we can still analyse the performance of the Rocchio and language model methods of adding terms to profiles. According to Table 1, the simple Rocchio method works much better, which was a surprise. Our hypothesis was that the language model would work well; because the new language model approach does not need a stopwords list. However, the idf weights in the BM25 scoring method penalized words with high idf, thus allowing Rocchio to work well. Possible reason is BM25 scoring method used in Yfilter is good for Rocchio, but not good for language model. We probably should replace BM25 with KL divergence, which is a more natural scoring mechanism to measure distributional similarities.

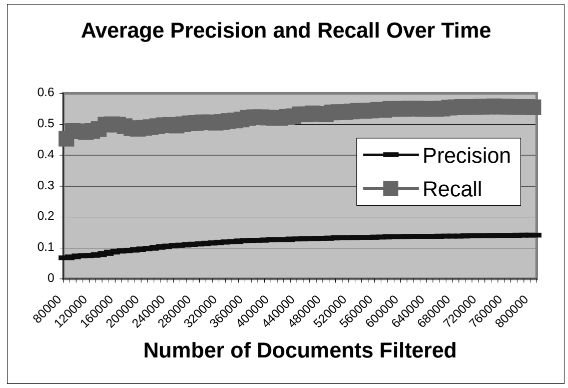

Figure 2 Filtering performances at different stages: Average Precision and Recall. (Run 1)

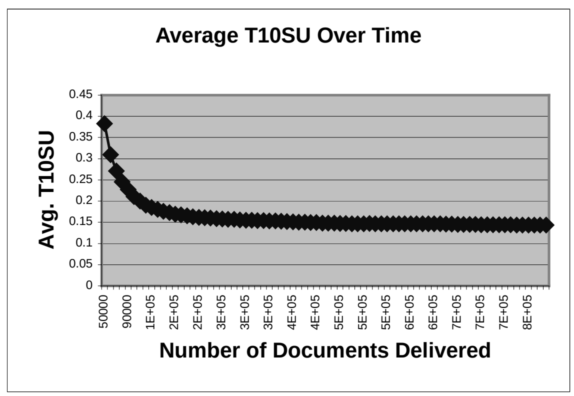

Figure 3 Filtering performances at different stages: T10SU Metric. (Run 1)

4.1 Score Density Distribution

We examined profile 77 on which our thresholding method did a very bad job. We used the learned final profile to score all of the documents in the corpus. Figure 4 shows the score density distribution of the relevant documents, which can be approximated by a normal distribution. Figure 5 shows the score density distribution of non-relevant documents that contain at least one profile term.

We found that using the exponential distribution to approximate the probability density function of non-relevant documents is problematic in this profile. A Beta distribution seems more appropriate. Since the exponential distribution is a special case of the beta distribution, using a beta distribution will also cover cases where the exponential distribution is right. Considering the maximum likelihood estimation method proposed in [9] does not require any specific distribution, we can plug the beta distribution into the general framework and find the optimal parameters. We have not yet implemented the algorithm to find the optimal beta distribution parameters. We simply observe that it may be a better approximation function than the exponential distribution proposed by [1] and used in our previous experiments [9]

4.2 Biased Training Problem for Profile Updating

We looked at the score distribution of profile 71 on which most of the systems did poorly. We fixed the profile terms and term weights (using the profile learned by the end) and scored all the documents in the corpus. We are surprised that the score density distribution of relevant documents looked more like an exponential distribution (Figure 6). Redrawing the score density distribution of the top scoring relevant documents for this profile (Figure 7) shows that the real distribution is actually like a mixture model of exponential and normal.

One possible explanation of this is the biased sampling problem [9]. In adaptive filtering, the user only provides feedback about documents delivered, so the training data is not sampled randomly, and the profile learned by the system is biased. The score density distribution of Profile 71 provides an extreme real case that illustrates this problem. Although we have proposed an algorithm to solve the sampling bias problem for threshold setting [9], we didn't develop a solution to solve the sampling bias problem when terms, term weights, and thresholds are all being adjusted simultaneously. One possible way is to deliver interesting "near miss" documents, so that the learning software gets a broader view of the surrounding information landscape. Theories in other research area, such as active learning (also known as experimental design) and reinforcement learning, are potentially useful considering the similarity of the tasks. Also, the bias problem for threshold learning and profile term updating are correlated and should be solved together. Another solution is to explicit modeling the sampling bias while profile term weights and threshold are changing, and more advanced analysis is needed.

4.3 Defects and explanation

Our results are disappointing on TREC10. There are several problems with the runs we submitted:

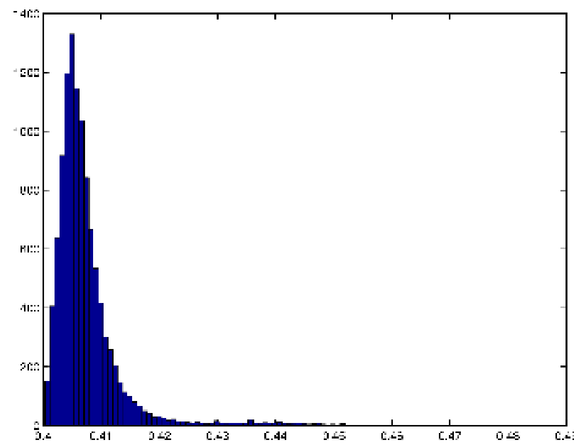

Figure 4 Score density distribution of relevant documents for profile 77.

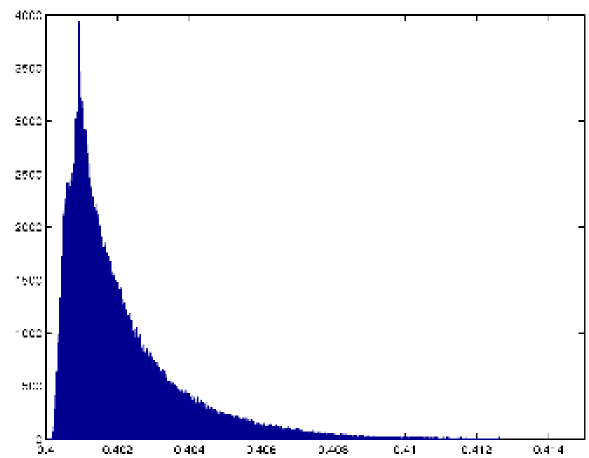

Figure 5 Score density distributions of non-relevant documents for profile 77.

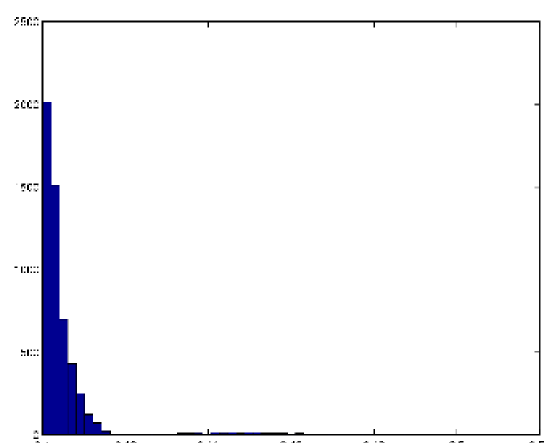

Figure 6 Score density distribution of relevant documents for profile 71.

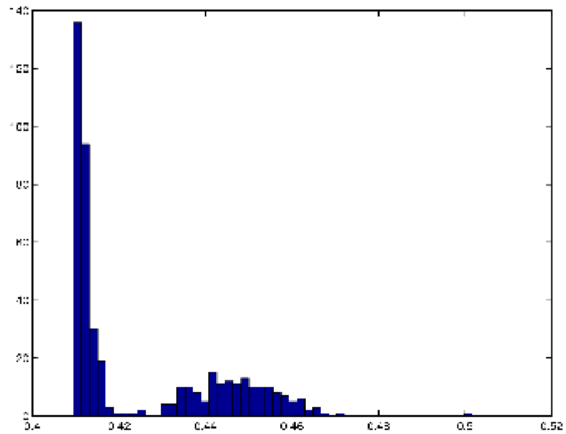

Figure 7 Score density distribution of top scoring relevant documents for profile 71.

(1) The methods deciding when to adjust terms and term weights was not integrated with threshold learning. Thus the system could change profiles without also adjusting thresholds to compensate for the changes in document scores.

(2) We used a heuristic rule to set thresholds when there is no solution based on maximum likelihood estimation. Unfortunately due to a programming error, we set the threshold too high (usually 1, which corresponded to delivering no document). Recovery speed was too slow.

(3) The words in the original profiles were stemmed before case conversion, but words in documents were stemmed after case conversion. The Porter stemmer is sensitive to case, so this difference produces inconsistent stemming in profiles and documents.

5. CONCLUSIONS

We tried a new profile-updating algorithm based on a mixture language model that we believe is more appropriate for the task of query expansion, and compared it with Rocchio. Compared with traditional language models, our new approach does select very discriminative words and requires no stop word list. The performance is encouraging. But what surprised us is how efficient (in terms of running time) and effective (in terms of performance) the old method of Rocchio is.

We noticed that the Beta distribution might be more appropriate for modeling the non-relevant document scores, although our previous experiments shows on some other dataset exponential distribution works well. We hypothesis that what kind of distribution to use is corpus/system dependant, although the Maximum Likelihood estimation we proposed does not require what kind of corpus to use, a real filtering should chose the right approximation function when applying our algorithm. We also noticed the effect of the sampling bias problem not only on profile threshold setting, but also on profile term weighting. Active learning and explicit modeling of the sampling bias while profile is changing are possible solutions for this problem.

6. ACKNOWLEDGEMENTS

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. REFERENCE

[1] Avi Arampatzis, A. Hameren. The Score-Distribution Threshold Optimization for Adaptive Binary Classification Task. *SIGIR01*

[2] J. Allan. 1996. Incremental Relevance Feedback for Information Filtering. *SIGIR96*

[3] J. Callan. 1998. Learning while Filtering. *SIGIR98*

[4] W. Kraaij, R. Pohlmann, D. Hiemstra. Twenty-One at TREC-8: using Language Technology for Information Retrieval. In *Proceeding of Eighth Text Retrieval Conference (TREC-8),* NIST.

[5] J. Lafferty, C. Zhai. 2001. Document Language Model, Query Models and Risk Minimization for Information Retrieval. *SIGIR01*

[6] M. F. Porter, 1980. An algorithm for suffix stripping.

[7] J. J. Rocchio. 1971. Relevance feedback in information retrieval in The SMART Retrieval System- Exp eriments in Automatic Document Processing, pages 313-323. Prentice Hall Inc.

[8] V. Rijsbergen. 1979 *Information Retrieval*

[9] Yi Zhang, J. Callan. Maximum Likelihood Estimation for Filtering Thresholds *ACM SIGIR01*

[10] Chengxiang Zhai, John Lafferty. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *Proceedings of Tenth International Conference on Information and Knowledge Management*