A Boosting Approach to Improving Pseudo-Relevance Feedback

Yuanhua Lv
Dept of Computer Science
University of Illinois at Urbana-Champaign
ylv2@uiuc.edu

ChengXiang Zhai
Dept of Computer Science
University of Illinois at Urbana-Champaign
czhai@cs.uiuc.edu

Wan Chen *
Wolfram Research, Inc.
100 Trade Center Drive
Champaign, IL
wanc@wolfram.com

ABSTRACT

Pseudo-relevance feedback has proven effective for improving the average retrieval performance. Unfortunately, many experiments have shown that although pseudo-relevance feedback helps many queries, it also often hurts many other queries, limiting its usefulness in real retrieval applications. Thus an important, yet difficult challenge is to improve the overall effectiveness of pseudo-relevance feedback without sacrificing the performance of individual queries too much. In this paper, we propose a novel learning algorithm, FeedbackBoost, based on the boosting framework to improve pseudo-relevance feedback through optimizing the combination of a set of basis feedback algorithms using a loss function defined to *directly measure both robustness and effectiveness*. FeedbackBoost can potentially accommodate many basis feedback methods as features in the model, making the proposed method a general optimization framework for pseudo-relevance feedback. As an application, we apply FeedbackBoost to improve pseudo feedback based on language models through combining different document weighting strategies. The experiment results demonstrate that FeedbackBoost can achieve better average precision and meanwhile dramatically reduce the number and magnitude of feedback failures as compared to three representative pseudo feedback methods and a standard learning to rank approach for pseudo feedback.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models, query formulation, relevance feedback

General Terms

Algorithms

Keywords

Pseudo-relevance feedback, FeedbackBoost, loss function, robustness, learning, optimization, boosting

*This work was done when Wan Chen was with University of Illinois at Urbana-Champaign.

1. INTRODUCTION

Pseudo-relevance feedback has proven effective for automatically improving the overall retrieval accuracy and average precision for informational queries in many experiments [24, 22, 25, 3, 23, 15, 31, 18]. The basic idea of pseudo feedback is to assume a certain number of top-ranked documents from an initial retrieval run to be relevant and extract useful information from these feedback documents to improve the original query.

However, although traditional pseudo feedback techniques generally improve retrieval performance (e.g., AP) on average, they are not robust in the sense that they tend to help some queries, but hurt other queries [10, 7], limiting its usefulness in real retrieval applications. Thus an important, yet difficult challenge is to *improve the overall effectiveness of pseudo-relevance feedback without sacrificing the performance of individual queries too much*. Although there has been a lot of work on pseudo feedback, little work has been devoted to address this issue, with only a few exceptions, e.g., [7]. In [7], the authors tried to reduce feedback failures in a constrained optimization approach. However their work was only able to optimize an objective function loosely related to the effectiveness and robustness of pseudo feedback.

In this paper, we propose a novel learning algorithm, FeedbackBoost, based on the boosting framework to improve pseudo-relevance feedback through combining a set of basis feedback algorithms optimally using a loss function defined to *directly measure both robustness and effectiveness*, which has not been achieved in any previous work on pseudo feedback. Specifically, like all other boosting algorithms [9, 26, 8, 30], FeedbackBoost iteratively selects and combines basis feedback methods. In each iteration, a basis feedback method is selected to improve those queries on which the already selected basis feedback methods perform poorly in terms of both effectiveness and robustness. At last, FeedbackBoost uses a linear combination of these basis feedback methods as its final feedback model.

There are several important differences between our work and previous work on improving pseudo feedback: (1) we cast the pseudo feedback problem as an optimization problem that can be solved in a supervised way; (2) we propose a novel objective function that directly measures the effectiveness and the number of failure cases of pseudo feedback; (3) FeedbackBoost can incorporate potentially many different basis feedback methods as features in the model, making it a general optimization framework for pseudo feedback; (4) FeedbackBoost does not introduce any extra parameter that needs to be manually tuned.

As an application, we apply FeedbackBoost to improve pseudo feedback based on language models through combining different document weighting strategies. One cause of the low robustness and effectiveness in pseudo feedback is that the feedback documents are simply assumed to be relevant whereas in reality, not all of them are relevant. Thus one way to improve pseudo feedback would be to assign appropriate weights to these documents. Indeed, our previous work [18] has already shown that with relevance-based document weighting, the relevance model [15] tends to be more robust and effective than alternative models for feedback with language models. In the existing work, however, such weighting is generally based on one heuristic or another, and is not optimized directly to improve feedback. Our main idea is to combine a variety of feedback methods each with a different strategy for document weighting under the framework of FeedbackBoost. Although we only try to leverage feedback methods with different document weighting methods in this work, the proposed boosting framework can potentially accommodate many other basis feedback methods.

We evaluate our method using two representative large test sets and compare FeedbackBoost with multiple baseline methods. The experiment results demonstrate that the proposed FeedbackBoost algorithm can improve average precision significantly and meanwhile reduce the number and magnitude of feedback failures dramatically as compared to two representative pseudo feedback methods based on language models, the mixture model and the relevance model. We also compare our algorithm with a recently proposed constrained optimization approach to robust feedback, and the results show that our method is more robust. In addition, we compare FeedbackBoost with a traditional learning to rank approach applied for pseudo feedback and observe that FeedbackBoost works clearly better.

2. RELATED WORK

Pseudo-relevance feedback has been shown to be effective for various retrieval models [24, 22, 25, 3, 23, 15, 31, 18]. In the vector space model, feedback is usually done by using the Rocchio algorithm [24]. The feedback method in classical probabilistic models is to select expansion terms primarily based on the Robertson/Sparck-Jones weight [22]. A number of model-based pseudo feedback techniques have been developed for the language modeling framework, including the relevance model [15] and the mixture-model feedback [31], which will be reviewed in Section 5.1.1 and 5.1.2 respectively. Most existing pseudo feedback algorithms aim at improving average precision alone but rarely address the robustness issue. In contrast, our work attempts to improve both average precision and robustness at the same time.

A few previous studies also attempted to improve the robustness of pseudo feedback [28, 27, 7]. Tao and Zhai [28] used a regularized EM algorithm to reduce the parameter sensitivity of the mixture-model feedback but did not minimize the feedback failures. Soskin et al. [27] leveraged multiple relevance models [15] in a heuristic unsupervised way to improve feedback performance. However, their method is not guaranteed to optimize the combination of feedback algorithms. Collins-Thompson [7] also tried to reduce feedback failures in an optimization framework. However this work was only able to optimize an objective function loosely related to the robustness of pseudo feedback. In contrast, we propose a general machine learning framework to directly optimize both the robustness and effectiveness of pseudo feedback, which can incorporate existing methods, such as [28], [27], and [7], as features. In this sense, our work offers a unified framework that can be used to potentially combine all the existing pseudo feedback methods.

Selective feedback [2] and adaptive feedback [17] are another stream of work to improve the robustness of pseudo feedback, where the idea is to disable query expansion if query expansion is predicted to be detrimental to retrieval [2] or to adaptively set the amount of query expansion in a per-query way [17]. However, these methods are not as general as our proposed framework. Besides, our work and the selective/adaptive feedback method are complementary in the following sense: our work can construct a strong ensemble feedback method, which can be used by selective/adaptive feedback to further improve its performance, while selective/adaptive feedback methods can be incorporated into our framework as features for boosting.

Recently, learning to rank [13, 8, 4, 30, 6, 33, 16] has attracted much attention in IR. Our work can also be regarded a novel use of machine learning to leverage multiple feedback-related features to improve ranking. However, a main difference of our work from traditional work on learning to rank is that we design a novel learning algorithm to directly optimize both robustness and effectiveness of pseudo feedback (novel objective function). Another difference is that most learning to rank work learns optimal ways to combine retrieval functions but fails to improve the query representation. Our work, however, uses machine learning to improve a content-based query representation. Therefore, our study is orthogonal to the existing learning to rank work, and existing learning to rank algorithms can be used to learn a retrieval function on the basis of our improved query representation to further improve retrieval performance.

Machine learning was also introduced to improve pseudo feedback through selecting good expansion terms [5] or good feedback documents [12]. However, neither work attempted to directly optimize robustness of pseudo feedback.

Boosting is a general method for improving the accuracy of supervised learning. The basic idea of boosting is to repeatedly construct "weak learners" by re-weighting training data and form an ensemble of weak learners so that the total performance of the ensemble is "boosted". Freund and Schapire have proposed the first and most popular boosting algorithm called AdaBoost for binary classification [9]. Extensions of boosting have been made to deal with the problems of multi-class classification [26], ranking [8, 30, 33], etc. Our work can be viewed as a novel extension of the boosting framework to improve pseudo-relevance feedback.

3. PROBLEM FORMULATION

Given a query q_i and a document collection C, a retrieval function F returns a ranked list of m documents $\mathbf{d} = \{d_1, \cdots, d_m\}$, where d_j denotes the j-th ranked document in the ranked list. In pseudo feedback, we assume the top-n documents are "relevant", and construct a feedback model $\phi_t(q_i, \mathbf{d}, n, C)$, or $\phi_t(q_i)$ for short, for query q_i by exploiting those assumed "relevant" documents. Here ϕ_t can be any pseudo feedback method that is able to output an improved query representation, and we call it a *weak* or *basis* feedback method; $\phi_t(q_i)$ is essentially an expanded representation of the original query q_i, and we call it a *weak* or *basis* feedback model for q_i. In general, the format of $\phi_t(q_i)$

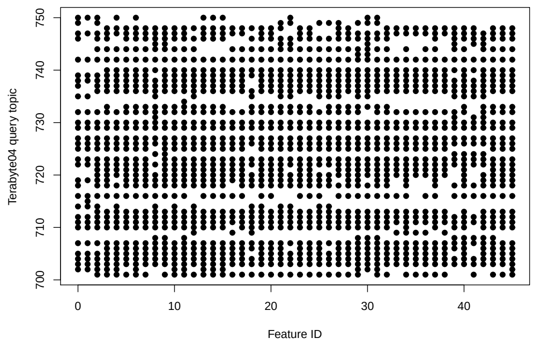

Figure 1: Plot of each query w.r.t. different weak feedback methods, where '•' indicates that the corresponding weak feedback improves performance.

would depend on the retrieval function F; for example, in the vector space model, $\phi_t(q_i)$ is represented as a vector of weighted terms, while in language modeling approaches, it is represented as a word distribution. We can use $\phi_t(q_i)$ as the new query and apply F to retrieve another ranked list of documents $\mathbf{d}'$.

Given a performance measure E and the relevance judgments set $J(q_i)$ for query q_i, we can compute the performance scores for the original query q_i and for the expanded query $\phi_t(q_i)$, which will be denoted as $E\left(F(q_i), J(q_i)\right)$ and $E\left(F(\phi_t(q_i)), J(q_i)\right)$, respectively, and will be represented as $E(q_i)$ and $E(\phi_t(q_i))$ in the rest of the paper for conciseness. We choose the widely accepted average precision (AP) as the performance measure E in this paper, though the proposed algorithm can in principle also work with other measures.

Although many pseudo feedback methods have been shown to improve the performance of a retrieval system on average, they all share a common deficiency, i.e., the average performance gain always comes inevitably at the cost of (sometimes significantly) degraded performance of some queries. That is, while on average, $\frac{1}{|Q|}\sum_{i=1}^{|Q|} E(\phi(q_i)) > \frac{1}{|Q|}\sum_{i=1}^{|Q|} E(q_i)$, it is almost always the case that for some queries, $E(\phi(q_j)) < E(q_j)$. Indeed, it has been a long-standing difficult challenge to improve the robustness of pseudo feedback so that we can improve average performance without sacrificing the performance of individual queries too much.

In this paper, we propose to use a learning method to address this problem. Specifically, given a query q_i, we assume there are a variety of basis feedback models $\phi_k(q_i)$ based on different feedback methods $\Phi = \{\phi_1, \cdots, \phi_m\}$, and our main idea is to combine a set of such basis feedback models $\phi_k(q_i)$ into a single feedback model $H(q_i)$ called the *final* or *combined* feedback model to reduce feedback failures:

$$H(q_i) = \sum_{k=1}^{t} \alpha_k \phi_k(q_i) \tag{1}$$

A linear combination is chosen because the final feedback model $H(q_i)$ should be in the same format as that of each basis model $\phi_k(q_i)$; that is, if $\phi_k(q_i)$ is a vector of weighted terms, so is $H(q_i)$, while if $\phi_k(q_i)$ is a language model, $H(q_i)$ should also be a language model by normalizing the learned α_k $(k = 1, \cdots, t)$ to make them sum to 1.

Our main motivation of combining multiple feedback methods is that different basis feedback methods often have relative strengths for different query topics and thus are complementary to each other. To illustrate it, we examine 46 basis feedback methods constructed using methods described in Section 5. The results are presented in Figure 1. We can see that many feedback methods indeed have relative strengths on different queries (e.g., topic 708 and 709). It seems there are only 3 very hard queries that no feedback method can help, while for other queries, there are always some successful feedback methods. Thus, constructing an ensemble feedback method H would be an effective strategy to reduce feedback loss. For example, suppose we have two weak feedback methods ϕ_1 and ϕ_2; ϕ_1 improves 0.2 and -0.1 (i.e., decreases by 0.1) on q_1 and q_2 respectively, while ϕ_2 improves -0.1 and 0.2 on q_1 and q_2 respectively. So the fbloss scores are 0.5 for both ϕ_1 and ϕ_2 on these two queries. However, an ensemble method $\alpha_1\phi_1 + \alpha_2\phi_2$ would probably have 0 fbloss while still achieving better or comparable average improvement if the combination coefficients α_1 and α_2 are chosen appropriately.

Our goal is to minimize the number of query instances for which the retrieval performance is decreased by the final feedback model $H(q_i)$ as compared to the original query q_i. The learning algorithm that we study attempts to find an H with a small number of query failures, a quality called the *feedback loss* and denoted by $\text{fbloss}_D(H)$. Formally,

$$\text{fbloss}_D(H) = \sum_{i=1}^{|Q|} D(q_i) \cdot I\{E(H(q_i)) < E(q_i)\} \tag{2}$$

Here and throughout this paper, the indicator function $I\{\pi\}$ is defined to be 1 if the predicate π holds and 0 otherwise. $D(q_i)$ is the weight of q_i, and we set it initially to uniform, i.e., $D(q_i) = 1/|Q|$ so that all queries are equally important; later the query weights would be updated iteratively in a boosting framework [9] so that queries that do not perform well would contribute more to the loss function more. We will also show later in Section 4.1 that the feedback loss can be bounded by the degradation of retrieval precision, which means that the feedback loss can actually measure both feedback failures and retrieval effectiveness, which is necessary in order to ensure both robustness and effectiveness.

In the following section, we will discuss how to optimize the combination of weak feedback methods so as to minimize fbloss on some training data, formally,

$$\arg\min_{\{\alpha_{1:t}\}} \sum_{i=1}^{|Q|} D(q_i) \cdot I\left\{E(\sum_{k=1}^{t} \alpha_k \cdot \phi_k(q_i)) < E(q_i)\right\} \tag{3}$$

4. A BOOSTING APPROACH TO IMPROVING PSEUDO FEEDBACK

With only a few basis feedback methods, it is possible to optimize their combination through manual parameter tuning. However, manual tuning is infeasible if there are many basis feedback methods as is often the case. Inspired by the AdaBoost [9], we devise a boosting approach, referred to as "FeedbackBoost", to solve this problem.

4.1 Minimizing Feedback Loss via Forward Stagewise Additive Modeling

Ideally we want to minimize the feedback loss on the training data as shown in Equation 3. However, it is difficult to solve this optimization problem because the combination is inside a retrieval function and not differentiable. To simplify this problem, we make an assumption that the feedback loss of the combined feedback H is less than or equal to that of a linear performance combination of the corresponding basis feedback methods. Formally,

$$\begin{aligned}&\sum_{i=1}^{|Q|} D(q_i)\cdot I\left\{E(\sum_{k=1}^{t}\alpha_k\cdot\phi_k(q_i)) < E(q_i)\right\}\\ \le &\sum_{i=1}^{|Q|} D(q_i)\cdot I\left\{\left[\sum_{k=1}^{t}\frac{\alpha_k}{\sum_{j=1}^{t}\alpha_j}E(\phi_k(q_i))\right] < E(q_i)\right\}\end{aligned} \tag{4}$$

Although we have not found a theoretical proof for the above inequality, we can empirically guarantee this assumption to be true in our algorithms since we can automatically adjust the algorithms to make sure this assumption holds (as shown in the step 5 of the pseudo code of the algorithm). In practice, this assumption turns out to work well. One possible explanation is that different feedback models often complement to each other, so the performance of a mixture model with reasonable coefficients is often better than the performance of any single model and thus also better than the weighted average performance of these single models. An example is that an appropriate interpolation of a pseudo feedback model and the original query model usually works better than either single model [31, 1, 18]. Thus we will minimize the following upper bound of the feedback loss.

$$\arg\min_{\{\alpha_{1:t}\}}\sum_{i=1}^{|Q|} D(q_i) I\left\{\sum_{k=1}^{t}\alpha_k E(\phi_k(q_i)) < \sum_{k=1}^{t}\alpha_k E(q_i)\right\} \tag{5}$$

However the above optimization problem is still hard to handle, because the indicator function I is non-continuous. Fortunately, we know that $I\{x < y\} \le e^{y-x}$ for all real x and y, so, instead of solving Equation 5 directly, we can alternatively minimize its upper bound below, which is essentially a measure of retrieval performance degradation.

$$\arg\min_{\{\alpha_{1:t}\}}\sum_{i=1}^{|Q|} D(q_i)\exp\left(\sum_{k=1}^{t}\alpha_k E(q_i) - \sum_{k=1}^{t}\alpha_k E(\phi_k(q_i))\right) \tag{6}$$

Now we can address this problem using forward stagewise additive modeling, which is an effective strategy to find an approximate solution to an optimization problem through sequentially adding new basis method without adjusting those already selected methods [11]. By forward stagewise additive modeling, the above formula can be expressed as follows where we would iteratively choose α_t and ϕ_t, as well as adjust $D_t(q_i)$:

$$\arg\min_{\{\alpha_t,\phi_t\}}\sum_{i=1}^{|Q|} D_t(q_i)\cdot\exp\left(\alpha_t\cdot[E(q_i) - E(\phi_t(q_i))]\right) \tag{7}$$

where

$$D_t(q_i) \propto \frac{D_{t-1}(q_i)\cdot\exp\left(\alpha_{t-1}\cdot[E(q_i) - E(\phi_{t-1}(q_i))]\right)}{Z_t} \tag{8}$$

Here, Z_t is a normalization factor to make D_t a distribution; such a normalization operation does not affect the choosing of α_t and ϕ_t, since D_t depends neither on α_t nor ϕ_t in the forward stagewise additive modeling [11]. D_t is defined in a recursive way, where $D_1(q_i) = D(q_i) = 1/|Q|$ is the initial weight for query q_i. After T iterations, we can obtain the desired combined feedback model as $H(q_i) = \sum_{t=1}^{T}\alpha_t\phi_t$.

Algorithm 1 The FeedbackBoost algorithm

```
Input:
    Query set $\{q_i, J(q_i)\}_{i=1}^{|Q|}$, retrieval model $F$, retrieval performance measure $E$, and the number of iterations $T$;
    A set of basis feedback methods $\Phi = \{\phi_1, \cdots, \phi_m\}$;
    Initial distribution $D_1$ over training queries: $D_1(i) = 1/|Q|$;
Output:
1: for $t = 1, 2, \cdots, T$ do
2:     Select basis feedback method $\phi_t$ with weighted distribution $D_t$ on training queries using Formula 14; if there is no $\phi_t$ selected, break;
3:     Compute $\mathrm{Eloss}_t(\phi_t)$ using Formula 9.
4:     Choose $\alpha_t$ based on Formula 12 and $\mathrm{Eloss}_t(\phi_t)$.
5:     While the inequality in Formula 4 does not hold, goto step 2 and choose the next optimal basis feedback as $\phi_t$; if there is no $\phi_t$ that satisfies the inequality, break;
6:     Update $D_{t+1}$ using Formula 8;
7: end for
8: Output the final feedback: $H = \sum_{t=1}^{T}\alpha_t\phi_t$;
```

4.2 FeedbackBoost

FeedbackBoost is designed to find a solution to the optimization problem in Formula 7 using a boosting approach. Specifically, like all other boosting algorithms, FeedbackBoost operates in rounds. At each round, we calculate a distribution of weights over training queries. In fact, D_t can be regarded as a weight that is applied to each query after $t-1$ iterations. From Equation 8, we can see that D_t will put more weights on queries that are hurt more by previously selected basis feedback methods.

We next select a basis feedback method ϕ_t that works well on those highly-weighted queries, and the selection of this basis feedback method is to optimize an objective function that is defined to directly measure the feedback loss. Specifically, we define the weighted performance degradation of ϕ at iteration t as

$$\mathrm{Eloss}_t(\phi) = \sum_{i=1}^{|Q|} D_t(q_i)\cdot(E(q_i) - E(\phi(q_i))) \tag{9}$$

And the feedback method ϕ_t with the minimum Eloss_t will be selected. Theoretical analysis is provided as follows:

The performance measure E, e.g., AP in this paper, is usually within range $[0, 1]$. Even a measure is beyond this range, we can still normalize it to $[0, 1]$. Therefore, for any basis feedback method ϕ_t, we have the performance degradation $E(q_i) - E(\phi_t(q_i)) \in [-1, 1]$. By the convexity of $e^{\alpha x}$ as a function of x when $x \in [-1, 1]$ [8], we thus have

$$\begin{aligned}\exp\left(\alpha_t[E(q_i) - E(\phi_t(q_i))]\right) \le &\left(\frac{1 + E(q_i) - E(\phi_t(q_i))}{2}\right)\exp(\alpha_t)\\ &+\left(\frac{1 - E(q_i) + E(\phi_t(q_i))}{2}\right)\exp(-\alpha_t)\end{aligned} \tag{10}$$

So Equation 7 can be approximated by

$$\arg\min_{\{\alpha_t,\phi_t\}}\frac{1 + \mathrm{Eloss}_t(\phi_t)}{2}\exp(\alpha_t) + \frac{1 - \mathrm{Eloss}_t(\phi_t)}{2}\exp(-\alpha_t) \tag{11}$$

We can easily minimize this equation by setting

$$\alpha_t^* = \frac{1}{2}\log\frac{1 - \mathrm{Eloss}_t(\phi_t)}{1 + \mathrm{Eloss}_t(\phi_t)} \tag{12}$$

which indeed makes sense since it suggests that a ϕ_t with less performance degradation will receive a larger weight α_t.

Then, by plugging Equation 12 into 11, we obtain the following optimal basis feedback.

$$\phi_t^* = \arg\min_{\{\phi_t\}} \sqrt{(1 - \text{Eloss}_t(\phi_t))(1 + \text{Eloss}_t(\phi_t))} \quad (13)$$

We can see that Equation 13 is minimized when $\text{Eloss}_t(\phi_t)$ is close to 1 or -1. With respect to the former case, we would choose a ϕ_t with the largest weighted performance degradation, and α_t will be negative, which, however, does not make sense: if a pseudo feedback ϕ_t leads to large performance degradation, a negative α_t may not make ϕ_t work well. Therefore, $\text{Eloss}_t(\phi_t)$ should be negative to keep the coefficient α_t positive. So we choose a basis pseudo feedback ϕ_t with the smallest negative Eloss_t so far.

$$\phi_t^* = \arg\min_{\{\phi_t\}} \{\text{Eloss}_t(\phi_t)\} \quad \text{subject to } \text{Eloss}_t(\phi_t) < 0 \quad (14)$$

4.3 Algorithm Summary

To summarize, FeedbackBoost works as follows. The input to the algorithm includes a set of queries and relevance judgments $\{q_i, J(q_i)\}_{i=1}^{|Q|}$, a set of basis feedback methods $\Phi = \{\phi_1, \cdots, \phi_m\}$, a document collection C, a retrieval function F, a retrieval performance measure E, and the iteration number T. FeedbackBoost works in an iterative way: during each iteration t, a basis feedback method ϕ_t is chosen based on its performance on training data with weight distribution D_t, i.e., $\text{Eloss}_t(\phi_t)$. Also the coefficient α_t of ϕ_t is calculated based on $\text{Eloss}_t(\phi_t)$. After that, the query weight distribution D_t is updated by increasing weights on queries for which ϕ_t performs poorly, leading to a new distribution D_{t+1}; D_{t+1} will be used in the next iteration to select ϕ_{t+1} and α_{t+1}. At last, the final feedback model H is created by linearly combining all the selected basis feedback methods. A sketch of the algorithm flow is shown in Algorithm 1.

5. APPLICATION OF FEEDBACKBOOST TO LANGUAGE MODELS

In order to apply FeedbackBoost, the main task is to design appropriate basis feedback methods $\{\phi\}$. A basis feedback method ϕ_k generally consists of three components: a weighting function h_k to assign weights to feedback documents, a weighting function g_k to calculate the importance of different expansion terms, and the retrieval model F to decide the representation format of the feedback model. Formally, $\phi_k = f(h_k, g_k, F)$. Given a retrieval model F, one feedback method differs from others often because it uses different document and/or term weighting functions [18]. We can thus naturally construct many basis feedback methods by varying the document and/or term weighting functions.

5.1 Basis Pseudo Feedback Methods based on Language Models

As a specific application, here we discuss how we can apply FeedbackBoost to improve pseudo feedback under the language modeling framework through combining different document weighting strategies. This application is especially interesting because (1) language models deliver state of the art retrieval performance [21, 14]; (2) feedback document weighting has been shown to be a critical factor affecting robustness and effectiveness of pseudo feedback methods [18]. However, our methodology could be applicable to other retrieval models and to optimizing term weighting methods as well, which we leave as future work.

We use the KL-divergence retrieval method [14] as our retrieval model (i.e., F), which scores a document d with respect to a query q by computing the negative KL divergence between the query and the document language model:

$$S(q, d) = -\sum_{w \in q} P(w|q) \log \frac{P(w|q)}{P(w|d)} \quad (15)$$

Two important instantiations of basis pseudo feedback methods in language models are the relevance model [15] and the mixture model [31], which are among the most effective and robust feedback techniques based on language models [18].

5.1.1 *Relevance Model*

The relevance model ϕ_r essentially uses the query likelihood as the weight for document d and takes an average of the probability of word w given by each document language model. Formally, let Θ represent the set of smoothed document models in the pseudo feedback collection $\Omega = \{d_1, \cdots, d_n\}$. The formula of the relevance model is:

$$P(w|\phi_r(q)) \propto \sum_{\theta_d \in \Theta} P(w|\theta_d) \prod_{w' \in q} P(w'|\theta_d) \quad (16)$$

Let's denote the original query model as $P(w|q)$. The relevance model $P(w|\phi_r(q))$ can be interpolated with the original query model $P(w|q)$ to improve performance using a interpolation coefficient α. In this paper, we will use the following interpolated model $P(w|\theta_q)$ as the new query model, which is often called RM3 [1]:

$$P(w|\theta_q) = (1 - \alpha)P(w|q) + \alpha P(w|\phi_r(q)) \quad (17)$$

In Equation 16, $h_r(d) = \prod_{w' \in q} P(w'|\theta_d)$ is the query likelihood score of document d, serving for document weighting, and $g_r(w, d) = P(w|\theta_d)$ works for term weighting. If we instantiate the document weighting strategy in a different way, e.g., h'_r, whereas the term weighting strategy is fixed to g_r, it will lead to a different "relevance model" ϕ'_r. Formally

$$P(w|\phi'_r(q)) \propto \sum_{\theta_d \in \Theta} P(w|\theta_d) \cdot h'_r(d) \quad (18)$$

which can also be used for feedback after a similar interpolation. Following this way, we can construct a set of relevance-model style basis feedback methods by varying their document weighting strategies.

5.1.2 *Mixture Model*

In the simple two-component mixture model (SMM) ϕ_m, the words in Ω are assumed to be drawn from two models: (1) background model $P(w|C)$ and (2) topic model $P(w|\phi_m(q))$. Thus the log-likelihood for the entire set of feedback documents is:

$$\log P(\Omega|\phi_m(q)) = \sum_{w \in V} c(w, \Omega) \log((1 - \lambda)P(w|\phi_m(q)) + \lambda P(w|C)) \quad (19)$$

where V is the word vocabulary, $c(w, \Omega)$ is the count of word w in feedback document set Ω, and $\lambda \in [0, 1]$ is the mixture parameter. The estimate of $\phi_m(q)$ can be computed using the EM algorithm to maximize the log-likelihood. Finally, $\phi_m(q)$ is also interpolated with the original query model $P(w|q)$ to update the query model with a coefficient α. We notice in Formula 19 that

$$c(w, \Omega) = \sum_{d \in \Omega} |d| \cdot P(w|d) \quad (20)$$

It means that the document length $|d|$ is used as a weight of document d (i.e., $h_m(d) = |d|$) to sum over the term evidence from each feedback document. As in the case of the relevance model, we can also use any other document weighting method h'_m to replace h_m while keeping g_m the same, which will lead to a new family of mixture-model style basis feedback method ϕ'_m.

5.2 Document Weighting Strategies

We next introduce a set of document weighting strategies $\{h_t\}$. As we have discussed, each of them can be plugged into the relevance model (Section 5.1.1) or the mixture model (Section 5.1.2), leading to a new basis feedback method ϕ_t.

Relevance Score: Relevance score is shown to be a critical factor for feedback document weighting in a recent work [18]. We explore the use of query likelihood [21] h_1 and BM25 score [23] h_2 for document weighting: w.r.t. the former, the Dirichlet smoothing method [32] with $\mu = 1,000$ is used to smooth the document language model; w.r.t. the latter, we fix $k_1 = 1.2$, $b = 0.5$, and $k3 = 1,000$.

Document Novelty: We estimate a novelty score for each document to reward novel information. Three different methods are proposed: (1) The distance between the centroid of all feedback documents $h_3(d_i) = 1 - \text{cosim}(\vec{d}_i, \frac{1}{k}\sum_{j=1}^{k}\vec{d}_j)$, where 'cosim' stands for cosine similarity; (2) The distance between the centroid of all feedback documents ranked before the document $h_4(d_i) = 1 - \text{cosim}(\vec{d}_i, \frac{1}{i-1}\sum_{j=1}^{i-1}\vec{d}_j)$. (3) The distance between the most similar document ranked before the document: $h_5(d_i) = 1 - \max_{j=1}^{i-1}\{\text{cosim}(\vec{d}_i, \vec{d}_j)\}$.

Query Term Proximity: Query term proximity has been largely ignored in traditional retrieval models [23, 21]. We use the recently proposed positional language model [19] and the minimum pair distance [29] to capture term proximity for improving document weighting: (1) we compute a positional query likelihood score based on the "best-matching" position [19], i.e., $h_6(d) = \max_{j=1}^{|d_i|}\prod_{w\in q} P(w|d,j)^{c(w,q)}$, where $c(w, q)$ is the count of w in q, and we follow work [17] to estimate the positional language model $P(w|d, j)$; (2) we use the normalized minimum pair-wise distance proposed in [29] by setting $\alpha = 1$ which prevents negative document weights, i.e., $h_7(d) = \log(\alpha + \exp(-\delta(q, d)))$.

Document Length: Though the heuristic of document length normalization has been incorporated into the relevance scores [23, 32], we still list it here because it was explicitly used in some existing pseudo feedback methods [18]: (1) raw document length: $h_8(d) = |d|$; (2) reciprocal of the raw document length: $h_9(d) = 1/h_8(d)$; (3) Dirichlet document length: $h_{10}(d) = |d|/(|d| + \mu)$, where we set $\mu = 1,000$; (4) reciprocal of the Dirichlet document length: $h_{11}(d) = 1/h_{10}(d)$.

Besides the above basic document weighting methods, for each h_t, we also introduce some of their variations as our document weighting, including $\exp(h_t)$, $(h_t)^2$, and $\sqrt{h_t}$. Also we include $\log(h_2)$ and $\log(h_8)$, since the values of h_2 and h_8 are usually larger than 1.0 for top-ranked documents. Overall, there are 46 methods in total, all of which are normalized to sum to 1.0 for each query.

5.3 Implementation Details

We next apply the FeedbackBoost algorithm to learn an ensemble feedback method. We denote $E_d(\phi(q))$ as the score of document d with respect to a basis feedback method ϕ on query q. In fact, with the KL-divergence retrieval method (Equation 15), we only need to retrieve and score documents once for each basis feedback method. Then during the training process, if we need to score a document d using any combined feedback $H' = \sum_{k=1}^{t}\alpha_k\phi_k$, we can do it efficiently by linearly combining the scores of the corresponding basis feedback methods.

$$\begin{aligned} E_d(H'(q)) &\propto \sum_{w\in q} P\left(w|\sum_k^t \frac{\alpha_k}{\sum_j^t \alpha_j}\phi_k(q)\right)\log P(w|d) \\ &= \sum_{w\in q}\left[\sum_{k=1}^{t}\frac{\alpha_k}{\sum_{j=1}^{t}\alpha_j}P(w|\phi_k(q))\right]\log P(w|d) \qquad (21) \\ &\propto \sum_{k=1}^{t}\alpha_k E_d(\phi_k(q)) \end{aligned}$$

So training FeedbackBoost would be as efficient as training general AdaBoost algorithm [9].

Besides, during the testing phase, $H(q)$ can also be estimated efficiently. For example, for the relevance model, we can plug the ensemble document weighting method used in $H(q)$ into Formula 18 to replace $h'_r(d)$ and estimate $H(q)$ directly; for the mixture model style $H(q)$, we can also replace $|d|$ with the combined document weighting methods in Formula 20, which does not affect feedback performance in the experiments. Hence, the efficiency of $H(q)$ for pseudo feedback would be comparable to that of basis feedback algorithms, which is also confirmed empirically.

6. EXPERIMENTAL SETUP

We evaluate our method using the Terabyte Web dataset and a large news dataset Robust04. Only title portions of the topics are taken as queries. For each dataset, we split the available topics into training, validate and test sets, where the training set is used solely for training the algorithms, the validate set is used for tuning the number of iterations T, and the test set is used for evaluation purposes. Table 1 shows some document set statistics. The preprocessing of the collections includes stemming using the Porter algorithm and stopword removal using a standard InQuery stoplist.

We train two FeedbackBoost models: in the first one, each basis feedback method implements a different document weighting strategy proposed in Section 5.2, while all of them use the same mixture-model style term weighting, and thus we refer to it as **BoostMM**; in the second one, each basis feedback method also implements a different document weighting strategy but all of them share the relevance-model style term weighting, thus the name **BoostRM**. That is, we parameterize Φ in different ways for BoostRM and BoostMM. This design allows us to meaningfully compare BoostMM and BoostRM with the corresponding two baseline feedback methods (i.e., SMM and RM3) and the basic query language model without feedback (labeled as "NoFB"). This set of experiments are mainly to compare FeedbackBoost with traditional pseudo feedback methods. A main hypothesis we would like to test is whether BoostMM and BoostRM are indeed more robust than the corresponding baseline SMM and RM3.

Furthermore, we also compare FeedbackBoost with two other lines of baseline methods. In the first line, we compare FeedbackBoost with another strong baseline representing a recent work on improving robustness of pseudo feedback, i.e., the **REXP-FB** method [7]. In the second line, we are interested in knowing if an existing learning to rank approach can also improve both robustness and effectiveness as much as FeedbackBoost does. So we introduce yet another baseline **AdaRank** [30], which performed well [16]. AdaRank

Collection	Description	#Docs	Training Topics	Validate Topics	Test Topics
Robust04	TREC disk 4&5 (minus CR)	528,155	301-450	601-650	651-700
Terabyte	2004 crawl of .gov domain	25,205,179	701-750	751-800	801-850

Table 1: Overview of TREC collections and topics

BoostMM Versus SMM

Collection		Metric	NoFB	fbDocCount = 20		fbDocCount = 50	
				SMM	BoostMM	SMM	BoostMM
Validate	Robust04	MAP	0.2850	0.3215*	**0.3447**$^{*+}$	0.2973	**0.3468**$^{*+}$
		Pr@20	0.3310	0.3610	**0.3790**	**0.3850**	**0.3850**
		RI	n/a	0.3600	**0.6400** (+77.8%)	0.0400	**0.6000** (+1400%)
		APloss	n/a	0.6191	**0.2877** (−53.5%)	1.3730	**0.2815** (−79.5%)
	Terabyte	MAP	0.3076	0.3477*	**0.3701**$^{*+}$	0.3445*	**0.3669**$^{*+}$
		Pr@20	0.5410	0.5570	**0.5970**	0.5540	**0.5820**
		RI	n/a	0.4400	**0.7200** (+63.6%)	0.3200	**0.6000** (+87.5%)
		APloss	n/a	0.6036	**0.1667** (−72.4%)	0.6963	**0.2433** (−65.1%)
Test	Robust04	MAP	0.2930	0.3265*	**0.3511**$^{*+}$	0.3067	**0.3453**$^{*+}$
		Pr@20	0.3796	0.4041	**0.4143**	0.3867	**0.4082**
		RI	n/a	0.3878	**0.5102** (+31.6%)	0.2653	**0.4286** (+61.6%)
		APloss	n/a	0.7108	**0.3493** (−50.9%)	1.3429	**0.4231** (−68.5%)
	Terabyte	MAP	0.3012	0.3061	**0.3331**$^{*+}$	0.3067	**0.3298**$^{*+}$
		Pr@20	0.4878	0.4765	**0.5255**	0.4663	**0.5112**
		RI	n/a	0.1429	**0.5102** (+257%)	0.1429	**0.5102** (+257%)
		APloss	n/a	0.6438	**0.1499** (−76.7%)	0.7546	**0.2511** (−66.7%)

BoostRM Versus RM3

Collection		Metric	NoFB	fbDocCount = 20		fbDocCount = 50	
				RM3	BoostRM	RM3	BoostRM
Validate	Robust04	MAP	0.2850	0.3431*	**0.3450***	0.3430*	**0.3490**$^{*+}$
		Pr@20	0.3310	**0.3840**	0.3800	0.3800	**0.3880**
		RI	n/a	0.2800	**0.4800** (+71.4%)	0.3600	**0.5200** (+44.4%)
		APloss	n/a	0.7084	**0.5056** (−28.6%)	0.6421	**0.4155** (−35.3%)
	Terabyte	MAP	0.3076	0.3471*	**0.3661**$^{*+}$	0.3466*	**0.3628**$^{*+}$
		Pr@20	0.5410	0.5840	**0.5890**	**0.5770**	0.5740
		RI	n/a	0.5200	**0.6800** (+30.8%)	0.4800	**0.5600** (+16.7%)
		APloss	n/a	0.9042	**0.2946** (−67.4%)	0.9458	**0.4247** (−55.1%)
Test	Robust04	MAP	0.2930	0.3483*	**0.3537**$^{*+}$	0.3462*	**0.3488**$^{*+}$
		Pr@20	0.3796	0.4010	**0.4122**	**0.4082**	**0.4082**
		RI	n/a	0.3061	**0.4286** (+40.0%)	0.3061	**0.4694** (+53.3%)
		APloss	n/a	0.5736	**0.3662** (−36.2%)	0.5184	**0.4261** (−17.8%)
	Terabyte	MAP	0.3012	0.3187	**0.3287***	0.3188	**0.3311**$^{*+}$
		Pr@20	0.4878	0.4939	**0.5010**	0.4980	**0.5173**
		RI	n/a	0.2245	**0.3469** (+54.5%)	0.1837	**0.3469** (+88.8%)
		APloss	n/a	0.6847	**0.2797** (−59.1%)	0.6608	**0.2614** (−60.4%)

Table 2: Comparison of BoostMM and BoostRM with SMM and RM3 respectively. Note that for APloss, lower is better (negative change is good), while for RI, higher is better. '*' and '+' mean the MAP improvement is statistically significant over NoFB and the corresponding baseline feedback method respectively. The improvement of APloss and RI is shown for BoostMM/BoostRM relative to SMM/RM3.

attempts to learn a ranking function through directly optimizing retrieval measures. One variation of AdaRank used in our comparison is **AdaRank.MAP**, which tries to optimize MAP. Since it is non-trivial to directly optimize the proposed fbloss using AdaRank, we further extend AdaRank to optimize a loss function that is similar to fbloss: a novel robustness-related measure E' that measures the performance degradation in feedback: $E'(q_i) = E\left(\sum_{k=1}^{t} \alpha_k \cdot \phi_k(q_i)\right) - E(q_i)$. According to the Formula 6 in [30], AdaRank turns out to minimize an exponential loss function $\sum_{i=1}^{|Q|} \exp\{-E'(q_i)\}$, leading to a new run **AdaRank.FB**. Note that AdaRank.FB is no longer the traditional AdaRank algorithm because the loss function is novel, thus it is a very strong baseline.

We use the Dirichlet smoothing method [32] for all document language models, where we set the $\mu = 1,000$. Besides, as suggested in our previous study [18], we set the mixture noise parameter λ to 0.9 for SMM and all the mixture-model style basis feedback methods. We also set the number of expansion terms to 40. These parameter settings work well and are used in our experiments unless otherwise stated.

It does not make sense to talk about fbloss alone, since we can always get 0 loss for any feedback method by setting the feedback coefficient α to 0. In the traditional evaluation strategy [31, 18], people often tune α to optimize one retrieval precision measure. We thus follow such a strategy to first optimize α in terms of MAP, on the basis of which we then try to reduce fbloss. Specifically, we give a priority to SMM and RM3 to optimize their feedback coefficient α on the training, validate and test sets respectively. However, for all the mixture-model style basis feedback methods, we simply set the feedback coefficients to those optimized for SMM, while for all the relevance-model style basis feedback methods, we use RM3's setting directly.

We are interested in both effectiveness and robustness of pseudo feedback methods, so besides MAP (on top-ranked 1,000 documents) and Pr@20, we also compare all runs in

Metric	fbDocCount = 20			fbDocCount = 50		
	AdaRank.MAP	AdaRank.FB	FeedbackBoost	AdaRank.MAP	AdaRank.FB	FeedbackBoost
MAP	0.3451	0.3535	**0.3537**	0.3418	0.3485	**0.3488**
RI	0.3061	0.3878	**0.4286**	0.3061	0.3469	**0.4694**
APloss	0.9776	0.4333	**0.3662**	1.0033	0.6329	**0.4261**

Table 3: Comparison of FeedbackBoost, AdaRank.MAP, and AdaRank.FB on the test set of Robust04. Note that AdaRank.FB is different from the traditional AdaRank algorithm, since it is armed with a novel robustness-related loss function.

Collection		NoFB-1	REXP-FB	NoFB-2	BoostRM	BoostMM
Robust04	MAP	0.2152	0.2451	0.2502	0.2617^+	0.2752^+
	RI	n/a	0.3773	n/a	0.5100	**0.5221**
Terabyte	MAP	0.2736	0.3004	0.2901	0.3086^+	0.3230^+
	RI	n/a	0.2624	n/a	0.5135	**0.5270**

Table 4: Comparison of FeedbackBoost and REXP-FB on the same collections, where cross-validation is used for training. '+' indicates that the improvement of MAP over NoFB-2 is statistically significant.

terms of two robustness measures, the robustness index (RI) and the accumulative loss of retrieval performance (APloss). $\mathrm{RI} = 1 - 2 \cdot \mathrm{fbloss}$, is essentially a transformation of the fbloss proposed in our paper; we show RI instead of fbloss mainly because RI was often used in previous studies, e.g., [7]. APloss is to measure the feedback loss in a finer degree, which is the accumulative AP degradation in failure cases (since AP is used as E in our paper), defined as: $\mathrm{APloss} = \sum_{i=1}^{|Q|} [E(q_i) - E(H(q_i))] \cdot I\{E(H(q_i)) < E(q_i)\}$

7. EXPERIMENT RESULTS

7.1 Performance of FeedbackBoost

Table 2 compares MAP, Pr@20, RI, and APloss for NoFB, SMM, RMM, BoostMM, and BoostRM on the validate and test sets. Note that each time all runs use the same set of feedback documents to make the comparison fair; that is, we fix the base ranking for all runs. Besides, the iteration number T in FeedbackBoost is chosen to minimize the corresponding fbloss on the validate sets. We also vary the number of feedback documents from 20 to 50.

For all cases, we can see that BoostMM and BoostRM significantly improve the robustness over SMM and RM3 respectively. For example, BoostMM *reduces* APloss as compared with SMM by amounts ranging from 50.9% to 77.8% when using 20 feedback documents and from 65.1% to 79.5% when using 50 feedback documents. Moreover, BoostMM and BoostRM also significantly improve MAP over SMM and RM3 in almost all cases. The results demonstrate that the FeedbackBoost algorithm does a good job to improve robustness while still achieving better effectiveness. Moreover, FeedbackBoost works consistently well when we use different number of feedback documents. The performance of FeedbackBoost shows that it is effective to combine multiple feedback methods using our FeedbackBoost to improve both robustness and effectiveness of pseudo feedback.

We next compare FeedbackBoost with AdaRank.MAP and AdaRank.FB. These three algorithms are all trained on the same set of relevance-model style basis feedback methods. The iteration number for all the algorithms are chosen to minimize fbloss on the validate set. We present the experiment results in Table 3. One interesting observation is that AdaRank.FB is more effective than AdaRank.MAP, suggesting that the proposed robustness-related measure is better than MAP as an objective function to improve pseudo feedback: one possible explanation is that the average precision does not work well to indicate the room for improvement of a query, so focusing on queries with lower average precision may not be a good strategy to fully exploit the potential of different queries; however, our new measure would be able to capture more precisely the potential room of a query through comparing its performance with the baseline performance. Moreover, we also see that FeedbackBoost is clearly better than AdaRank.FB, though they use similar objective functions, sugggesting that our optimization framework is more effective for pseudo feedback.

We finally compare FeedbackBoost with a state-of-the-art feedback method, REXP-FB [7], which attempted to reduce failures of pseudo feedback but was only able to optimize an indirect objective function. They also used Terabyte and Robust04 as their test collections. So we used a 3-fold (701-750, 751-800, and 801-850) and a 2-fold (301-450 and 601-700) cross validation method to evaluate FeedbackBoost on the whole Terabyte and Robust04 topics respectively in order to compare with their reported numbers. In this comparison, we use the same collection and parameter settings as were used in [7]. The comparison is reported in Table 4.

There are clear performance improvements of our baseline runs (NoFB-2) over theirs (NoFB-1), probably because we use a latest version of Indri search engine (2.10) [1]. We still can see that, in terms of relative improvements, BoostMM performs similarly to REXP-FB, although REXP-FB used a lot of constraints to improve the selection of expansion terms while we only use the default term weighting. However, the most interesting observation is from the comparison of RI, which may be more comparable across systems. We can see that our algorithms achieve a significantly *higher RI* in all cases, even though our baseline run is even harder to beat. This finding confirms the conclusion in [18] that document weighting plays a key role in affecting the robustness of feedback. Furthermore, it suggests that our way of directly optimizing robustness indeed works more effectively.

7.2 Robustness Histograms

To examine in details how badly queries are hurt by a pseudo feedback algorithm, we show in Figure 2 the robustness histograms by combining two test sets (query 651-701 and query 801-850). The x-axis represents the individual APloss in percentage (i.e., $[E(q) - E(\phi(q))]/E(q)$, where ϕ is

[1]http://www.lemurproject.org/

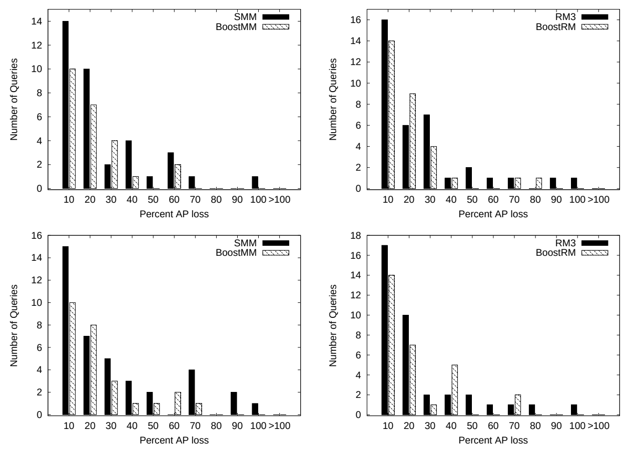

Figure 2: Histograms that compare robustness of SMM vs BoostMM (first and third) and RM3 vs BoostRM (second and fourth) on the combination of two test sets. 20 (50) feedback documents are used in the left (right) two histograms respectively.

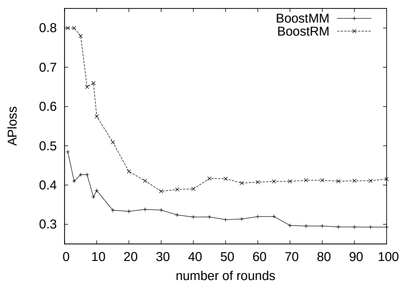

Figure 5: Sensitivity of BoostMM and BoostRM to the iteration number on Robust04 validate set.

the corresponding feedback method) for individual failure queries, and the y-axis stands for the number of queries with the corresponding percentage APloss. We can see that for the worst cases, where a query's AP is decreased by more than 40%, both BoostMM and BoostRM perform much better than SMM and RM3 respectively. It suggests that the proposed FeedbackBoost algorithm indeed concentrates more on difficult queries that are hurt seriously by baseline feedback methods, i.e., SMM and RM3, and the significant reduction of APloss and improvement of RI shown in Table 2 could be mainly due to the elimination of those worst cases.

7.3 Parameter Sensitivity

Usually there is a tradeoff between robustness and effectiveness: if we use a smaller feedback coefficient α, there would be fewer feedback failures, but we may not fully improve the effectiveness; if we increase this α to some appropriate value, the overall retrieval precision could be optimized, which, however, may lead to more feedback failures. Although we have shown that FeedbackBoost improves both robustness and effectiveness at the same time, we are still interested in how the performance of FeedbackBoost interacts with α (where α in FeedbackBoost is used to control the feedback interpolation of each basis feedback method involved.) We thus draw the sensitivity curves for FeedbackBoost in terms of the percentage MAP degradation and the overall MAP improvement in Figure 3 and 4 respectively. The percentage MAP degradation is essentially the relative APloss, i.e., $\text{APloss}/[\sum_i^{|Q|} E(q_i)]$; the overall MAP improvement of ϕ is defined as $[\sum_i^{|Q|} E(\phi(q_i))]/[\sum_i^{|Q|} E(q_i)] - 1.0$. We use 50 feedback documents in all curves. The curves clearly show that FeedbackBoost consistently improves the robustness and effectiveness over two baseline algorithms. Additionally, our algorithm is also less sensitive to α in both curves, and setting α around 0.8 often leads to a large improvement in precision with only a small amount of failures.

Finally, we show in Figure 5 the curves of performance changes as we increase the iteration number T. We see that the APloss decreases steadily and quickly as the training goes on, until it reaches its plateau. In our experiments, we can usually find the best parameter T within 100 rounds.

8. CONCLUSIONS

In this paper, we propose a novel learning algorithm, FeedbackBoost, based on the boosting framework to improve pseudo feedback. A major contribution of our work is to optimize pseudo feedback based on a novel loss function that *directly measures both robustness and effectiveness*, which has not been achieved in any previous work.

The experiment results show that the proposed FeedbackBoost algorithm can improve average precision effectively and meanwhile reduce the number and magnitude of feedback failures dramatically as compared to two representative pseudo feedback methods based on language models, the mixture model and the relevance model. We also compare our algorithm with a recently proposed robust feedback method, and the results show that our method is more robust. In addition, we compare FeedbackBoost with a well-performing learning to rank approach applied for pseudo feedback and observe that FeedbackBoost works clearly better. These results show that the proposed FeedbackBoost is more effective and robust than any of the existing method for pseudo feedback, including both traditional pseudo feedback methods and new learning-based approaches.

Our work can be extended in several ways. First, in our current work, we only use basis feedback methods with different document weighting strategies, so a straightforward future work is to also introduce term weighting methods to construct a larger set of basis feedback methods. Second, we are also interested in combining even more recently proposed pseudo feedback algorithms, e.g., [2, 28, 7, 20], and other families of feedback methods, e.g., [24, 22], to diversify our basis feedback methods. Third, personalized search is another scenario which shares similar effectiveness-robustness tradeoff issues; thus it is also interesting to improve personalized search by exploring the FeedbackBoost framework.

9. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their useful comments. This material is based upon work supported by the National Science Foundation under Grant Numbers IIS-0713581 and CNS-1028381. The first author is also supported by a Yahoo! Key Scientific Challenge Award.

10. REFERENCES

[1] N. Abdul-Jaleel, J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, D. Metzler, M. D. Smucker, T. Strohman, H. Turtle, and C. Wade. Umass at trec 2004: Novelty and hard. In *TREC '04*, 2004.

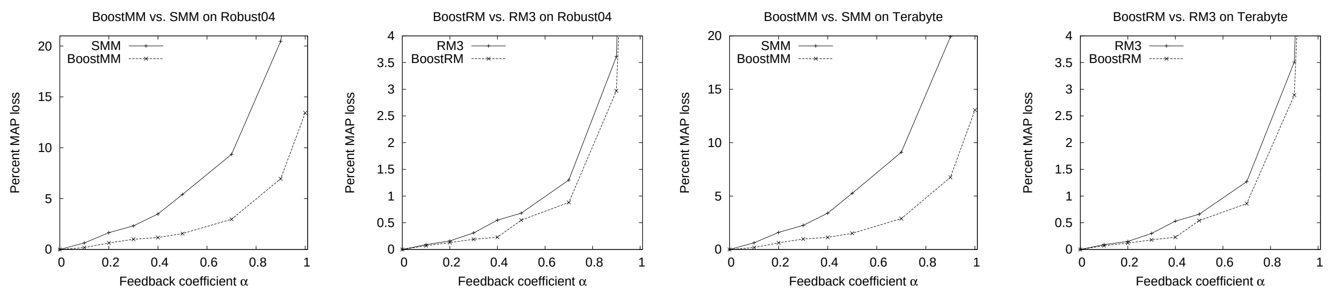

Figure 3: Sensitivity of the percentage degradation of MAP in failure queries to the feedback coefficient α. SMM versus BoostMM and RM3 versus BoostRM are shown in the odd and even-number figures respectively.

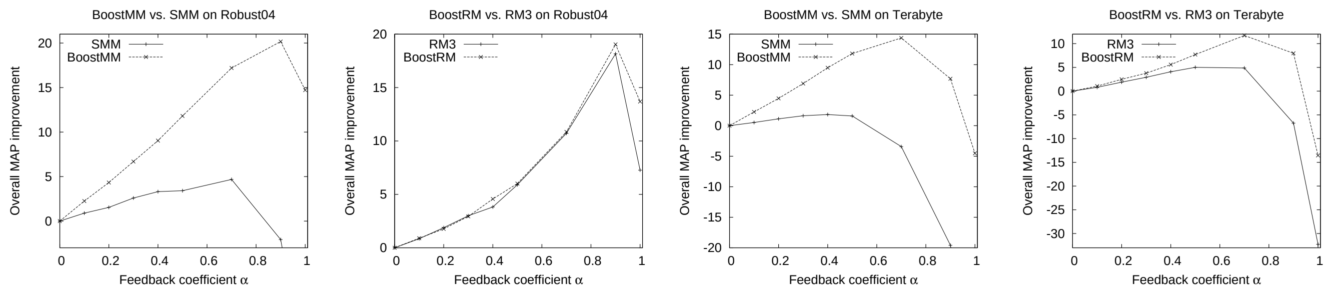

Figure 4: Sensitivity of the average improvement of MAP in all queries to the feedback coefficient α. SMM versus BoostMM and RM3 versus BoostRM are shown in the odd and even-number figures respectively.

[2] G. Amati, C. Carpineto, and G. Romano. Query difficulty, robustness, and selective application of query expansion. In *ECIR '04*, pages 127–137, 2004.

[3] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec 3. In *TREC '94*, pages 69–80, 1994.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05*, pages 89–96, 2005.

[5] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *SIGIR*, pages 243–250, 2008.

[6] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of ICML*, pages 129–136, 2007.

[7] K. Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *CIKM '09*, pages 837–846, 2009.

[8] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT '95*, pages 23–37, London, UK, 1995. Springer-Verlag.

[10] D. Harman and C. Buckley. The nrrc reliable information access (ria) workshop. In *SIGIR '04*, pages 528–529, 2004.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[12] B. He and I. Ounis. Finding good feedback documents. In *CIKM '09*, pages 2011–2014, 2009.

[13] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM KDD 2002*, pages 133–142, 2002.

[14] J. D. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01*, pages 111–119, 2001.

[15] V. Lavrenko and W. B. Croft. Relevance-based language models. In *SIGIR '01*, pages 120–127, 2001.

[16] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[17] Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. In *CIKM '09*, 2009.

[18] Y. Lv and C. Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of CIKM '09*, 2009.

[19] Y. Lv and C. Zhai. Positional language models for information retrieval. In *SIGIR '09*, pages 299–306, 2009.

[20] Y. Lv and C. Zhai. Positional relevance model for pseudo-relevance feedback. In *SIGIR '10*, pages 579–586, 2010.

[21] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98*, pages 275–281, 1998.

[22] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *JASIS*, 27(3):129–146, 1976.

[23] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC '94*, pages 109–126, 1994.

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

[25] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *JASIS*, 41(4):288–297, 1990.

[26] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *COLT' 98*, pages 80–91, 1998.

[27] N. Soskin, O. Kurland, and C. Domshlak. Navigating in the dark: Modeling uncertainty in ad hoc retrieval using multiple relevance models. In *ICTIR '09*, pages 79–91, 2009.

[28] T. Tao and C. Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *SIGIR '06*, pages 162–169, 2006.

[29] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *SIGIR '07*, pages 295–302, 2007.

[30] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR '07*, pages 391–398, 2007.

[31] C. Zhai and J. D. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM '01*, pages 403–410, 2001.

[32] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01*, pages 334–342, 2001.

[33] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR '07*, pages 287–294, 2007.