Ranking Refinement and Its Application to Information Retrieval

Rong Jin
Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
rongjin@cse.msu.edu

Hamed Valizadegan
Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
valizade@cse.msu.edu

Hang Li
Microsoft Research Asia
4F, Sigma Center
No.49 Zhichun Road, Haidian
Beijing, 100080, China
hangli@microsoft.com

ABSTRACT

We consider the problem of **ranking refinement**, i.e., to improve the accuracy of an existing ranking function with a small set of labeled instances. We are, particularly, interested in learning a better ranking function using two complementary sources of information, ranking information given by the existing ranking function (i.e., the base ranker) and that obtained from users' feedbacks. This problem is very important in information retrieval where feedbacks are gradually collected. The key challenge in combining the two sources of information arises from the fact that the ranking information presented by the base ranker tends to be imperfect and the ranking information obtained from users' feedbacks tends to be noisy. We present a novel boosting algorithm for ranking refinement that can effectively leverage the uses of the two sources of information. Our empirical study shows that the proposed algorithm is effective for ranking refinement, and furthermore it significantly outperforms the baseline algorithms that incorporate the outputs from the base ranker as an additional feature.

Categories and Subject Descriptors: H.3.3 [Information Systems]: Information Search and Retrieval; I.2.6 [Artificial Intelligence]: Learning

General Terms: Design, Experimentation, Theory.

Keywords: Learning to Rank, Background Information, Boosting, Incremental Learning.

1. INTRODUCTION

Learning to rank is a relatively new area of study in machine learning. It aims to learn an assignment of scores to objects and rank the objects on the basis of the scores. It has received much attention in recent years because of its important role in information retrieval. Most research in learning to rank is conducted in the supervised fashion, in which a ranking function is learned from a given set of training instances. The drawback with the supervised approach is that they tend to fail when the number of training instances is small.

In several real-world applications, in addition to the labeled training instances, *a base ranker* is available that can be used to rank the objects. Then, the research question is how to exploit the outputs from the base ranker when learning a ranking function from a small number of labeled instances. We refer to this problem as **Ranking Refinement** to distinguish it from supervised learning for ranking. Below we show two examples for the application of ranking refinement:

Relevance feedback In information retrieval, documents are often ordered by a predefined relevance ranking function, such as BM25 [1] and Language Model for IR [2], that assesses the relevancy of documents to a given query. Relevance feedback techniques are proposed to improve the retrieval accuracy by allowing users to provide relevance judgments for the first a few retrieved documents. The research question here is how to enhance the accuracy of relevance feedback by combining the uses of the two types of information. In this case, the base ranker is the relevance ranking function, and the training instances are the documents that are judged by the users.

Recommender system The goal of a recommender system is to rank the items according to the interest of an active user (i.e., the test user). Usually, a few rated items are provided to indicate the preference of the active user. Using the collaborative filtering techniques [3], we can come up with a preliminary list of items ranked by using the preference information of other users. The research question here is how to enhance the final ranking accuracy by leveraging the two types of information. In this case, the base ranker is the collaborative filtering algorithm, and the labeled instances are the items labeled by the active user.

Furthermore, any online learning of ranking functions can be viewed as a ranking refinement problem in that the ranking function is updated iteratively with new training instances collected on the fly.

A straightforward approach toward ranking refinement is to view the scores of the base ranker as an additional feature, and learn a ranking function over the augmented features. As will be shown in the experiments, this is not the best approach for exploiting the information hidden in the base ranking function. We believe that the most valuable information behind the base ranker is not its scores but the ranked list of objects it produces. We therefore view the base ranker and the labeled instances as two complementary sources of information. The key challenge in combining these two sources of information is that the ranked list

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

generated by the base ranker tends to be imperfect while the labeled instances tend to be noisy. In this paper, we present a boosting algorithm for ranking refinement that can effectively utilize the two types of information. Our empirical study with relevance feedback and recommender system show that the proposed algorithm is effective for ranking refinement, and it significantly outperforms the baseline algorithms that incorporate the outputs from the base ranker as an additional feature for the objects.

2. RELATED WORK

Most learning to rank algorithms are designed for the setting of supervised learning, in which a ranking function is learned from labeled instances. The problem of learning to rank is often cast as a classification problem where the goal is to correctly classify the ordering relationship between any two instances. Three well-known approaches in this category are Ranking-SVM [4, 5], RankBoost [6], and RankNet [7]. Ranking-SVM minimizes the number of incorrectly ordered pairs within the maximum margin framework. Several variants [8, 9] are developed to further enhance the performance of Ranking-SVM. RankBoost learns a ranking model based on the same consideration, but by means of Boosting. RankNet [7] is a neural network based approach that uses cross entropy as its loss function. Recently, Xu et al. [10] proposed another approach that is aimed at directly optimizing the performance measures in information retrieval, such as Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG). A special group of ranking problem is called ordinal regression [4], in which the output of the ranking function is restricted to a few ordinal categories. Example algorithms for ordinal regression include the maximum margin based approach [11] and the Gaussian process based approach [12]. The ranking refinement problem differs from the supervised ranking problem in that an imperfect base ranker is provided in addition to the labeled training instances.

The ranking problem is essential to information retrieval, whose goal is to rank a collection of documents by their relevance to a given query. In particular, relevance feedback techniques [13] are developed to improve the accuracy of the existing retrieval algorithms. There are two types of relevance feedback. The first type, termed user relevance feedback, enhances the retrieval accuracy by collecting the user relevance judgments for the documents that are ranked on the top of the list. As pointed out in the introduction section, the user relevance feedback problem can be treated as a problem of ranking refinement. In the empirical study, we will show that the proposed algorithm for ranking refinement significantly outperforms the standard relevance feedback algorithm (i.e., the Rocchio algorithm) over several datasets. The second type of relevance feedback, often termed pseudo relevance feedback, does not explicitly collect the user relevance judgments. Instead, it treats the top ranked documents as relevant to the given query, and the documents ranked at the bottom as irrelevant. These pseudo relevance judgments are used to improve the existing ranking function. It is well known in information retrieval that pseudo relevance feedback may result in degradation of retrieval performance given the high probability of errors in pseudo relevance judgments [13]. This is similar to the noise of training instances in ranking refinement.

3. RANKING REFINEMENT

3.1 Problem Definition

Let $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ denote the set of instances to be ordered, where each instance $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of d dimensions. Let $G : \mathbb{R}^d \to \mathbb{R}$ denote the base ranking function (base ranker), and $g_i = G(\mathbf{x}_i)$ denote the ranking score assigned to $\mathbf{x}_i$ by the base ranking function G. Instance $\mathbf{x}_i$ is ranked before $\mathbf{x}_j$ if $g_i > g_j$. To make our problem general, we assume the label information collected from user feedback is presented as a set of ordered pairs, denoted by $\mathcal{O} = \{(\mathbf{x}_{i_k} \succ \mathbf{x}_{j_k}) | k = 1, \ldots, m\}$ where each pair $\mathbf{x}_i \succ \mathbf{x}_j$ indicates that instance $\mathbf{x}_i$ is ranked before $\mathbf{x}_j$[1]. The goal of ranking refinement is to learn a ranking function $F : \mathbb{R}^d \to \mathbb{R}$ by exploiting both the labeled pairs in $\mathcal{O}$ and the ranking information given by G.

3.2 Encoding Ranking Information

The first important question for ranking refinement is how to encode the ranking information provided by the base ranking function G. A straightforward approach is to use the ranking scores computed by G as an additional feature, and apply the existing algorithms, such as RankBoost and Ranking-SVM, to learn a ranking function from the labeled instances. The drawback of this approach is twofold:

- First, this approach only utilizes the ranking scores of the labeled instances. The ranking information generated by the base ranking function for the unlabeled instances is completely ignored by this approach. Since the number of labeled instances collected from users' feedbacks is considerably smaller than the number of unlabeled instances, this approach is not optimal in exploiting the information provided by the base ranking function.
- Second, we believe that the ranking orders generated by the base ranking function is substantially more reliable than the numerical values of the ranking scores. Similar observation is found in the study of meta search whose goal is to combine the retrieval results of multiple search engines to create a better ranking list [14]. Empirical studies [14] showed that the meta search algorithms based on the document ranks often outperform the algorithms that directly use the relevance scores.

To address the above problems, we encode the order information generated by the base ranking function G with matrix $W \in [0, 1]^{n \times n}$. Each $W_{i,j}$ in the matrix represents the probability of ranking $\mathbf{x}_i$ before $\mathbf{x}_j$ and is defined as follows

$$W_{i,j} = \frac{\exp(\lambda g_i)}{\exp(\lambda g_i) + \exp(\lambda g_j)} \quad (1)$$

In the above, $W_{i,j}$ is defined by a softmax function and the parameter $\lambda \geq 0$ represents the confidence of the base ranking function. To see the effect of λ, we consider two extreme cases:

- $\lambda = 0$. In this case, we have $W_{i,j} = 0.5$, which indicates that the ordering information generated by the base ranker is completely ignored.

[1] This is because any labeled instances can be converted into ordered pairs while the converse is not true.

- $\lambda = \infty$. In this case, we have

$$W_{i,j} = \begin{cases} 1 & g_i > g_j \\ 0.5 & g_i = g_j \\ 0 & g_i < g_j \end{cases} \quad (2)$$

Thus, W is almost a binary matrix, which implies that we completely trust ranked list generated by the base ranking function.

By varying the parameter λ, we are able to alleviate the negative effect from the base ranking function. In our experiment, we set λ to be inverse to the standard deviation of ranking scores of the first 10 retrieved documents.

Similarly, we encode the ordering information inside the set $\mathcal{O}$ with matrix T as follows:

$$T_{i,j} = \begin{cases} 1 - \eta/2 & (\mathbf{x}_i \succ \mathbf{x}_j) \in \mathcal{O} \\ \eta/2 & \text{otherwise} \end{cases} \quad (3)$$

where parameter $\eta \in [0, 1]$. $T_{i,j}$ represents the probability of ranking ranking $\mathbf{x}_i$ before $\mathbf{x}_j$ in the training data. The parameter η reflects the error rate of training data, and is particularly useful when the labeled instances are derived from *implicit* user feedback that is usually noisier. In our experiment, we set $\eta = 1/2$.

3.3 Objective Function

The goal of ranking refinement is to learn a ranking function $F : \mathbb{R}^d \rightarrow \mathbb{R}$ from matrix W and T that produces a more accurate ranked list than the base ranking function G. In particular, the optimal ranking function F should be consistent with the ranking information in W and T. To this end, we measure the ranking errors of F with respect to both W and F, i.e.,

$$err_w = \sum_{i,j=1}^{n} W_{i,j} I(F_j \geq F_i) \quad (4)$$

$$err_t = \sum_{i,j=1}^{n} T_{i,j} I(F_j \geq F_i) \quad (5)$$

In the above, we introduce $F_i = F(\mathbf{x}_i)$ and the indicator function $I(x)$ that outputs 1 when the input boolean variable x is true and zero otherwise. There are two problems with directly using the ranking errors err_w and err_t as the objective function:

- First, both error functions are non-smooth functions since the indicator function $I(x)$ is non-smooth. It is well know that optimizing a non-smooth function is computationally more challenging than optimizing a smooth function because the derivative of a non-smooth function is not well defined [15].

- Second, with two objectives at hand, the problem is essentially a multi-objective optimization problem [16]. Thus, another important question is how to combine multiple objectives into one single objective.

In the following subsections, we will address these two questions separately.

3.3.1 Relaxation with Exponential Functions

To address the problem with non-smooth objective functions, we follow the idea of boosting by replacing the indicator function $I(x \geq y)$ with an exponential function $\exp(x - y)$. The resulting new objective functions are:

$$\widehat{err}_w = \sum_{i,j=1}^{n} W_{i,j} \exp(F_j - F_i) \quad (6)$$

$$\widehat{err}_t = \sum_{i,j=1}^{n} T_{i,j} \exp(F_j - F_i) \quad (7)$$

Note that since $\exp(x - y) \geq I(x \geq y)$, by minimizing the errors $\widehat{err}_w$ and $\widehat{err}_t$, we are effective in reducing the original ranking errors err_w and err_t. Another advantage of using $\widehat{err}_w$ and $\widehat{err}_t$ comes from the theoretic result of AdaBoost [17], i.e., by minimizing the exponential loss function, the resulting classifier will not only reduce the training errors but also maximize the classification margin. The enlarged classification margin is the key to guarantee a low generalization error for testing instances [17].

Remark: It is interesting to examine the effect of the smoothing parameter η on the ranking error $\widehat{err}_t$. By substituting the expression (3) for $T_{i,j}$ in (7), we have $\widehat{err}_t$ expressed as follows:

$$\begin{aligned} \widehat{err}_t &= (1-\eta) \sum_{(\mathbf{x}_i \succ \mathbf{x}_j) \in \mathcal{O}} \exp(F_j - F_i) \\ &\quad + \frac{\eta}{2} \sum_{i,j=1}^{n} [\exp(F_i - F_j) + \exp(F_j - F_i)] \\ &\approx (1-\eta) \sum_{(\mathbf{x}_i \succ \mathbf{x}_j) \in \mathcal{O}} \exp(F_j - F_i) + \frac{\eta}{2} \sum_{i,j=1}^{n} (F_i - F_j)^2 \\ &= (1-\eta) \left(\sum_{(\mathbf{x}_i \succ \mathbf{x}_j) \in \mathcal{O}} \exp(F_j - F_i) + \frac{\eta}{2(1-\eta)} \|\mathbf{F}\|_S^2 \right) \quad (8) \end{aligned}$$

where $\|\mathbf{F}\|_S^2$ is a norm of vector $\mathbf{F} = (F_1, \ldots, F_n)$ defined as follows:

$$\|\mathbf{F}\|_S^2 = \mathbf{F}^\top (nI - \mathbf{ee})\mathbf{F}$$

where I is the identity matrix and $\mathbf{e}$ is a vector of all ones. The approximation of the second step in the above derivation follows the Taylor expansion of the exponential function. Clearly, the second term in (8), i.e., $\eta\|\mathbf{F}\|_S^2/2(1-\eta)$, is similar to the regularization term used by Support Vector Machines (SVM) [18]. Thus, the parameter η plays the role of coefficient in the regularized ranking error $\widehat{err}_t$.

3.3.2 Combination of Two Objectives

The problem of optimizing multiple objectives is usually called multi-objective optimization problem [16]. The most common approach is to linearly combine objectives, which in our case is to linearly combine the two error functions, i.e.,

$$L_a = \gamma \widehat{err}_w + \widehat{err}_t = \sum_{i,j=1}^{n} (\gamma W_{i,j} + T_{i,j}) \exp(F_j - F_i) \quad (9)$$

where parameter γ is used to combine two classification errors. We refer to the approach based on the above objective function as "**Linear Ranking Refinement**", or **LRR**, for short. The main problem with using the linearly combined objectives is how to decide an appropriate value for γ. In our experiments, we will show that different γ could result in very different performance in information retrieval.

To resolve the difficulty, we consider another approach which makes combination of the two errors by their products, i.e.,

$$\begin{aligned} L_p &= \widehat{err}_w \times \widehat{err}_t \\ &= \left(\sum_{i,j=1}^{n} T_{i,j}\exp(F_j - F_i)\right)\left(\sum_{i,j=1}^{n} W_{i,j}\exp(F_j - F_i)\right) \end{aligned} \quad (10)$$

We refer to the approach as "**Multiplicative Ranking Refinement**", or **MRR** for short.

The first concern on using the product is whether the resulting solution is Pareto efficient [16]. A solution $\mathbf{F} = (F_1, \ldots F_n)$ is Pareto efficient for the objectives $\widehat{err}_w$ and $\widehat{err}_t$ if there does not exist any other solution $\mathbf{F}' = (F'_1, \ldots, F'_n)$ that is either

1. $\widehat{err}_w(F') < \widehat{err}_w(F)$ and $\widehat{err}_t(F') \leq \widehat{err}_t(F)$, or

2. $\widehat{err}_w(F') \leq \widehat{err}_w(F)$ and $\widehat{err}_t(F') < \widehat{err}_t(F)$.

In other words, if $\mathbf{F}$ is Pareto efficient, it guarantees that no solution is able to further reduce the two objectives simultaneously than $\mathbf{F}$. It is well known that, according to multi-objective optimization theory [16], the solution found by minimizing L_a is guaranteed to be Pareto efficient. Regarding the Pareto efficiency when minimizing L_p in (10), we have the following theorem:

THEOREM 1. *The optimal solution $\mathbf{F} = (F_1, \ldots, F_n)$ found by minimizing the objective function L_p is Pareto efficient.*

The proof of this theorem can be found in Appendix A. The main advantage of using L_p rather than L_a is that it does not need a weight parameter. This will be revealed in our empirical studies in that minimization of L_p usually significantly outperforms minimization of L_a even when the optimal combination weight γ is used for L_a.

In order to compare the properties of the two different approaches for combination, we examine their first order derivatives. Let ξ denote the parameters used by the ranking function $F(\mathbf{x})$. Then, the first order derivatives of L_a and L_p with respect to ξ are given as follows:

$$\nabla_\xi L_a = \sum_{i,j=1}^{n} (T_{i,j} + \gamma W_{i,j})\exp(F_j - F_i)(\nabla_\xi F(\mathbf{x}_j) - \nabla_\xi F(\mathbf{x}_i))$$

$$\nabla_\xi L_p = L_p\left(\sum_{i,j=1} (a_{i,j} + b_{i,j})\exp(F_j - F_i)(\nabla_\xi F(\mathbf{x}_j) - \nabla_\xi F(\mathbf{x}_i))\right)$$

where

$$a_{i,j} = \frac{W_{i,j}\exp(F_j - F_i)}{\sum_{i,j=1}^{n} W_{i,j}\exp(F_j - F_i)} \quad (11)$$

$$b_{i,j} = \frac{T_{i,j}\exp(F_j - F_i)}{\sum_{i,j=1}^{n} T_{i,j}\exp(F_j - F_i)} \quad (12)$$

Note that both derivative shares similar structures. The key difference between $\nabla_\xi L_a$ and $\nabla_\xi L_p$ is that in $\nabla_\xi L_p$, $a_{i,j}$ and $b_{i,j}$ are used to weight the contribution from W and T for instance pair $(\mathbf{x}_i, \mathbf{x}_j)$ when computing the derivative. This is in contrast to $\nabla_\xi L_a$ where the weights for instance pair $(\mathbf{x}_i, \mathbf{x}_j)$ are $\gamma W_{i,j}\exp(F_j - F_i)$ and $T_{i,j}\exp(F_j - F_i)$. The main advantage of using $a_{i,j}$ and $b_{i,j}$ comes from the fact that they are normalized, i.e., $\sum_{i,j=1}^{n} a_{i,j} = \sum_{i,j=1}^{n} b_{i,j} = 1$, and therefore the contributions from W and T are naturally balanced when calculating the derivative.

Algorithm 1 Boosting algorithm for minimizing L_p

```
Compute $W_{i,j}$ and $T_{i,j}$ based on the ranked list and the set of labeled pairs
Initialize $F(\mathbf{x}) = 0$ for all instances
repeat
    Compute $\gamma_{i,j}$ for each instance pair as
        $\gamma_{i,j} = a_{i,j} + b_{i,j}$   (13)
    where $a_{i,j}$ and $b_{i,j}$ are defined in (11) and (12).
    Compute the weight for each instance as
        $w_i = \sum_{j=1}^{n} \gamma_{i,j} - \gamma_{j,i}$   (14)
    Assign each instance the class label $y_i = \text{sign}(w_i)$.
    Train a classifier $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \{0, 1\}$ that maximizes the following quantity
        $\theta = \sum_{i=1}^{n} |w_i| f(\mathbf{x}_i) y_i$   (15)
    Predict $f_i$ for all instances in $\mathcal{D}$
    Compute combination weight $\alpha$ as follows:
        $\alpha = \frac{1}{2}\log\frac{\sum_{i,j=1}^{n} \gamma_{i,j}\delta(f_i, 1)\delta(f_j, 0)}{\sum_{i,j=1}^{n} \gamma_{i,j}\delta(f_j, 1)\delta(f_i, 0)}$   (16)
    where $f_i = f(\mathbf{x}_i)$. $\delta(x, y)$ outputs 1 if $x = y$ and zero otherwise. Break the loop if $\alpha \leq 0$
    Update the ranking function as
        $F(\mathbf{x}) \leftarrow F(\mathbf{x}) + \alpha f(\mathbf{x})$   (17)
until reach the maximum number of iterations
```

3.4 Boosting Algorithm for Ranking Refinement

In this section, we will consider algorithms for learning the ranking function $F(\mathbf{x})$ by respectively minimizing the objective function L_a and L_p. The objective function L_a is similar to the objective function used by Rank-Boost except that a weight $(T_{i,j}+\gamma W_{i,j})$ is used for each instance pair. We thus can simply modify the Rank-Boost algorithm to learn the optimal ranking function $F(\mathbf{x})$. Hence, in the sequel, we will focus on the boosting algorithm for minimizing L_p.

To learn the optimal ranking function $F(\mathbf{x})$, we follow the greedy approach of boosting algorithms. Since the information for training is a set of labeled instance pairs, a straightforward boosting approach is to iteratively update the weights of instance pairs and train a new ranking function for the given weighted pairs. This is the strategy employed in the Rank-Boost algorithm [6]. However, since the number of instance pairs is $\mathcal{O}(n^2)$, this approach could be computationally expensive when the number of instance n is large.

To address the above problem, we present a new boosting algorithm that converts the weights of instance pairs into weights for individual instances. The key idea behind the new boosting algorithm is to derive an upper bound

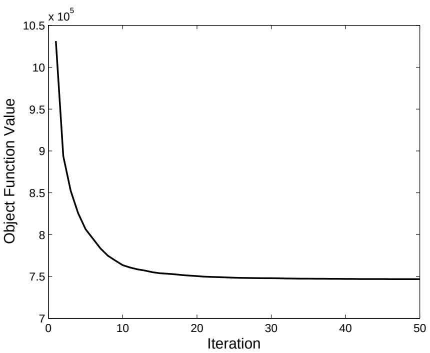


Figure 1: Reduction of the objective function L_p using the OHSUMED Data Set

for the target objective that decouples functions for pairs of instances into functions for individual instances. It is this decoupling that makes it possible to infer weights for individual instances from weights for instance pairs. In addition, the new boosting algorithm is able to derive an appropriate binary class label for each instance using the computed weights. Using both the weights and the class labels of instances, we can train a binary classifier $f : \mathbb{R}^d \to \{0, +1\}$ and update the overall ranking function by $F'(\mathbf{x}) = F(\mathbf{x}) + \alpha f(\mathbf{x})$ where α is the combination weight. Note that by converting a ranking problem into a series of binary classification problems, the new boosting algorithm avoids the high computational cost arising from the large number of instance pairs.

Algorithm 1 summarizes the overall procedures for the proposed boosting algorithm minimizing L_p. As the first step in the iteration, we compute $\gamma_{i,j}$ for every pair of instances that measures the uncertainty of ranking instance $\mathbf{x}_i$ ahead of $\mathbf{x}_j$. Next, we calculate the weight for instance $\mathbf{x}_i$ as $w_i = \sum_{j=1}^n \gamma_{i,j} - \gamma_{j,i}$. It is important to note that w_i can be both positive and negative. In particular, $w_i > 0$ indicates that it is more likely to have $\mathbf{x}_i$ ranked on the top of the ranked list than on the bottom of the list; $w_i \leq 0$ indicates the opposite. Hence, we can derive the class label y_i for $\mathbf{x}_i$ based on the sign of w_i: a positive class for placing instances on the top of the ranked list, and a negative class for placing instances on the bottom of the list. Since $|w_i|$ indicates the overall confidence in deciding the ranking position of $\mathbf{x}_i$ in the list, it is used to weight the importance of individual instances. With this information, we will train a classifier that maximizes θ in (15), which can be interpreted as a sort of classification accuracy. Since most binary classifiers are unable to take weights into consideration, we will divide the training procedure into two steps: in the first step, we sample s instances according to the distribution that is proportional to the weights $|w_i|$; we then train a binary classifier $f : \mathbb{R}^d \to \{0, +1\}$ using the sampled instances. We manually set $s = \max(20, n/5)$ in our empirical study. A similar strategy is employed in the AdaBoost algorithm [6] and its effectiveness has been verified in empirical studies.

In the remaining of this section, we will give justification to the proposed algorithm described in Table 1. The main result is summarized in Theorem 2.

Theorem 2. *Let $f^k(\mathbf{x})$ denote the binary classification function obtained in the kth iteration, and $\gamma_{i,j}^k$ denote $\gamma_{i,j}$ learned in that iteration. The objective function after T iterations, denoted by L_p^T, is bounded as follows:*

$$L_p^T \leq \left(\sum_{i,j=1}^n T_{i,j} \sum_{i,j=1}^n W_{i,j}\right) \exp\left(-\sum_{k=1}^T (\sqrt{\mu_k} - \sqrt{\nu_k})^2\right) \quad (18)$$

where

$$\begin{aligned} \mu_k &= \sum_{i,j=1}^n \gamma_{i,j}^k \delta(f^k(\mathbf{x}_i), 1)\delta(f^k(\mathbf{x}_j), 0) \\ \nu_k &= \sum_{i,j=1}^n \gamma_{i,j}^k \delta(f^k(\mathbf{x}_i), 0)\delta(f^k(\mathbf{x}_j), 1) \end{aligned}$$

The above theorem essentially shows that by using the proposed algorithm, the objective function L_p will be reduced exponentially.

The key to proving Theorem 2 is to establish the relationship between the objective function L_p of two consecutive iterations. This is because by upper bounding the log-ratio between L_p of two consecutive iterations, i.e.,

$$r_t \quad \geq \quad \log L_p^t - \log L_p^{t-1}, \quad (19)$$

we will have

$$L_p^T \quad = \quad L_p^0 \prod_{t=1}^T \frac{L_p^t}{L_p^{t-1}} \leq L_p^0 \exp\left(\sum_{t=1}^T r_t\right) \quad (20)$$

For the convenience of presentation, in the following, we only consider two consecutive iterations without specifying the index of iteration. Instead, we denote the quantities of the current iteration by symbol ˜ to differentiate the quantities of the previous iteration. In order to establish an upper bound for the log ratio, we first introduce the following lemma

Lemma 1. *Assume $\tilde{F}(\mathbf{x}) = F(\mathbf{x}) + \alpha f(\mathbf{x})$ where $\tilde{F}(\mathbf{x})$ and $F(\mathbf{x})$ are the ranking functions of two consecutive iterations, respectively. $f : \mathbb{R}^d \to \{0, 1\}$ is a binary classifier and α is the combination weight. We have the following inequality hold for any F, f, and α:*

$$\log \frac{\tilde{L}_p}{L_p} \quad \leq \quad -2 + \sum_{i,j=1}^n (a_{i,j} + b_{i,j}) \exp(\alpha(f_j - f_i)) \quad (21)$$

where $a_{i,j}$ and $b_{i,j}$ are defined (11) and (12), respectively.

The proof of Lemma 1 can be found in Appendix B. Using the Lemma 1, we present the proof of Theorem 2 in Appendix C.

Finally, we can show the relationship between the objective function L_p and the quantity θ (in (15)) that is used to guide the training of binary classifiers in iterations. This result is summarized in the following theorem:

Theorem 3. *Let θ_k denote the value of the quantity θ (in (15)) that is maximized by the binary classifier $f^k(\mathbf{x})$ learned in the kth iteration. Assume that $\theta_k \geq 0$ for each iteration. Then, the objective function after T iterations, denoted by L_p^T, is bounded as follows:*

$$L_p^T \leq \left(\sum_{i,j=1}^n T_{i,j}\right)\left(\sum_{i,j=1}^n W_{i,j}\right) \exp\left(-\sum_{k=1}^T \theta_k\right) \quad (22)$$

The proof of the above theorem can be found in Appendix D. Theorem 3 provides a theoretical justification for Algorithm 1.

In particular, by maximizing θ, Algorithm 1 effectively reduces the objective function L_p. This is further confirmed by our empirical study. Figure 1 shows an example of reduction in the objective function L_p. We clearly see that the objective function is reduced exponentially and receives the largest reduction during the first few iterations.

4. EXPERIMENTS

In this section, we evaluate the proposed algorithm for ranking refinement by two tasks, i.e., user relevance feedback and recommender system. The objectives of our experiments are: (1) to compare the proposed algorithm for ranking refinement to the existing ranking algorithms, (2) to examine the performance of the proposed algorithm for ranking refinement with different numbers of training instances, (3) to examine the effect of different base rankers on the performance of the proposed algorithm, and (4) to examine the time efficiency of the proposed algorithm for ranking refinement.

4.1 Datasets

For the **Relevance Feedback** experiment, we used the LETOR testbed [20] that includes the OHSUMED dataset and the datasets from TREC 2003 and 2004. The OHSUMED dataset consists of 106 queries. For each query, a number of documents are retrieved and their relevance to the query is given at three levels: definitely (2), possibly (1), or irrelevant (0). There are a total of $16,140$ query-document relevance judgments. For each query-document pair, a total of 25 ranking features are extracted. There are 50 queries in the dataset of TREC 2003, and 75 queries in TREC 2004. For each query, about 1000 documents are retrieved, which amounts to a total of $49,171$ query-document pairs for TREC 2003 and $74,170$ query-document pairs for TREC 2004. A binary relevance judgment is provided for each query-document pair. There are 44 features extracted for each query-document pair. The detailed information about OHSUMED and TREC data sets are available in [20].

For the **Recommender System** experiment, we used the MovieLens dataset, available at [19], contains $100,000$ ratings (from 1 to 5) for 1682 movies given by 943 users. Each movie is represented by 51 binary features: 19 features are derived from the genres of movies and the rest 32 features are derived from the keywords that are used to describe the content of movies[2].

4.2 Experimental Setup

4.2.1 Algorithms

To examine the effectiveness of the proposed algorithm for ranking refinement, we compared the following ranking algorithms:

Base Ranker: It is the base ranker used in the ranking refinement.

Rocchio: This algorithm extends the standard Rocchio algorithm for user relevance feedback and it creates a new query vector by linearly combining the query vector and vectors of feedback documents. Given the initial query Q_0, the relevant documents $(R_1, R_2, ..., R_{n_1})$ and non-relevant documents $(S_1, S_2, ..., S_{n_2})$, the new query according to Rocchio is:

$$Q = Q_0 + \alpha \sum_{i=1}^{n_1} \frac{R_i}{n_1} - \beta \sum_{i=1}^{n_2} \frac{S_i}{n_2} \tag{23}$$

Note, in our case, that each document is not represented by a vector of word frequency, but a vector of features that are computed based on its match to the query. Hence, we don't have Q_0, i.e., the representation vector for query itself. We therefore set Q_0 to be a vector of all zeros. We used the inner product between the new query and documents as the scores to rank the documents. To obtain the best performance, we vary α and β from 1 to 10 and choose the best setting.

SVM: This implements the Ranking-SVM algorithm using the SVM light package. Note that it is commonly believed that Rank-Boost performs equally well as Ranking SVM. The experimental results provided in the LETOR collection also confirm this. Hence, we only compare the proposal algorithm with Ranking-SVM, but not Rank-Boost.

MRR: This is the Multiplicative Ranking Refinement algorithm that minimizes L_p in (10).

LRR: This is the Linear Ranking Refinement algorithm that minimizes L_a in (9). Since the performance of LRR depends on the parameter γ, we run LRR with 100 different values from 0.1 to +10 and choose the best and worst performance. We referred them to as **LRR-Worst** and **LRR-Best**, respectively.

For a fair comparison, the output from the base ranker is used as an extra feature when using SVM (i.e., Ranking-SVM) and Rocchio.

4.2.2 Evaluation Metrics

To evaluate the performance of different algorithms, we used precision and normalized discounted cumulative gain (NDCG) at rank position k. Let $(d_{R_1}, d_{R_2}, ..., d_{R_n})$ denote the top ranked documents according to the ranker R, and $(r_{R_1}, r_{R_2}, .., r_{R_n})$ denote their binary judgments. The precision at rank position k measures the relevancy of the first k documents and is defined as follows

$$P_R@k = \sum_{i=1}^{k} r_{R_i}/k$$

For the OHSUMED dataset, a document is deemed relevant when its score is two. In the case of MovieLense data, a movie is deemed to be interesting to a test user when its rating is no less than 4[3]. Since the first top documents are more important than the other documents, we also employ the NDCG metric [21] that is defined as follows:

$$NDCG_R@k = \frac{DCG_R@k}{DCG_T@k}$$

[2] We downloaded the keywords of each movie from the online movie database IMBD. The 32 most popular keywords used by the 1682 movies were selected.

[3] We have experimented with other thresholds and find similar results in our empirical study. We did not present results for the other thresholds due to the space limitation.

where T stands for the oracle ranker and $DCG_X@k$ is defined as follows [21]:

$$DCG_X@k = \begin{cases} r_{X_1} & if\, k = 1 \\ r_{X_1} + \sum_{i=2}^{k} \frac{r_{X_i}}{\log_2 i} & if\, k > 1 \end{cases}$$

4.2.3 Evaluation Protocol

Unless specified, for all the experiments with relevance feedback, we used the standard BM25 retrieval algorithm (i.e., the 21st feature in OHSUMED data set and 16th in TREC data sets) as the base ranker. We followed the common practice of user relevance feedback by collecting the relevance judgments for the first 10 retrieved documents. These user relevance judgments served as labeled instances in ranking refinement.

For the experiment with recommender system, the base ranker was created by applying a collaborative filtering algorithm, more specifically, the Personality Diagnosis algorithm [3], to the user rating data. In particular, 20 users were randomly selected as the training users, and the remaining 923 users were used for testing. For each test user, 10 rated movies were randomly selected and were used by the collaborative filtering algorithm to identify the 20 training users who share the common interests with the test user. Note that we did not compare the proposed algorithm to other information filtering algorithms because the focus of this study is to examine the effectiveness and the generality of the proposed approach for ranking refinement.

4.3 Results for Relevance Feedback

Figure 2 and 3 show the performance results of different algorithms in terms of precision and NDCG for the first 25 ranked documents. First, by comparing the performance of the two variants of ranking refinement, we observed that the Multiplicative Ranking Refinement (MRR) algorithm is significantly more effective than the Linear Ranking Refinement (LLR) algorithm. Indeed, MRR performs significantly better than the best case of LRR (i.e., LRR-best). The key difference between MRR and LRR is that MRR minimizes the product of the two error functions while LRR minimizes the weighted sum. We believe it is the normalization scheme brought by MRR (see equations in (11) and (12)) that makes it performing better than LRR. Second, comparing to the other three baseline algorithms, i.e., the base ranker, Rocchio, Ranking-SVM, we observed that MRR always significantly outperforms the baseline algorithms in all the cases. More noticeable is the improvement made by the ranking refinement over the first a few ranking positions. We thus conclude that Multiplicative Ranking Refinement is more effective than the baseline algorithms for user relevance feedback in information retrieval.

4.4 Effect of Quality of Base Rankers

To examine the robustness of the proposed algorithm with respect to the imperfectness of the base ranker, we tested the MRR algorithms with three different base rankers. We plotted the results of all different features and selected three features which cover a good range of ranking quality. The chosen base rankers for OHSUMED data set are the features 7, 11, and 21 and those for the TREC data sets are features 16, 21, and 36. Figure 4 shows how the MRR algorithm performs using different base rankers (NDCG shows similar results). The result indicates that the quality of base rankers has a direct impact on the performance of the MRR algorithm. We also observed that the proposed algorithm is able to significantly improve the performance even with a poor base ranker. More impressively, by comparing Figure 4 to Figure 2, we observed that even using the worst base ranker (i.e., feature 7 for OHSUMED, 21 for TREC 2003 ,and 36 for TREC 2004), the retrieval accuracy of MRR is comparable to the other methods using the best base ranker (i.e., the BM25 retrieval algorithm). We thus conclude that the MRR algorithm is resilient to the imperfectness of base rankers.

4.5 Effect of Size of Feedback Data

To investigate the effect of the number of feedback documents on the performance, we ran the MRR algorithm by varying the number of feedback documents from 5 to 20. Figure 5 shows the result using varied number of feedback documents. We clearly observed that the number of feedback documents have a direct effect on the performance of ranking refinement. However, even with a small amount of feedback, MRR is able to improve the retrieval performance considerably, particularly for the accuracy of the first few ranked documents. We thus conclude that the proposed algorithm for ranking refinement is robust to the size of feedback data.

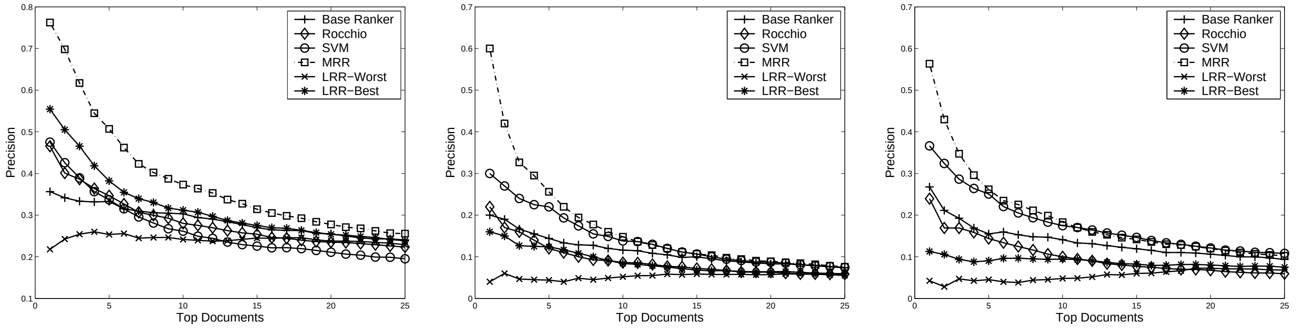
4.6 Results for Recommender System

We evaluated the generality of the proposed algorithm by applying it to recommender system (movie recommendation). Figure 6(a) and Figure 6(b) show the results of different algorithms when applied on the MovieLens dataset. It is surprising to observe that the results of LRR, the linear ranking refinement algorithm, even with the tuned parameter γ, is not even comparable to the the performance of the base ranker. In contrast, the MRR algorithm is able to significantly improve the accuracy of the base ranker and outperform the other baseline algorithms considerably. This result further indicates the importance of appropriately combining the two information sources, i.e., the ranking information behind the base ranker and the feedback information provided by users.

Figure 6(c) shows the sensitivity of MRR to the size of feedback data by varying the number of rated movies by the test user from 5 to 20. Similar to the result for relevance feedback, we observed that the size of feedback data affects the performance of MRR considerably. However, even with 5 rated movies, the MRR algorithm is able to make a noticeable improvement in the prediction accuracy compared to the base ranker. This result further confirms the robustness of the proposed algorithm for ranking refinement with respect to the size of feedback data.

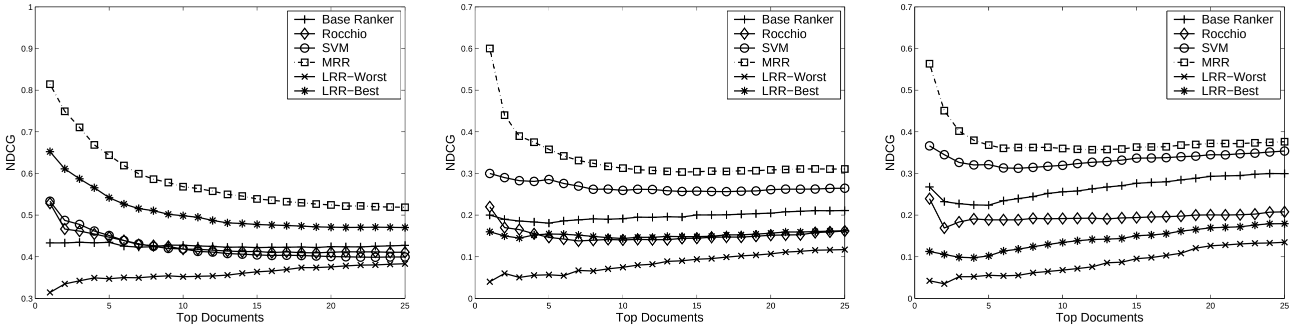
4.7 Time Efficiency of Ranking Refinement

Figure 7 shows the efficiency of the MRR algorithm in terms of the running time for different numbers of rated movies for each test user. We chose movies data set for the experiment because it provides a good range for the number of objects. We partitioned the test users into groups where each group of users has a different number of rated movies. The running time of MRR for each group is calculated by averaging it across all the users in the group. As pointed in Section 3.4 and seen in Figure 7, the running time is linear in the number of instances. Note that the relatively long running time is due to the MATLAB implementation.



(a) OHSUMED (b) TREC 2003 (c) TREC 2004

Figure 2: Precision of relevance feedback for different algorithms



(a) OHSUMED (b) TREC 2003 (c) TREC 2004

Figure 3: NDCG of relevance feedback for different algorithms

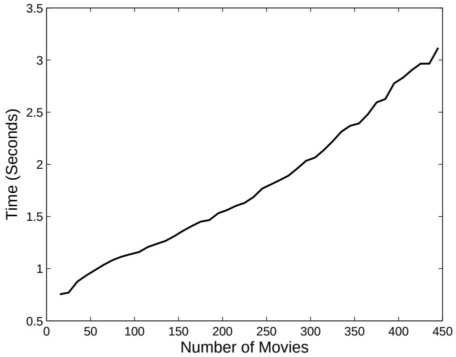


Figure 7: Running time of the MMR algorithm for different numbers of movies rated by test users

5. CONCLUSION

In this paper, we propose the problem of ranking refinement, whose goal is to improve a given ranking function by a small number of labeled instances. The key challenge in combining the ranking information from the base ranker and the labeled instances arises from the fact that the information in the base ranker tends to be inaccurate and the information from the training data tends to be noisy. We present a boosting algorithm for ranking refinement that is resilient to the errors. Empirical studies with relevance feedback and recommender system show promising performance of the proposed algorithm.

6. ACKNOWLEDGEMENTS

The work was supported by the National Science Foundation (IIS-0643494) and National Institute of Health (1R01-GM079688-01). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and NIH.

7. REFERENCES

[1] S. Robertson and D. A. Hull. The trec-9 filtering track final report. In *TREC9*, pages 25–40, 2000.

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

[3] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis. In *UAI*, 2000.

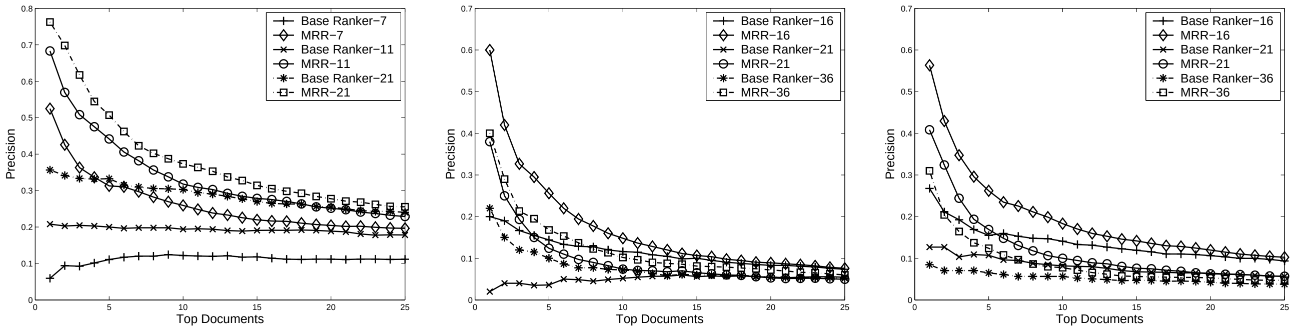
[4] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132, 2000.

[5] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*, pages 133–142, 2002.

[6] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Machine Learning Research*, 4:933–969, 2003.

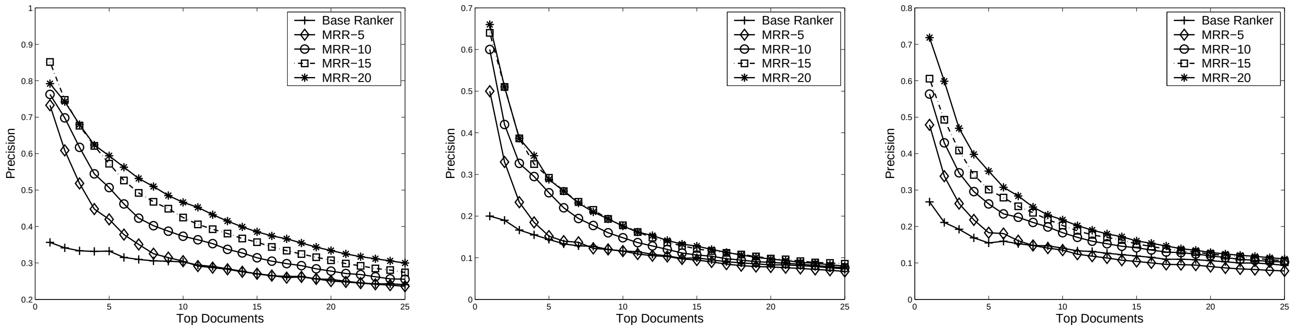
[7] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[8] J. Gao, H. Qi, X. Xia, and J.-Y. Nie. Discriminant model for information retrieval. In *SIGIR*, pages 290–297, 2005.



(a) OHSUMED data set (b) Trec 2003 data set (c) Trec 2004 data set

Figure 4: Precision of MRR with different base rankers for relevance feedback



(a) OHSUMED data set (b) Trec 2003 data set (c) Trec 2004 data set

Figure 5: Precision of MRR with different numbers of feedback documents for relevance feedback

[9] Y. Cao, J. Xu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *SIGIR*, pages 186–193, 2006.

[10] J. Xu and H. Li. A boosting algorithm for information retrieval. In *SIGIR*, pages 473–480, 2007.

[11] A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. In *NIPS*, 2003.

[12] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. Technical report, 2004.

[13] D. Harman. Relevance feedback revisited. In *SIGIR*, 1992.

[14] Mark Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 538–548. ACM, 2002.

[15] R.T. Rockafellar. *Convex analysis*. Princeton University Press, Princeton, N.J., 1970.

[16] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, 546 pp, 1986.

[17] Robert E. Schapire. Theoretical views of boosting and applications. In *Algorithmic Learning Theory, 10th International Conference, ALT '99*, volume 1720, pages 13–25. Springer, 1999.

[18] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[19] GroupLens. *MovieLens Data sets*. http://www.grouplens.org/node/12, 2006.

[20] Microsoft Research Asia. *LETOR: Benchmark Datasets for Learning to Rank*. http://research.microsoft.com/users/tyliu/letor/, 2006.

[21] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

APPENDIX

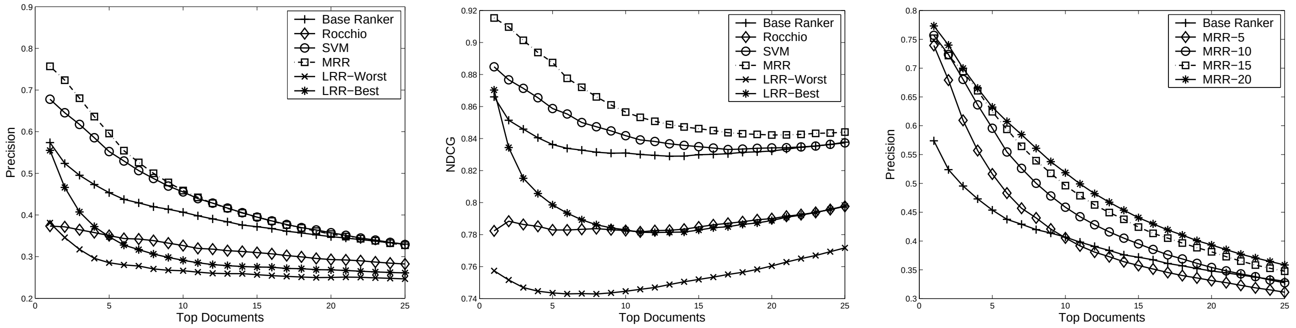
A. PROOF OF THEOREM 1

Proof. First, note that the objective function L_p is convex in terms of $\mathbf{F}$. This is because L_p can be expanded as follows:

$$L_p \quad = \quad \sum_{i,j,k,l=1}^{n} T_{i,j} W_{i,j} \exp(F_j - F_i + F_k - F_l)$$

Since $\exp(F_j - F_i + F_k - F_l)$ is a convex function, L_p is convex. Since L_p is a convex function, the solution found by minimizing L_p will always be global optimal, instead of local optimal.

Second, to show that the optimal solution found by minimizing L_p is Pareto efficient, we prove by contradiction. Let $\mathbf{F}^*$ denote the global minimizer of function L_p. By assuming that Theorem 1 is not correct, there will exist a solution $\mathbf{F} \neq \mathbf{F}^*$ that either (1) $\widehat{err}_w(\mathbf{F}) < \widehat{err}_w(\mathbf{F}^*)$ and $\widehat{err}_t(\mathbf{F}) \leq \widehat{err}_t(\mathbf{F}^*)$, or (2) $\widehat{err}_w(\mathbf{F}) \leq \widehat{err}_w(\mathbf{F}^*)$ and $\widehat{err}_t(\mathbf{F}) < \widehat{err}_t(\mathbf{F}^*)$. We can easily infer $L_p(\mathbf{F}) < L_p(\mathbf{F}^*)$ since (1) both $\widehat{err}_w$ and $\widehat{err}_t$ are non-negative for any solution $\mathbf{F}$, and (2) $L_p = \widehat{err}_w \times \widehat{err}_t$. Clearly, this conclusion contracts the fact that $\mathbf{F}^*$ is a global minimizer of L_p. □



(a) Precision chart (b) NDCG chart (c) Robustness

Figure 6: Precision and NDCG of recommender system for different algorithms

B. PROOF OF LEMMA 1

PROOF. Since $\tilde{F}(\mathbf{x}) = F(\mathbf{x}) + \alpha f(\mathbf{x})$, we have

$$\begin{aligned}
\frac{\tilde{L}_p}{L_p} &= \left(\sum_{i,j=1}^{n} W_{i,j}\exp(F_j - F_i + \alpha(f_j - f_i))\right) \times \\
&\quad \left(\sum_{i,j=1}^{n} T_{i,j}\exp(F_j - F_i + \alpha(f_j - f_i))\right) \\
&= \left(\sum_{i,j=1}^{n} a_{i,j}\exp(\alpha(f_j - f_i)\right)\left(\sum_{i,j=1}^{n} b_{i,j}\exp(\alpha(f_j - f_i)\right)
\end{aligned}$$

where $a_{i,j}$ and $b_{i,j}$ are defined in (11) and (12). Thus, we have an upper bound of the log ratio as follows

$$\begin{aligned}
\log\frac{\tilde{L}_p}{L} &= \log\left(\sum_{i,j=1}^{n} a_{i,j}\exp(\alpha(f_j - f_i))\right) \\
&\quad + \log\left(\sum_{i,j=1}^{n} b_{i,j}\exp(\alpha(f_j - f_i))\right) \\
&\leq -2 + \sum_{i,j=1}^{n}(a_{i,j} + b_{i,j})\exp(\alpha(f_j - f_i))
\end{aligned}$$

The second inequality follows the concaveness of the logarithm function, i.e., $\log x \leq x - 1$ for any $x > 0$. □

C. PROOF OF THEOREM 2

PROOF. Using the upper bound expressed in Lemma 1, we have

$$\begin{aligned}
\log\frac{\tilde{L}_p}{L_p} + 2 &\leq \sum_{i,j=1}^{n}\gamma_{i,j}\exp(\alpha(f_j - f_i)) \\
&= \left(\sum_{i,j=1}^{n}\gamma_{i,j}\delta(f_j,1)\delta(f_i,0)\right)\exp(\alpha) \\
&\quad + \left(\sum_{i,j=1}^{n}\gamma_{i,j}\delta(f_j,0)\delta(f_i,1)\right)\exp(-\alpha)
\end{aligned}$$

Using the definition of α in (16), we have

$$\begin{aligned}
&\log\frac{\tilde{L}_p}{L_p} \leq -2+ \\
&\quad 2\sqrt{\left(\sum_{i,j=1}^{n}\gamma_{i,j}\delta(f_j,1)\delta(f_i,0)\right)\left(\sum_{i,j=1}^{n}\gamma_{i,j}\delta(f_j,0)\delta(f_i,1)\right)} \\
&= -2 + 2\sqrt{\mu\nu}
\end{aligned}$$

In the above, we use the definitions of μ and ν in Theorem 1 to simplify the expression. Since $2 = \sum_{i,j=1}^{n}\gamma_{i,j} \geq \mu + \nu$, we have

$$\begin{aligned}
\log\frac{\tilde{L}_p}{L_p} &\leq -2 + 2\sqrt{\mu\nu} \\
&\leq -\mu - \nu + 2\sqrt{\mu\nu} = -\left(\sqrt{\mu} - \sqrt{\nu}\right)^2
\end{aligned}$$

We thus have

$$\log\frac{L_p^t}{L_p^{t-1}} \leq r_t = -\left(\sqrt{\mu_t} - \sqrt{\nu_t}\right)^2$$

Substituting the above expression for r_t into (20), and further using the fact

$$L_0 = \sum_{i,j=1}^{n} T_{i,j} + W_{i,j},$$

we obtain the result in Theorem 2. □

D. PROOF OF THEOREM 3

PROOF. We rewrite the quantity θ as follows:

$$\begin{aligned}
\theta &= \sum_{i=1}^{n} f_i y_i |w_i| \\
&= \sum_{i=1}^{n} f_i\left(\sum_{j=1}^{n}\gamma_{i,j} - \gamma_{j,i}\right) \\
&= \sum_{i,j=1}^{n}\gamma_{i,j}(f_i - f_j) = \mu - \nu
\end{aligned}$$

Since

$$\begin{aligned}
\mu - \nu &= \left(\sqrt{\mu} - \sqrt{\nu}\right)\left(\sqrt{\mu} + \sqrt{\nu}\right) \\
&\geq \left(\sqrt{\mu} - \sqrt{\nu}\right)^2,
\end{aligned}$$

we have $\theta \geq \left(\sqrt{\mu} - \sqrt{\nu}\right)^2$. Substituting this result into the expression of Theorem 2, we have Theorem 3. □