Reinforcement Learning to Rank with Markov Decision Process

Zeng Wei, Jun Xu*, Yanyan Lan, Jiafeng Guo, Xueqi Cheng
CAS Key Lab of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences
zengwei@software.ict.ac.cn,{junxu,lanyanyan,guojiafeng,cxq}@ict.ac.cn

ABSTRACT

One of the central issues in learning to rank for information retrieval is to develop algorithms that construct ranking models by directly optimizing evaluation measures such as normalized discounted cumulative gain (NDCG). Existing methods usually focus on optimizing a specific evaluation measure calculated at a fixed position, e.g., NDCG calculated at a fixed position K. In information retrieval the evaluation measures, including the widely used NDCG and P@K, are usually designed to evaluate the document ranking at all of the ranking positions, which provide much richer information than only measuring the document ranking at a single position. Thus, it is interesting to ask if we can devise an algorithm that has the ability of leveraging the measures calculated at all of the ranking postilions, for learning a better ranking model. In this paper, we propose a novel learning to rank model on the basis of Markov decision process (MDP), referred to as MDPRank. In the learning phase of MDPRank, the construction of a document ranking is considered as a sequential decision making, each corresponds to an action of selecting a document for the corresponding position. The policy gradient algorithm of REINFORCE is adopted to train the model parameters. The evaluation measures calculated at every ranking positions are utilized as the immediate rewards to the corresponding actions, which guide the learning algorithm to adjust the model parameters so that the measure is optimized. Experimental results on LETOR benchmark datasets showed that MDPRank can outperform the state-of-the-art baselines.

KEYWORDS

learning to rank; Markov decision process

ACM Reference format:
Zeng Wei, Jun Xu*, Yanyan Lan, Jiafeng Guo, Xueqi Cheng. 2017. Reinforcement Learning to Rank with Markov Decision Process. In *Proceedings of SIGIR17, August 7–11, 2017, Shinjuku, Tokyo, Japan,* , 4 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080685

1 INTRODUCTION

Learning to rank has been widely used in information retrieval and recommender systems. Among the learning to rank methods, directly optimizing the ranking evaluation measures is a representative approach and has been proved to be effective. Following the idea of directly optimizing evaluation measures, a number of methods have been proposed, including SVMMAP [17] , Adarank [14] , and PermuRank [15] etc. In training, these methods first construct a loss function on the basis of a predefined ranking evaluation measure. Then, approximations of the loss function or convex upper bounds upon the loss function are constructed and then optimized, leading to different directly optimizing learning to rank algorithms. In ranking, each document is assigned a relevance score based on the learned ranking model.

Several learning to rank models that directly optimize evaluation measure have been proposed and applied to variant ranking tasks. Different loss functions and optimization techniques are adopted in these methods. For example, the loss function of SVMMAP is constructed on the basis of MAP. The upper bound of the hinge loss is defined and is optimized with structure SVM. AdaRank, another directly optimizing method, construct its loss function on the basis of any evaluation measure whose values are between 0 and 1. Exponential upper bound is constructed and Boosting is adopted for conduct the optimization.

In general, all the methods that directly optimize evaluation measures perform well in many ranking tasks, especially in terms of the specific evaluation measure used in the training phase. We also note that some widely used evaluation measures are designed to measure the goodness of a document ranking at all of the ranking positions. For example, the evaluation measure of NDCG can be calculated at all of the ranking positions, each reflects the goodness of the document ranking from the beginning to the corresponding position. Existing methods, however, can only directly optimize the evaluation measure calculated at a predefined ranking position. For example, the AdaRank algorithm will focus on optimizing NDCG at rank K, if its loss function is configured based on the evaluation measure NDCG@K. The information carried by the documents after the rank K are ignored, because they have no contribution to NDCG@K. Thus, it is natural to ask is it possible to devise a learning to rank algorithm that directly optimizes evaluation measures and has the ability to leverage the measures at all of the ranks?

To answer the above question, we propose a new learning to rank model on the basis of Markov decision process (MDP) [10], called MDPRank. The training phase of MDPRank considers the construction of a document ranking as a process of sequential decision making and learns the the model parameters through maximizing the cumulated rewards to all of the decisions. Specifically, the ranking of M documents is considered as a sequence of M discrete time steps where each time step corresponds to a ranking position. The ranking of documents, thus, is formalized as a sequence of M decisions and each action corresponds to selecting one document. At each time step, the agent receives the environment's state and

* Corresponding author: Jun Xu

SIGIR17, August 7–11, 2017, Shinjuku, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/3077136.3080685

chooses an action on the basis of the state. One time step later, as a consequence of the action the system transit to a new state. At each time step, the chosen of the action depends on a policy, which is a function maps from the current state to a probability distribution of selecting each possible action.

Reinforcement learning is employed to train the model parameters. Given a set of labeled queries, at each time step, the agent can receive a numerical action-dependent reward which is defined upon the evaluation measure calculated at that position. The policy gradient algorithm of REINFORCE [10] is adopted to adjust the model parameters so that expected long-term discounted rewards in terms of the evaluation measure is maximized. Thus, MDPRank can directly optimize the evaluation measure and leverages the performances at all of the ranks as rewards.

Compared with existing methods that directly optimize IR measure, MDPRank enjoys the following advantages: 1) the ability of utilizing the IR measures calculated at all of the ranking positions as supervision in training; 2) directly optimizes the IR measure on the training data without approximation or upper bounding.

To evaluate the effectiveness of MDPRank, we conducted experiments on the basis of LETOR benchmark datasets. The experimental results showed that MDPRank can outperform the state-of-the-art learning to rank models including the methods that directly optimize evaluation measures such as SVM-MAP and AdaRank.

2 RELATED WORK

In learning to rank for information retrieval, one of the most straightforward way to learn the ranking model is directly optimizing the measure used for evaluating the ranking performance. A number of methods have been developed in recent years. Some methods try to construct a continuous and differentiable approximation of the measure-based ranking error. For example, SoftRank [11] makes use of the expectation of NDCG over all possible rankings as an approximation of the original evaluation measure NDCG. SmoothRank [2] smooth the evaluation measure by approximating the rank position. Some other methods try to optimize a continuous and differentiable upper bound of the measure-based ranking error. For example, SVMMAP [17] , SVMNDCG [1] optimize the relaxation of IR evaluation measures of MAP and NDCG, respectively. Structure SVM is adopted for conducting the optimization in both of these two methods. AdaRank [14] directly optimizes the exponential upper bound of the ranking error with a Boosting procedure. [15] summarized the framework of directly optimizing evaluation measure and new algorithms can be derived and analyzed under the framework. Recently, the idea is applied to search result diversification. For example, in Xu et al. [16] and Xia et al. [13], structure Perceptron is utilized to directly optimizes the diversity evaluation measures.

In this paper we propose to directly optimize ranking evaluation measure with MDP [10], which has been widely used in variant IR applications. For example, in [6], a win-win search framework based on partially observed Markov decision process (POMDP) is proposed to model session search as a dual-agent stochastic game. In the model, the state of the search users are encoded as a four hidden decision making states. In [18], the log-based document re-ranking is also modeled as a POMDP to improve the re-ranking performances. MDP is also used for building recommender systems. For example, [9] designed an MDP-based recommendation model for taking both the long-term effects of each recommendation and the expected value of each recommendation into account. The multi-bandit also widely used for ranking [4, 8] and recommendation [5].

3 MDP FORMULATION OF LEARNING TO RANK

In supervised learning settings, we are given N labeled training queries $\left\{(q^{(n)}, X^{(n)}, Y^{(n)})\right\}_{n=1}^{N}$, where $X^{(n)} = \left\{\mathbf{x}_1^{(n)}, \cdots, \mathbf{x}_{M_n}^{(n)}\right\}$ and $Y^{(n)} = \left\{y_1^{(n)}, \cdots, y_{M_n}^{(n)}\right\}$ are the query-document feature set and the relevance label [1] set for documents retrieved by query $q^{(n)}$, respectively; and M_n is the number of the candidate documents $\{d_1, \cdots, d_{M_n}\}$ retrieved by query $q^{(n)}$.

3.1 Ranking as MDP

The process of document ranking can be formalized as an MDP, in which the construction of a document ranking can be considered as a sequential decision making where each time step corresponds to a ranking position and each action selects a document for the corresponding position. The model, referred to as MDPRank, can be is represented by a tuple $\langle S, A, \mathcal{T}, \mathcal{R}, \pi \rangle$ composed by states, actions, transition, reward, and policy, which are respectively defined as follows:

States S is a set of states which describe the environment. In ranking, the agent should know the ranking position as well as the the candidate document set that the agent can choose from. Thus, at time step t, the state s_t is defined as a pair $[t, X_t]$ where X_t is the remaining documents for ranking.

Actions A is a discrete set of actions that an agent can take. The set of possible actions depends on the state s_t, denoted as $A(s_t)$. At the time step t, $a_t \in A(s_t)$ selects a document $\mathbf{x}_{m(a_t)} \in X_t$ for the ranking position $t+1$, where $m(a_t)$ is the index of the document selected by a_t.

Transition $\mathcal{T}(S, A)$ is a function $\mathcal{T} : S \times A \rightarrow S$ which maps a state s_t into a new state s_{t+1} in response to the selected action a_t. Choosing an action a_t means removing the document $\mathbf{x}_{m(a_t)}$ out of the candidate set, as shown in the following equation:

$$s_{t+1} = \mathcal{T}([t, X_t], a_t) = [t+1, X_t \setminus \{\mathbf{x}_{m(a_t)}\}],$$

Reward $\mathcal{R}(S, A)$ is the immediate reward, also known as reinforcement. In ranking, the reward can be considered as an evaluation of the quality of the selected document. It is natural to define the reward function on the basis of the IR evaluation measures. In this paper, we define the reward received in responds to choosing action a_t as the promotion of the DCG [3]:

$$\mathcal{R}_{\mathrm{DCG}}(s_t, a_t) = \begin{cases} 2^{y_{m(a_t)}} - 1 & t = 0 \\ \frac{2^{y_{m(a_t)}} - 1}{\log_2(t+1)} & t > 0 \end{cases}, \tag{1}$$

where $y_{m(a_t)}$ is the relevance label of the selected document $d_{m(a_t)}$. Note that the calculation of each reward corresponds the DCG at each ranking position, which enables the learning of MDPRank to fully utilize the DCG values calculated at all ranking positions.

Policy $\pi(a|s) : A \times S \rightarrow [0, 1]$ describes the behaviors of the agent, which is a probabilistic distribution over the possible actions.

[1] The relevance labels are a set of ranks $\{r_1, \cdots, r_\ell\}$ and there exists a total order between the ranks: $r_\ell \succ r_{\ell-1} \succ \cdots \succ r_1$ where '$\succ$' denotes a preference relationship.

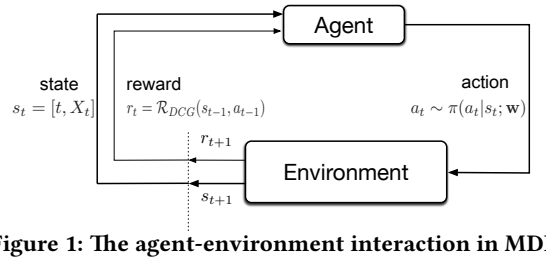

Figure 1: The agent-environment interaction in MDP.

Specifically, the policy of MDPRank calculates the probabilities of selecting each of the documents for the current ranking position:

$$\pi(a_t|s_t; \mathbf{w}) = \frac{\exp\left\{\mathbf{w}^T \mathbf{x}_{m(a_t)}\right\}}{\sum_{a \in A(s_t)} \exp\left\{\mathbf{w}^T \mathbf{x}_{m(a)}\right\}}, \tag{2}$$

where $\mathbf{w} \in \mathbb{R}^K$ is the model parameters whose dimension is same with the ranking feature.

Construction of a document ranking given a training query can be formalized as follows. Given a user query $\mathbf{q}$, the set of M retrieved documents X, and the corresponding human labels Y, the system state is initialized as $s_0 = [0, X]$. At each of the time steps $t = 0, \cdots, M-1$, the agent receives the state $s_t = [t, X_t]$, chooses an action a_t which selects the document $\mathbf{x}_{m(a_t)}$ from the document set and places it to the rank t. Moving to the next step $t+1$, the state becomes $s_{t+1} = [t+1, X_{t+1}]$. In the same time, on the basis of the human labels $y_{m(a_t)}$ for the selected documents, the agent receives immediate reward $r_{t+1} = \mathcal{R}(s_t, a_t)$. The process is repeated until all of the M documents are selected. Figure 1 illustrates the agent-environment the interaction in MDPRank.

In the online ranking/testing phase, there is no reward available because there exists no labels. The model fully trust the learned policy π and choose the action with maximal probability at each time step. Thus, the online ranking is equivalent to assigning scores $f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \mathbf{x}$ to all of the retrieved documents and sorting the documents in descending order.

3.2 Learning with policy gradient

MDPRank has parameters $\mathbf{w}$ to determine. In this paper, we propose to learn the parameters with the REINFORCE [10, 12], a widely used policy gradient algorithm in reinforcement learning. The goal of the learning algorithm is to maximize the expected long term return from the beginning:

$$J(w) = \mathbb{E}_{\mathcal{Z} \sim \pi_{\mathbf{w}}}[G(\mathcal{Z})]$$

where $\mathcal{Z} = \{\mathbf{x}_{m(a_0)}, \mathbf{x}_{m(a_1)} \cdots, \mathbf{x}_{m(a_{M-1})}\}$ is the ranking list sampled from the MDPRank model and $G(\mathcal{Z})$ is the long-term return of the sampled ranking list, which is defined as the discounted sum of the rewards:

$$G(\mathcal{Z}) = \sum_{k=1}^{M} \gamma^{k-1} r_k.$$

Note that the definition of G is identical to the IR evaluation measure DCG if $\gamma = 1$. Thus, the learning of MDPRank is actually directly optimizing the evaluation measure.

According to REINFORCE algorithm, the gradient $\nabla_{\mathbf{w}} J(\mathbf{w})$ can be calculated as

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \gamma^t G_t \nabla_{\mathbf{w}} \log \pi_{\mathbf{w}}(a_t|s_t; \mathbf{w}),$$

Algorithm 1 MDPRank learning

Input: Labeled training set $D = \{(q^{(n)}, X^{(n)}, Y^{(n)})\}_{n=1}^{N}$, learning rate η, discount factor γ, and reward function R
Output: $\mathbf{w}$

Initialize $\mathbf{w} \leftarrow$ random values
repeat
  $\Delta\mathbf{w} = \mathbf{0}$
  for all $(q, X, Y) \in D$ **do**
    $(s_0, a_0, r_1, \cdots, s_{M-1}, a_{M-1}, r_M) \leftarrow$ SampleAnEpisode($\mathbf{w}, q, X, Y, \mathcal{R}$) {Algorithm (2), and $M = |X|$}
    for $t = 0$ **to** $M-1$ **do**
      $G_t \leftarrow \sum_{k=1}^{M-t} \gamma^{k-1} r_{t+k}$ {Equation (3)}
      $\Delta\mathbf{w} \leftarrow \Delta\mathbf{w} + \gamma^t G_t \nabla_{\mathbf{w}} \log \pi(a_t|s_t; \mathbf{w})$ {Equation (4)}
    end for
  end for
  $\mathbf{w} \leftarrow \mathbf{w} + \eta\Delta\mathbf{w}$
until converge
return $\mathbf{w}$

Algorithm 2 SampleAnEpisode

Input: Parameters $\mathbf{w}$, q, X, Y, and $\mathcal{R}$
Output: An episode

Initialize $s_0 \leftarrow [0, X]$, $M \leftarrow |X|$, and episode $E \leftarrow \emptyset$
for $t = 0$ **to** $M-1$ **do**
  Sample an action $a_t \in A(s_t) \sim \pi(a_t|s_t; \mathbf{w})$ {Equation (2)}
  $r_{t+1} \leftarrow \mathcal{R}(s_t, a_t)$ {Equation (1), calculation on the basis of Y}
  Append (s_t, a_t, r_{t+1}) at the end of E
  State transition $s_{t+1} \leftarrow \left[t+1, X \setminus \{\mathbf{x}_{m(a_t)}\}\right]$
end for
return $E = (s_0, a_0, r_1, \cdots, s_{M-1}, a_{M-1}, r_M)$

where is further be estimated with Monte-Carlo sampling and shown in Algorithm 1. Algorithm 2 shows the procedure of sampling an episode for Algorithm 1. Specifically, Algorithm 1 updates the parameters via Monte-Carlo stochastic gradient ascent. At each iteration, an episode (consisting a sequence of M states, actions, and rewards) is sampled according to current policy. Then, at each time step t of the sampled episode, the model parameters are adjusted according to the gradients of the parameters $\nabla_{\mathbf{w}} \log \pi(a_t|s_t; \mathbf{w})$, scaled by the step size η, the discount rate γ^t, and the long-term return of the sampled episode starting from t, denoted as G_t:

$$G_t = \sum_{k=1}^{M-t} \gamma^{k-1} r_{t+k}. \tag{3}$$

The gradient of $\mathbf{w}$ at time step t is $\nabla_{\mathbf{w}} \log \pi(a_t|s_t; \mathbf{w})$, which the direction that most increase the probability of repeating the action a_t on future visits to state s_t, and is defined as

$$\begin{aligned} \nabla_{\mathbf{w}} \log \pi(a_t|s_t; \mathbf{w}) &= \frac{\nabla_{\mathbf{w}} \pi(a_t|s_t; \mathbf{w})}{\pi(a_t|s_t; \mathbf{w})} \\ &= \mathbf{x}_{m(a_t)} - \frac{\sum_{a \in A_t} \mathbf{x}_{m(a)} \exp\left\{\mathbf{w}^T \mathbf{x}_{m(a)}\right\}}{\sum_{a \in A_t} \exp\left\{\mathbf{w}^T \mathbf{x}_{m(a)}\right\}}. \end{aligned} \tag{4}$$

Intuitively, the setting of G_t let the parameters move most in the directions that favor actions that yield the highest return. Note that if $\gamma = 1$, G_0 is exactly the evaluation measure calculated at the final rank of the document list, i.e., DCG@M.

4 EXPERIMENTS

4.1 Experimental settings

We conducted experiments to test the performances of MDPRank using two LETOR benchmark datasets [7]: OHSUMED and Million Query track of TREC2007 (MQ2007). Each dataset consists of queries, corresponding retrieved documents and human judged labels. The possible relevance labels are relevant, partially relevant, and not relevance. Following the LETOR configuration, we conducted 5-fold cross-validation experiments on these two datasets. The results reported were the average over the five folds. In all of the experiments, we used LETOR standard features and set $\gamma = 1$ for making the algorithm to directly optimize DCG. NDCG at position of 1, 3, 5 and 10 were used for evaluation. We compared the proposed MDPRank with several state-of- the-art baselines in LETOR, including pairwise methods of RankSVM, listwise methods of ListNet, and methods that directly optimizing evaluation measures: AdaRank-MAP, AdaRank-NDCG [14], and SVMMAP [17] .

4.2 Experimental results

Table 1 and Table 2 report the performances of MDPRank and all of the baseline methods on OSHUMED and MQ2007, respectively, in terms of NDCG at the positions of 1, 3, 5, and 10. Boldface indicates the highest score among all runs. From the results, we can see that MDPRank outperformed all of the baselines, except on MQ2007 in terms of NDCG@10. We conducted significant testing (t-test) on the improvements of MDPRank over the best baseline. The experimental results indicated that the improvements on OHSUMED are significant. The results show that MDPRank is effective algorithm to directly optimize IR evaluation measures.

One advantage of MDPRank is that it has the ability of utilizing evaluation measure calculated at all of the ranking positions (the rewards) as the supervision in training. To test the effectiveness of this, we modified algorithm MDPRank so that the algorithm only utilizes the long term return of the whole episode for training, denoted as MDPRank(ReturnOnly). The modification actually controls the Algorithm 1 to execute line 7 and 8 only when $t = 0$. From the results shown in Table 1 and Table 2, we can see that MDPRank(ReturnOnly) underperformed the original MDPRank, indicating that fully utilizing the evaluation measures calculated at all of the ranking positions is effective for improving the ranking performances.

5 CONCLUSION

In this paper we have proposed to formalize learning to rank as an MDP and training with policy gradient, referred to as MDPRank. By defining the MDP rewards on the basis of IR evaluation measure and adopting the REINFORCE algorithm for the optimization, MDPRank actually directly optimizes IR measures with Monte-Carlo stochastic gradient assent. Compared with existing learning to rank algorithms, MDPRank enjoys the advantage of fully utilizing the IR evaluation measures calculated at all of the ranking positions in the training phase. Experimental results based on LETOR benchmarks show that MDPRank cam outperform the state-of-the-art baselines. The experimental results also showed that utilizing the IR evaluation measures calculated at all of the ranking positions do help to improve the performances.

Table 1: Ranking accuracies on OHSUMED dataset.

Method	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM	0.4958	0.4207	0.4164	0.4140
ListNet	0.5326	0.4732	0.4432	0.4410
AdaRank-MAP	0.5388	0.4682	0.4613	0.4429
AdaRank-NDCG	0.5330	0.4790	0.4673	0.4496
SVMMAP	0.5229	0.4663	0.4516	0.4319
MDPRank	**0.5925**	**0.4992**	**0.4909**	0.4587
MDPRank(ReturnOnly)	0.5363	0.4885	0.46949	**0.459**1

Table 2: Ranking accuracies on MQ2007 dataset.

Method	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM	0.4045	0.4019	0.4072	0.4383
ListNet	0.4002	0.4091	0.4170	**0.4440**
AdaRank-MAP	0.3821	0.3984	0.407	0.4335
AdaRank-NDCG	0.3876	0.4044	0.4102	0.4369
SVMMAP	0.3853	0.3899	0.3983	0.4187
MDPRank	**0.4061**	**0.4101**	**0.4171**	0.4416
MDPRank(ReturnOnly)	0.4033	0.4059	0.4113	0.4350

6 ACKNOWLEDGMENTS

The work was funded by the 973 Program of China under Grant No. 2014CB340401, National Key R&D Program of China under Grant No. 2016QY02D0405, the National Natural Science Foundation of China (NSFC) under Grants No. 61232010, 61472401, 61433014, 61425016, and 61203298, the Key Research Program of the CAS under Grant No. KGZD-EW-T03-2, and the Youth Innovation Promotion Association CAS under Grants No. 20144310 and 2016102.

REFERENCES

[1] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. 2008. Structured Learning for Non-smooth Ranking Losses. In *Proceedings of SIGKDD*. 88–96.
[2] Olivier Chapelle and Mingrui Wu. 2010. Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval* 13, 3 (2010), 216–235.
[3] Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and Diversity in Information Retrieval Evaluation. In *Proceedings of SIGIR*. 659–666.
[4] Nathan Korda, Balazs Szorenyi, and Shuai Li. 2016. Distributed Clustering of Linear Bandits in Peer to Peer Networks. In *Proceedings of ICML*, 1301–1309.
[5] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. 2016. Collaborative Filtering Bandits. In *Proceedings of SIGIR*. 539–548.
[6] Jiyun Luo, Sicong Zhang, and Hui Yang. 2014. Win-win Search: Dual-agent Stochastic Game in Session Search. In *Proceedings SIGIR*. 587–596.
[7] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2009. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. *Information Retrieval* (2009).
[8] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning Diverse Rankings with Multi-armed Bandits. In *Proceedings of ICML*. 784–791.
[9] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *Machine Learning Research* 6 (2005), 1265–1295.
[10] Richard S. Sutton and Andrew G. Barto. 2016. *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
[11] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. SoftRank: Optimizing Non-smooth Rank Metrics. In *Proceedings of WSDM*. 77–86.
[12] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3 (1992), 229–256.
[13] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2015. Learning Maximal Marginal Relevance Model via Directly Optimizing Diversity Evaluation Measures. In *Proceedings of SIGIR*. 113–122.
[14] Jun Xu and Hang Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proceedings SIGIR*. 391–398.
[15] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. 2008. Directly Optimizing Evaluation Measures in Learning to Rank. In *Proceedings of SIGIR*. 107–114.
[16] Jun Xu, Long Xia, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Directly Optimize Diversity Evaluation Measures: A New Approach to Search Result Diversification. *ACM TIST* 8, 3 (2017), 41:1–41:26.
[17] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A Support Vector Method for Optimizing Average Precision. In *Proceedings of SIGIR*. 271–278.
[18] Sicong Zhang, Jiyun Luo, and Hui Yang. 2014. A POMDP Model for Content-free Document Re-ranking. In *Proceedings of SIGIR*. 1139–1142.