Reinforcement Learning to Rank with Pairwise Policy Gradient

Jun Xu[1,2], Zeng Wei[3], Long Xia[4], Yanyan Lan[5], Dawei Yin[3], Xueqi Cheng[5], Ji-Rong Wen[1,2]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods
[3]Baidu Inc.; [4]School of Information Technology, York University;
[5]CAS Key Lab of Network Data Science & Technology, Institute of Computing Technology
{junxu, jrwen}@ruc.edu.cn; weizeng@baidu.com; longxia@yorku.ca; yindawei@acm.org; {yanyanlan, cxq}@ict.ac.cn

ABSTRACT

This paper concerns reinforcement learning (RL) of the document ranking models for information retrieval (IR). One branch of the RL approaches to ranking formalize the process of ranking with Markov decision process (MDP) and determine the model parameters with policy gradient. Though preliminary success has been shown, these approaches are still far from achieving their full potentials. Existing policy gradient methods directly utilize the absolute performance scores (returns) of the sampled document lists in its gradient estimations, which may cause two limitations: 1) fail to reflect the relative goodness of documents within the same query, which usually is close to the nature of IR ranking; 2) generate high variance gradient estimations, resulting in slow learning speed and low ranking accuracy. To deal with the issues, we propose a novel policy gradient algorithm in which the gradients are determined using *pairwise comparisons of two document lists sampled within the same query*. The algorithm, referred to as Pairwise Policy Gradient (PPG), repeatedly samples pairs of document lists, estimates the gradients with pairwise comparisons, and finally updates the model parameters. Theoretical analysis shows that PPG makes an unbiased and low variance gradient estimations. Experimental results have demonstrated performance gains over the state-of-the-art baselines in search result diversification and text retrieval.

CCS CONCEPTS

• **Information systems → Learning to rank**; • **Computing methodologies → Reinforcement learning**.

KEYWORDS

Learning to rank; reinforcement learning; policy gradient

ACM Reference Format:
Jun Xu[1,2], Zeng Wei[3], Long Xia[4], Yanyan Lan[5], Dawei Yin[3], Xueqi Cheng[5], Ji-Rong Wen[1,2]. 2020. Reinforcement Learning to Rank with Pairwise Policy Gradient. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401148

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401148

1 INTRODUCTION

Learning to rank, a family of machine learning techniques where the training objective is to provide the right ranking order of documents for a given query, has played a vital role in the field of information retrieval (IR) [18, 20]. In recent years, reinforcement learning (RL) [31] techniques have been applied to the task and a number of RL models have been developed [10, 12, 14, 22, 26, 39–41, 43, 44, 46], referred to as "reinforcement learning to rank" in this paper. One branch of the research formulates the process of constructing a document list for a query as sequential decision making and models it with Markov decision process (MDP). For example, in [35], the document ranking in search result diversification is modeled as an MDP in which each time step corresponds to a ranking position and each action corresponds to choosing a document for the position. Given a set of labeled training data, policy gradient [31] is utilized to learn the MDP parameters.

Despite the apparent successes, there remain limitations in these methods. Specifically, at each of its training iteration, the policy gradient algorithm (e.g., REINFORCE [31]) first samples a document list as its training instance. Then, the gradient is estimated according to the list, weighted by the absolute performance score of the list (i.e., returns). In this way, if the performance score is high, the model parameters are updated with a high probability of repeating the document selection in the future visits to similar state.

When being applied to document ranking in IR, estimating gradients in this way has two limitations. First, it ignores the relative ordering nature of IR ranking. It has been widely observed that predicting relative orders of documents is close to the nature of ranking. This observation has motivated the success of pairwise learning to rank, and clearly shows that ranking cares more about the relative goodness of a document compared to others, rather than its absolute relevance score. Moreover, in IR ranking the comparisons should only be conducted between the documents within the same query. Existing policy gradient algorithms, however, sample one document list per iteration and thus only focus on the absolute scores of the chosen actions (selection of documents). The relative orders of the documents within each query are not considered.

Second, it estimates the gradients with high variance. From the viewpoint of RL, directly leveraging the absolute performance scores (also called returns) as the gradient weights leads to high variance gradient estimations. A popular solution in traditional RL is comparing each absolute performance score to a state baseline, resulting in a relative performance score as the weight. In IR ranking, however, estimating a reasonable baseline function is very difficult, due to the extremely huge and separated state space.

In this paper, we aim to develop a new policy gradient algorithm that can fully take the relative ordering nature of IR ranking into consideration and estimate gradients with low variance. Inspired by pairwise learning to rank, we propose to develop a novel policy gradient algorithm using intra-query pairwise comparisons, referred to as Pairwise Policy Gradient (PPG). PPG utilizes MDPs as its ranking models. In its learning procedure, PPG repeats the process of sampling a pair of document lists starting from the same state (and thus within the same query), comparing these two lists in terms of their performance scores and gradient directions, and finally update the MDP parameters with the comparison results.

Theoretical analysis shows that the gradients calculated in PPG are unbiased estimations and have low variance (under some conditions), assuring it has good theoretical convergence properties and can produce fast learning. We also show that the proposed PPG is a natural generalization of REINFORCE, and it can be considered as a variation of REINFORCE with baseline.

PPG offers several advantages: ease in the implementation, theoretical soundness, efficiency in training, and high accuracy in the ranking. Empirically, we have implemented PPG for two IR ranking tasks: search result diversification and text retrieval, through configuring the MDP states, actions, rewards, state transitions, and policies accordingly. In our experiments, we found that PPG outperformed all of the state-of-the-art baselines in terms of popularly used evaluation measures (e.g., α-NDCG[7] in search result diversification and NDCG [15] in text retrieval) in both of the ranking tasks, indicating the effectiveness of PPG for document ranking. Empirical analysis showed that PPG converged faster and had lower estimation variance than REINFORCE, verified the conclusions drawn in the theoretical analysis.

2 RELATED WORK

Existing learning to rank studies can be categorized into pointwise approaches[8, 23], pairwise approaches [1, 3, 16], and listwise approaches [2, 4, 36]. Specifically, the pairwise methods consider the preference pairs composed of two documents with different relevance levels under the same query and construct classifier. However, the non-differentiable ranking metrics make it hard to optimize the evaluation measures directly.

In recent years, reinforcement learning has been applied in the IR ranking.Radlinski et al. [27] proposed two online learning bandit algorithms to learn a diverse ranking of the documents based on users clicking behaviors. Yue and Joachims [39] formalized the interactively optimizing of information retrieval systems as a dueling bandit problem, called Dueling Bandit Gradient Descent (DBGD). In [13], DBGD was further improved so that the click data can be used to judge the user preference of the document rankings. In [17], a cascading bandits model is proposed to identify the K most attractive document for users.MDP has also shown its effectiveness in learning to rank. In [40], the process of document ranking is formalized with MDP and solved with the classic policy gradient algorithm of REINFORCE. Similar MDP configurations are used to model the sequential document selection process in search result diversification [11, 35] and multi-page search [41]. The query change model [37] formalizes the problem of session search as an MDP. The win-win search framework models session search as a dual-agent stochastic game, on the basis of partially observed Markov decision process (POMDP) [22]. In [42], a log-based document re-ranking algorithm is proposed, also based on POMDP. DRM [24] makes use of deep RL to deal with the complex ranking problems in which both the user preferred document order and display position order for result presentation is considered.

RL has also been widely used for the IR tasks beyond document ranking. For example, Shi et al. [30] use reinforcement learning for commodity search. Li et al. [19] model collaborative filtering with bandit. Shani et al. [29] design an MDP-based recommendation model for taking both the long-term effects of each recommendation and the expected value of each recommendation into account. Lu and Yang [21] propose POMDP-Rec, a neural-optimized POMDP algorithm, for building a collaborative filtering recommender system. In [44], the recommendation is modeled with MDP and deep Q-learning is used to conduct the optimization. In [32], the process of ranking is modeled with generative adversarial networks (GANs) and solved with reinforcement learning, called IRGAN. IRGAN can be used for both document ranking and item recommendation.

3 REINFORCEMENT LEARNING TO RANK

Given N labeled training queries $\{(Q^{(n)}, X^{(n)}, Y^{(n)})\}_{n=1}^{N}$, where $X^{(n)} = \{d_1^{(n)}, \cdots, d_{M_n}^{(n)}\}$ and $Y^{(n)} = \{y_1^{(n)}, \cdots, y_{M_n}^{(n)}\}$ are the sets of candidate documents and labels associated with query $Q^{(n)}$, respectively. The ranking models aim at constructing a document list by putting the relevant documents to the top of the list.

3.1 MDP formulation of ranking

MDP is a powerful mathematical model used to describe an interaction system between the environment and the agent. The MDP model has been used to model the process of document ranking. Formally, the MDP formulation of the document ranking considers the construction of a document list as sequential decision making. In the MDP, each time step t ($t = 0, 1, \cdots$) corresponds to a position in the document list. Specifically, the action $a_t \in \mathcal{A}(s_t)$ denotes to select the document $d_{m(a_t)}$ from the candidates and move it to the rank $t+1$, and the policy $\pi(a|s;\theta)$ defines a probabilistic distribution over the available candidate documents. $\mathcal{A}(s_t)$, $m(a_t)$, and θ denote the set of available actions, index of the document corresponding to a_t, and the model parameters, respectively.

Construction of a document list with the MDP can be described as: given a user query Q and the set of retrieved M documents $X = \{d_1, \cdots, d_M\}$, the MDP state is initialized as $s_0 = \mathcal{S}(Q, X)$, where $\mathcal{S}$ is the state initialization function. At each of the time steps $t = 0, \cdots, M-1$, the agent observes the state s_t, calculates the policy $\pi(\cdot|s_t;\theta)$, and chooses an action a_t which selects the document $d_{m(a_t)}$ from the document set and places it to the rank $t+1$. The user feedbacks the reward $\mathcal{R}(s_t, a_t)$ to measure the selected document. Moving to the next time step $t+1$, the state becomes $s_{t+1} = \mathcal{T}(s_t, a_t)$, where $\mathcal{T}$ is the state transition function. The process is repeated until all of the M documents are ranked.

3.2 Learning with policy gradient

RL methods are widely used to determine the MDP model parameters θ (may consist of the parameters in the functions of policy

π, state initialization $\mathcal{S}$, and the state transition $\mathcal{T}$ etc.). With the labeled queries, the MDP environment can emit an immediate numerical reward $r_{t+1} = \mathcal{R}(s_t, a_t)$[1] for each issued action a_t. With the rewards, the policy gradient algorithms such as REINFORCE can be employed to learn the parameters θ. Specifically, the policy gradient algorithms employ stochastic gradient ascent iterations for conducting the optimization. At each iteration, a document list (episode) $\tau = (S_0, A_0, R_1, \cdots, S_{M-1}, A_{M-1}, R_M) \sim \pi(\cdot|\cdot;\theta)$ is sampled, where S_t, A_t, and R_{t+1} are the observed state, the chosen action, and the received reward, respectively. Then, at each t, the gradient of the parameters θ is estimated as:

$$\Delta\theta = G_t \nabla \log \pi(A_t|S_t;\theta), \tag{1}$$

where G_t is the long-term return starting from t: $G_t = \sum_{k=1}^{M-t} \gamma^{k-1} R_{t+k}$, where $\gamma \in (0, 1]$ is the discount factor.

3.3 Example applications: search result diversification and text retrieval

The MDP formulation of ranking has been applied to the ranking tasks of search result diversification and text retrieval and promising results have been presented [35, 40]. The configurations of the MDPs, including the definitions of time step, state, state transition, reward, policy, and the gradient of policy, are shown in Table 1. Please refer to [35, 40] for more details.

3.4 Problem Analysis

Though significant progresses have been achieved, the MDP formulation of document ranking still has limitations. More specifically, the gradient in Equation (1) consists of two terms: the gradient $\nabla \log \pi(A_t|S_t;\theta)$ which points to the direction that most increases the probability of repeating the action A_t on future visits to the state S_t, and the long term return G_t which lets the parameters move most in the directions that favor actions that yield the highest return.

When being directly applied in IR ranking, the gradient setting has two limitations: (1) fails to consider the intra-query relative ordering nature of IR ranking, and (2) estimates the gradients with high variance.

3.4.1 The intra-query relative ordering nature. The aim of document ranking in IR is to come up with optimal orderings of documents retrieved by queries. Compared with traditional machine learning tasks such as classification or regression, document ranking is unique in that: (1) ranking does not care much about the exact relevance score that each document gets, but cares more about the relative goodness of a document compared to others; (2) the existence of queries poses a further restriction on the comparisons: only the documents retrieved by the same query can be compared with each other. The cross-query comparisons are not meaningful because the two documents will not appear in the same list. Existing policy gradient approaches, however, fail to capture both of these two important properties in IR ranking.

(1) At each rank position t, the gradient weight G_t in Equation (1) is calculated as the absolute goodness of the chosen document $d_{m(A_t)}$, calculated based on a sampled document list. The goodness of choosing other documents at the same position t (and at the same state) is not taken into consideration. Thus, with these gradients and weights, the policy gradient still cares about the absolute document scores, not the relative document orders.

(2) When being applied to document ranking [35, 40], the final gradients are the linear combinations of the gradient directions at all positions of all sampled document lists. The combination weights (G_t's) are usually set as the absolute performance scores based on DCG or α-DCG, without considering which query they come from. In another word, all of the queries are equally treated. However, it have been observed in IR that there exists high variance in performance among the queries [6]. Some queries are in nature more difficult than others. The reasons could be, for example, the query may have very few relevant documents available in its candidates, or there exists a big semantic gap between the query and the relevant documents. The performance of the optimal ranking for a difficult query may be lower than the performance of a suboptimal ranking for an easy query, as the examples shown in Figure 1. Since the easy queries always generate document lists with high-performance scores while policy gradient treats them equally, the trained ranking model may be biased to easy queries.

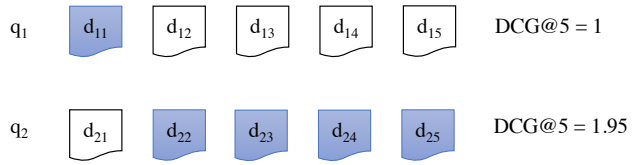

Figure 1: DCG@5 of the optimal document list for a difficult query q_1 and a suboptimal document list for an easy query q_2. The shaded icons denote the relevant documents.

3.4.2 High variance gradient estimation. In policy gradient, it has been widely observed that directly using returns as the gradient weights leads to high variance gradient estimation [31]. As the solutions for traditional RL tasks, policy gradient with baseline and actor-critic have been proposed for reducing the estimation variance. In these methods, the absolute return of each action is compared to the baseline of the state. Thus, the weights become relative returns. Intuitively, in some states, all actions have high values and high baselines are needed to differentiate the higher valued actions from the less highly valued ones; in other states all actions have low values and low baselines are appropriate. Usually, the baseline is estimated using the historically estimated episodes.

In IR ranking, however, it is very difficult (if not impossible) to estimate reasonable baselines because the state space is huge and separated. In the MDP model for ranking, the ranked document lists are usually involved in the state. Suppose that a query is associated with M documents (on average $M \approx 150$ in OHSUMED), the number of possible document lists per query is $M!$. The size of the state space makes it impossible to sample enough data to estimate the baseline values. Moreover, in training N different training queries are provided. Since each query retrieves its own set of documents for ranking, the whole state space is separated into N disjoint subsets. The document lists sampled in one query cannot be used to learn the baselines in another query.

[1]The reward can be calculated, for example, on the basis of IR evaluation measures of DCG, α-DCG, S-recall etc.

Table 1: MDP configurations for search result diversification and text retrieval.

MDP configuration	search result diversification [35]	text retrieval [40]
state at time t: s_t	$s_t = [Z_t, X_t, \mathbf{h}_t]$, where Z_t is ranked doc list, $\mathbf{h}_t$ is latent vector	$s_t = [t, X_t]$
initial state s_0	$s_0 = \mathcal{S}(Q, X) = [\emptyset, X, \sigma(\mathbf{V}_q\mathbf{q})]$, where $\mathbf{q}$ is query embedding	$s_0 = \mathcal{S}(Q, X) = [0, X]$, where X contains all docs
state transition $\mathcal{T}(s_t, a_t)$	$\mathcal{T}(s_t, a_t) = [Z_t \cup \{d_{m(a_t)}\}, X_t \setminus \{d_{m(a_t)}\}, \sigma(\mathbf{V}\mathbf{x}_{m(a_t)} + \mathbf{W}\mathbf{h}_t)]$, where $\mathbf{x}_{m(a_t)}$ is doc embedding, and $\mathbf{V}$, $\mathbf{W}$ are parameters	$\mathcal{T}(s_t, a_t) = [t+1, X_t \setminus \{d_{m(a_t)}\}]$
reward $\mathcal{R}(s_t, a_t)$	$\mathcal{R}(s_t, a_t) = \alpha$-DCG[t+1]$-\alpha$-DCG[t], where α-DCG[0] = 0	$\mathcal{R}(s_t, a_t) = \begin{cases} 2^{y_{m(a_t)}} - 1 & t = 0 \\ (2^{y_{m(a_t)}} - 1)/\log_2(t+1) & t > 0 \end{cases}$
policy $\pi(a_t \mid s_t; \theta)$	$\pi(a_t \mid s_t) = \frac{\exp\left\{\mathbf{x}_{m(a_t)}^T \mathbf{U}\mathbf{h}_t\right\}}{\sum_{a \in A(s_t)} \exp\left\{\mathbf{x}_{m(a)}^T \mathbf{U}\mathbf{h}_t\right\}}$, where $\mathbf{U}$ is the parameter matrix	$\pi(a_t \mid s_t; \theta) = \frac{\exp\left\{\theta^T \phi(Q, d_{m(a_t)})\right\}}{\sum_{a \in A(s_t)} \exp\left\{\theta^T \phi(Q, d_{m(a)})\right\}}$, where $\phi(Q, d_{m(a_t)})$ is ranking feature vector
Gradient $\nabla \log \pi(a \mid s)$	refer to [35] for details	$\phi(Q, d_{m(a_t)}) - \sum_{a \in A(s_t)} \pi(a \mid s_t; \theta)\phi(Q, d_{m(a)})$

4 OUR APPROACH: PPG

To deal with the issues and inspired by the intra-query preference pair generation mechanism in pairwise learning to rank, in this section we propose to combine the intra-query pairwise comparisons with conventional policy gradient algorithm of REINFORCE, achieving a new algorithm tailored for document ranking, referred to as pairwise policy gradient (PPG).

4.1 Policy gradient with pairwise comparison

4.1.1 Gradient representation with pairwise comparisons. First, we show theoretically that the gradient of the ranking objective can be represented in the form of pairwise comparisons. Formally, the goal of the learning algorithm is to maximize the expected long term return (performance of the list) from the beginning:

$$J(\theta) = v_\pi(s_0) = \mathbb{E}_{\tau \sim \pi}[G_0], \tag{2}$$

where $v_\pi(s_0)$ is the expected return starting from $t = 0$, $\tau = \{s_0, a_0, r_1, \cdots, s_{M-1}, a_{M-1}, r_M\}$ (corresponds to the document list of $\{d_{m(a_0)}, \cdots, d_{m(a_{M-1})}\}$) is the episode sampled according to current policy π, and G_0 is the long-term return of the episode starting from time step $t = 0$: $G_0 = \sum_{k=1}^{M} \gamma^{k-1} r_k$.

The policy gradient method calculate the gradient based on the Policy Gradient Theorem which can be represented as ([31], Chapter 13.2):

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s, a) \cdot \nabla \pi(a|s; \theta) \tag{3}$$

where $\mu(s)$ is on-policy distribution over the state s under π, $q_\pi(\cdot, \cdot)$ is the Q-function whose value is the discounted total reward expected after performing the action in the state and then following π, and $\nabla \pi(\cdot|\cdot; \theta)$ is the partial derivatives of $\pi(\cdot|\cdot; \theta)$ w.r.t. θ.

In retrieval scenario, the search engine constructs the ranking list given a query, which can reflect the preference of each document. Just as the equation 3 shown, the traditional policy gradient methods measure the sampled action by the absolute accumulative rewards gained in future. However, the distributions of the candidate documents various over different queries. The absolute reward may lead to unfairness to different queries.

Inspired by the pairwise ranking methods, which optimize the rank model based on the preference document pairs. We consider the search engine samples two action at once to construct the "preference pair". We modify the policy gradient method to utilize the difference of rewards gained by these two document to optimize the rank model and prove that the gradient of $J(\theta)$ can be expressed in the form of pairwise comparisons, as shown in Theorem 4.1.

Theorem 4.1. *The gradient of $J(\theta)$ in Equation (2) can be represented as*

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \sum_b (q_\pi(s, a) - q_\pi(s, b)) \cdot (\pi(b|s; \theta)\nabla \pi(a|s; \theta) - \pi(a|s; \theta)\nabla \pi(b|s; \theta)),$$

In Theorem 4.1, the gradient of the policy is represented in the form of pairwise comparisons between actions a and b. The proof of Theorem 4.1 can be found in Section A.1.

4.1.2 Gradient estimation with Monte Carlo sampling. Theorem 4.1 gives a new analytic expression for $\nabla J(\theta)$. Following the practices in deriving the REINFORCE algorithm, we estimate $\nabla J(\theta)$ with Monte Carlo sampling.

Firstly, the search engine interacts with the user based on current policy $\pi(a|s, \theta)$, then the sampled state satisfies $S_t \sim \mu(s)$

$$\nabla J(\theta) \propto \mathbb{E}_\pi \left[\sum_a \sum_b (q_\pi(S_t, a) - q_\pi(S_t, b)) \cdot (\pi(b|S_t; \theta)\nabla \pi(a|S_t; \theta) - \pi(a|S_t; \theta)\nabla \pi(b|S_t; \theta)) \right]$$

Then we utilize the multipliers $\pi(a_t|S_t, \theta)$ and $\pi(b_t|S_t, \theta)$, and sample the action A_t and B_t follow the current policy. Thus

$$\begin{aligned} \nabla J(\theta) &\propto \mathbb{E}_\pi \left[\sum_a \sum_b \pi(a|S_t; \theta)\pi(b|S_t; \theta) \cdot (q_\pi(S_t, a) - q_\pi(S_t, b)) \cdot \left(\frac{\nabla \pi(a|S_t; \theta)}{\pi(a|S_t; \theta)} - \frac{\nabla \pi(b|S_t; \theta)}{\pi(b|S_t; \theta)} \right) \right] \\ &= \mathbb{E}_\pi \left[(q_\pi(S_t, A_t) - q_\pi(S_t, B_t)) \cdot (\nabla \log \pi(A_t|S_t; \theta) - \nabla \log \pi(B_t|S_t; \theta)) \right], \end{aligned} \tag{4}$$

Algorithm 1 Pairwise Policy Gradient (PPG)

INPUT: Training set $D = \{(Q^{(n)}, X^{(n)}, Y^{(n)})\}_{n=1}^{N}$, learning rate η, discount factor γ, and reward function R

Initialize $\theta \leftarrow$ random values
repeat
  $\Delta\theta \leftarrow \mathbf{0}$
  for all $(Q, X, Y) \in D$ **do**
    Initial state $S = \mathcal{S}(Q, X)$ {Init state with Q}
    for $t = 0$ **to** $M - 1$ **do**
      $\tau^A = \{S, A_t, R^A_{t+1}, S^A_{t+1}, \cdots, S^A_{M-1}, A_{M-1}, R^A_M\} \sim \pi(\theta)$
      $G^A \leftarrow \sum_{k=1}^{M-t} \gamma^{k-1} R^A_{t+k}$ {long term return of τ^A}
      $\tau^B = \{S, B_t, R^B_{t+1}, S^B_{t+1}, \cdots, S^B_{M-1}, B_{M-1}, R^B_M\} \sim \pi(\theta)$
      $G^B \leftarrow \sum_{k=1}^{M-t} \gamma^{k-1} R^B_{t+k}$ {long term return of τ^B}
      $\Delta\theta \leftarrow \Delta\theta + (G^A - G^B) \cdot \big(\nabla \log \pi(A_t|S_t;\theta) - \nabla \log \pi(B_t|S_t;\theta)\big)$ {according to Equation (5)}
      $S \leftarrow \begin{cases} S^A_{t+1} & G^A \geq G^B \\ S^B_{t+1} & \text{otherwise} \end{cases}$
    end for
  end for
  $\theta \leftarrow \theta + \eta\Delta\theta$
until converge
return θ

where the state-action value function $q_\pi(S_t, A_t)$ is the expected discount reward:

$$\begin{aligned} q_\pi(S_t, A_t) &= \mathbb{E}[R^A_t + \gamma R^A_{t+1} + \cdots + \gamma^{(T-t)} R^A_T | S_t, A_t] \\ &= \mathbb{E}[G^A_t | S_t, A_t]. \end{aligned}$$

$q_\pi(S_t, B_t)$ can be calculated in a similar way. In Equation (4), the first equation holds because the overall gradient is a sum over states weighted by how often the states occur under the target policy π. Thus we can get rid of $\mu(s)$ because the state will be encountered in these proportions following the π; the second equation introduces a weighting $\pi(a|S_t;\theta)\pi(b|S_t;\theta)$ without changing the equality by multiplying and dividing it; and the third equation replaces a by action A_t and b by action B_t, both sampled following the policy $\pi(\cdot|S_t;\theta)$.

4.1.3 Learning with stochastic gradient ascent. Using the gradient in Equation (4) to instantiate the generic stochastic gradient ascent, we get the gradient at each position t:

$$\Delta\theta = (G^A_t - G^B_t)\big(\nabla \log \pi(A_t|S_t;\theta) - \nabla \log \pi(B_t|S_t;\theta)\big). \quad (5)$$

The equation indicates that the gradient of the policy parameters can be estimated with pairwise comparisons: given a query and starting from S_t, the algorithm samples two episodes τ^A and τ^B. The unbiased gradient of the policy, then, can be calculated based on the pairwise comparisons between the two lists.

Algorithm 1 shows the proposed pairwise policy gradient (PPG) algorithm. The process of sampling the episodes and choosing the next states in Algorithm 1 is illustrated in Figure 2. We can see that PPG successfully introduces intra-query pairwise comparisons into policy gradient and captures the intra-query relative ordering nature of document ranking. Note that the two lists in a pair start from the same state (under a query), which assures only intra-query comparisons are allowed in PPG.

The time complexity of the PPG training algorithm is of $O(TNM(M+1))$, where T is the number of training iterations, N is the number of queries in training data, and M is the average number of labeled documents per query.

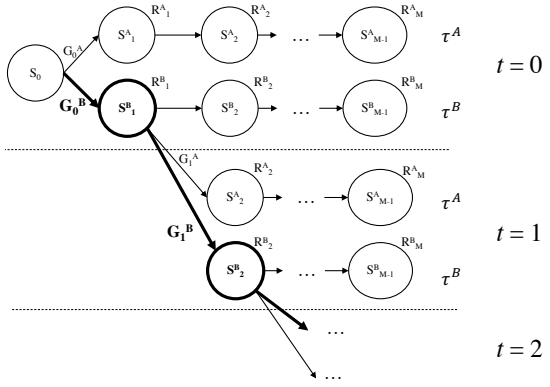

Figure 2: Illustration of the sampling process in PPG. The highlighted arrows and circles shows the chosen actions and the resulting next states. Given the initial state S_0 and ranking position $t = 0$, PPG samples two episodes τ^A and τ^B, and conducts the intra-query pairwise comparison. Then, at $t = 1$, the system follows the winner episode one step (τ^B in the example because $G^A_0 < G^B_0$) and moves to a new state S^B_1. Again, starting from S^B_1, PPG samples two new episodes for another comparison. The process continues until $t = M - 1$. Note that for an M-size candidate set, PPG will sample $2M + 2(M-1) + \cdots + 2 = M \cdot (M+1)$ times.

4.2 Discussions

4.2.1 Bias and variance of the gradients. Theorem 4.1 shows the form of the gradient $\nabla J(\theta)$ and Equation (4) derives an unbiased Monte Carlo sampling. Therefore Algorithm 1 makes an unbiased estimation of $\nabla J(\theta)$, guaranteeing that it will converge asymptotically to an optimum.

Moreover, the intra-query pairwise comparisons have the ability to estimate the gradients with low variance (under some conditions), as shown in the following Theorem 4.2.

Theorem 4.2. *Given state $s \sim \mu_\pi$ where μ_π is is on-policy distribution under π, and given two actions $a \sim \pi(\cdot|s;\theta)$ and $b \sim \pi(\cdot|s;\theta)$. Considering the following two representations of the gradient:*

$$\begin{aligned} \mathbf{g}_1 &= (q_\pi(s,a) - q_\pi(s,b)) \cdot (\nabla \log \pi(a|s;\theta) - \nabla \log \pi(b|s;\theta)), \\ \mathbf{g}_2 &= q_\pi(s,a) \cdot \nabla \log \pi(a|s;\theta) + q_\pi(s,b) \cdot \nabla \log \pi(b|s;\theta). \end{aligned}$$

The variances of $\mathbf{g}_1$ and $\mathbf{g}_2$ satisfy

$$\mathrm{Var}(\mathbf{g}_1) \leq \mathrm{Var}(\mathbf{g}_2),$$

if $q_\pi(\cdot,\cdot) \geq 0$ and

$$\mathbb{E}_{\mu_\pi,\pi}\Big[(q(s,a) - q(s,b)) \cdot \Big(q(s,b)\|\nabla \log \pi(a|s;\theta)\|^2 - q(s,a)\|\nabla \log \pi(b|s;\theta)\|^2\Big)\Big] \geq 0,$$

where $\mathrm{Var}(\mathbf{g}) = \mathrm{tr}(\mathrm{cov}(\mathbf{g},\mathbf{g}))$ is the trace of the covariance matrix, and 'tr' and 'cov' denote trace and covariance, respectively.

Proof of the theorem is given in Section A.2. Note that $\mathbf{g}_1$ is the gradient used in PPG and $\mathbf{g}_2$ equals to the gradient used in REINFORCE[2].

Theorem 4.2 indicates that PPG has the ability to reduce variance in gradient estimation when the conditions being satisified. In stochastic optimization, the estimation with low variance usually leads to fast in learning and accurate in ranking.

The first condition $q_\pi(\cdot,\cdot) \geq 0$ can be satisfied naturally in IR, because the commonly used IR evaluation measures such as DCG and α-DCG are nonnegative and increases monotonically with the ranking positions. The second condition can be explained as follows: under the π and μ_π, the state-action value and the norm of gradient (multiplied by another state-action value) are positively correlated.

4.2.2 Relation with pairwise learning to rank. PPG is inspired by the intra-query pairwise comparison mechanism in pairwise learning to rank. In that sense, they share a number of common advantages such as capturing the relative ordering nature of ranking and avoiding cross-query comparisons. However, PPG also has several striking differences from pairwise learning to rank.

First, the criterion for "preference" are different. In pairwise learning to rank, a document is preferred because it has a higher relevance label. In PPG, however, an action (choosing a document) is preferred because selecting the document at current state will get higher long term return. In this sense, PPG looks at the future effects and current state when evaluating a document, which is a more reasonable criteria to get optimal rankings.

Second, the times for generating the pairs are different. In pairwise learning to rank, the algorithms generate all of the pairs before the learning starts. PPG, on the other hand, generate the pairs at each of the training iterations with an "online" manner. Therefore, it is possible for PPG to dynamically generate the most valuable pairs at each of the training iterations.

4.2.3 Relation with REINFORCE and REINFORCE with baseline. PPG is derived under the policy gradient framework and it is a nature generalization of REINFORCE. Comparing the estimated gradients by PPG (Equation (5) and line 11 of Algorithm 1) and that of by REINFORCE (Equation (1)), we can see that PPG automatically degenerates to REINFORCE by setting τ^A (or τ^B) to an empty episode. For example, by setting τ^B to empty, the gradient of PPG becomes the gradient of REINFORCE:

$$\begin{aligned}\Delta\theta =&(G_t^A - G_t^B)(\nabla \log \pi(A_t|S_t;\theta) - \nabla \log \pi(B_t|S_t;\theta))\\ =&(G_t^A - 0)(\nabla \log \pi(A_t|S_t;\theta) - \mathbf{0}) = G_t^A \nabla \log \pi(A_t|S_t;\theta).\end{aligned}$$

PPG can also be considered as a variation of REINFORCE with baseline. REINFORCE with baseline compares the action value G_t to the baseline value of S_t and its gradient can be written as:

$$\Delta\theta = (G_t - b(S_t))\nabla \log \pi(A_t|S_t;\theta),$$

where $b(S_t)$ is the base line function that does not vary with action. From this viewpoint, the gradient of PPG can be decomposed as the sum of two gradients with baselines:

$$\begin{aligned}\Delta\theta =&(G_t^A - G_t^B)(\nabla \log \pi(A_t|S_t;\theta) - \nabla \log \pi(B_t|S_t;\theta))\\ =&(G_t^A - G_t^B)\nabla \log \pi(A_t|S_t;\theta) + (G_t^B - G_t^A)\nabla \log \pi(B_t|S_t;\theta).\end{aligned}$$

The first term $(G_t^A - G_t^B)\nabla \log \pi(A_t|S_t;\theta)$ is the gradient estimated based on τ^A, using G_t^B as the baseline; and second term $(G_t^B - G_t^A)\nabla \log \pi(B_t|S_t;\theta)$ is the gradient estimated based on τ^B, using G_t^A as the baseline. Since policy gradient with baseline can reduce the variance, it is intuitive that PPG can also produce low variance estimations.

5 EXPERIMENTS

We conducted experiments to test the performances of PPG. The source code of PPG can be found at https://github.com/wzeng316/PPG-Rank.

5.1 Experiments on search result diversification

We tested the performances of PPG on the ranking task for search result diversification. Specifically, following the practices in [35], the experiments were conducted on the combination of four TREC benchmark datasets (TREC Web Track 2009 ~ 2012). The dataset consists of 200 queries and in total about 45,000 labeled documents. Each query includes several subtopics identified by the TREC annotators. The document relevance labels are made at the subtopic level and the labels are binary. The candidate documents were retrieved from the ClueWeb09 Category B data collection, which is comprised of 50 million English web documents. Porter stemming, tokenization, and stop-words removal were applied to the documents as preprocessing. The queries were randomly split into five folds and we conducted 5-fold cross-validation experiments. The results reported were the averages over the five trials.

We compared the proposed PPG with state-of-the-art baselines in search result diversification, including the heuristic methods of MMR [5], xQuAD [28], and PM-2 [9]; and the learning methods of SVM-DIV [38], R-LTR [45], PAMM [33], NTN-DIV [34], and MDP-DIV [35].

In MDP-DIV and PPG, the reward is calculated based on α-DCG and the discount factor $\gamma = 1$. The evaluation metrics include α-NDCG, S-recall, and ERR-IA, at the positions of 5 and 10.

Table 2 reports the ranking accuracies of PPG and all of the baseline methods. Boldface indicates the highest scores in all runs. From the results, we can see that PPG outperformed all baselines in terms of all evaluation metrics, demostrating the effectiveness of PPG in the task of search result diversification. We conducted significant tests (t-test) on the improvements of PPG over the best baseline MDP-DIV. The results showed that most of the improvements were significant (p-value < 0.05 and denoted with '$*$').

5.2 Experiments on text retrieval

We also conducted experiments on the task of text retrieval. The experiments were conducted on three traditional learning to rank benchmark datasets: OHUSMED, MQ2008 [25], and MSLR-Web10K. Each dataset consists of queries, corresponding retrieved documents and human judged labels, and the statistics over the three dataset

[2]Note that for making fair comparisons, both $\mathbf{g}_1$ and $\mathbf{g}_2$ are based on two sampled actions a and b. $\mathbf{g}_2$ is an estimation of the original REINFORCE gradient $q_\pi(s,a) \cdot \nabla \log \pi(a|s;\theta)$ based on two sampled actions.

Table 2: Performance comparison on TREC web track datasets

Method	α-NDCG@5	α-NDCG@10	S-rescall@5	S-rescall@10	ERR-IA@5	ERR-IA@10
MMR	0.2753	0.2979	0.4388	0.5151	0.2005	0.2309
xQuAD	0.3165	0.3941	0.4933	0.6043	0.2314	0.2890
PM-2	0.3047	0.3730	0.4910	0.6012	0.2298	0.2814
SVM-DIV	0.3030	0.3730	0.4910	0.6012	0.2298	0.2814
R-LTR	0.3498	0.4132	0.5397	0.6511	0.2521	0.3011
PAMM	0.3712	0.4327	0.5561	0.6612	0.2619	0.3029
NTN-DIV	0.3962	0.4577	0.5817	0.6872	0.2773	0.3285
MDP-DIV	0.4493	0.4924	0.5718	0.6826	0.3485	0.3712
PPG	**0.4799***	**0.5122***	**0.6099***	**0.6928**	**0.3727***	**0.3914***

Table 3: Statistics of L2R datasets

Method	#Query	#LabeledDoc	#Feature	# LabelLevel
OHSUMED	106	16,140	45	3
MQ2008	784	9,360	46	3
MSLR-Web10K	10,000	1,200,192	136	5

Table 4: Performance comparison on LETOR OHSUMED.

Method	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM	0.4958	0.4958	0.4958	0.4140
RankNet	0.4785	0.4516	0.4464	0.4367
ListNet	0.5326	0.4732	0.4432	0.4410
AdaRank	0.4790	0.3730	0.4673	0.4496
MDPRank	0.5743	0.5045	0.4784	0.4558
PPG	**0.5771**	**0.5218**	**0.4911**	**0.4664**

Table 5: Performance comparison on LETOR MQ2008.

Method	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM	0.3626	0.4286	0.4695	0.2279
RankNet	0.3202	0.3984	0.4408	0.2094
ListNet	0.3754	0.4324	0.4747	0.2303
AdaRank	0.3826	0.4420	0.4821	0.2307
MDPRank	0.3827	0.4420	0.4881	0.2327
PPG	**0.3877**	**0.4511**	**0.4910**	**0.2455**

Table 6: Performance comparison on MSLR-WEB10K.

Method	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM	0.3447	0.3589	0.3687	0.3881
RankNet	0.3768	0.3862	0.3942	0.4105
ListNet	0.3878	0.3879	0.3969	0.4135
AdaRank	0.3437	0.3272	0.3337	0.3475
MDPRank	0.4182	0.4052	0.4082	0.4189
PPG	**0.4230**	**0.4104**	**0.4144**	**0.4254**

are shown in Table 3. In all of the experiments, we conducted 5-fold cross-validation experiments on these datasets. The results reported were the average over the five folds. For OHUSMED and MQ2008 dataset, we used the provided standard features and the linear score function. Specifically, for MSLR-Web10K dataset, we normalize each provided feature by the mean/standard deviation.

We compared the proposed PPG to the traditional learning to rank baselines, including RankSVM [16], RankNet [2], ListNet [4], AdaRank [36], and MDPRank [40].

In both MDPRank and PPG, the rewards are calculated based on DCG and the discount factor $\gamma = 1$. As for evaluation measures, NDCG at the positions of 1, 3, 5 and 10 were used for evaluation. Table 4, Table 5, and Table 6 report the performances of PPG and the baselines on OHSUMED, MQ2008 and MSLR-Web10K, respectively. Boldface indicates the highest scores among all runs. From the results, we can see that PPG outperformed all the baselines, including the traditional learning to rank methods and reinforcement learning-based method, on both datasets in terms of all of the evaluation measures. The results showed the effectiveness of PPG for the task of text retrieval.

5.3 Analysis of the experimental results

In this section, we conducted experiments to investigate how PPG works and outperformed the baselines, using the results on the first fold of the combined TREC Web Track data as example.

5.3.1 Convergence. Theorem 4.1 and Theorem 4.2 show that PPG makes unbiased and low variance estimations, guaranteeing to converge to an optimum fast. We conducted experiments to verify the theoretical conclusions. Specifically, we compared the convergence curves of PPG with the baseline method of MDP-DIV and MDPRank (using REINFORCE in its learning).

Figure 3 illustrates the learning curves where the x-axis is number of training epochs and y-axis is the average rewards received. The curves in Figure 3 show that PPG converged much faster than MDPRank and MDP-DIV. Moreover, PPG converged to a better optimum. The results verified that: (1) the unbiased estimations made PPG to converge to an optimum; (2) the low variance estimations made PPG to converge fast.

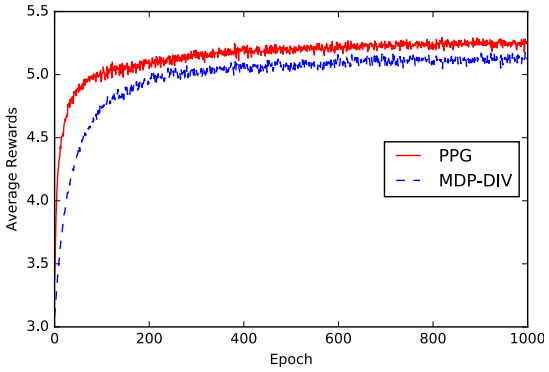

Figure 3: Learning curves of PPG and MDP-DIV.

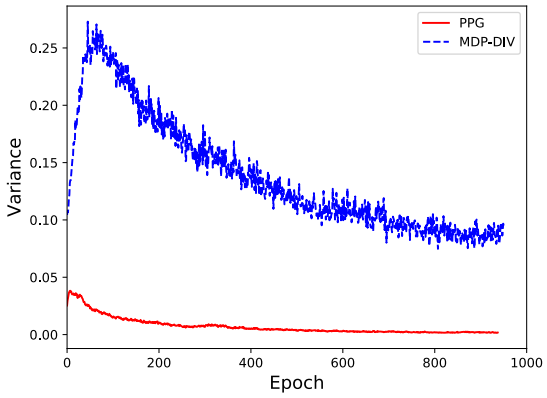

Figure 4: Variance curves of PPG and MDP-DIV.

5.3.2 Reducing the variance. Theorem 4.2 shows that PPG can produce low variance estimations, under some conditions. We conducted experiments to directly compare the variances of the estimations by PPG and MDP-DIV. Specifically, we trained diverse ranking models with PPG and MDP-DIV. At each iteration, the gradient vectors calculated on all of the sampled document lists were recorded and the variance (trace of the covariance matrix) was calculated. Figure 4 illustrated the variance curves by PPG and MDP-DIV. We can see that the variance curve by PPG is much lower than that of by MDP-DIV at all of the training epochs. The results verified the conclusion drawn in Theorem 4.2 on real applications. It also indicated that the conditions in Theorem 4.2 is reasonable and can be well satisfied in real applications.

5.3.3 Improving difficult queries. The analysis in Section 3.4 shows that the document ranking models trained by REINFORCE may be biased to easy queries, because of the high variance in performance among queries. We conducted experiments to show whether the bias can be alleviated in PPG.

Specifically, we made statistics on the number of relevant documents per query in the training data. The queries are clustered into different groups based on percentage of relevant documents among the candidates. Intuitively, more number of relevant documents makes the query easier. Figure 5 shows the distribution of the query groups. In the figure, for example, '0% ~ 20%' is the group of queries

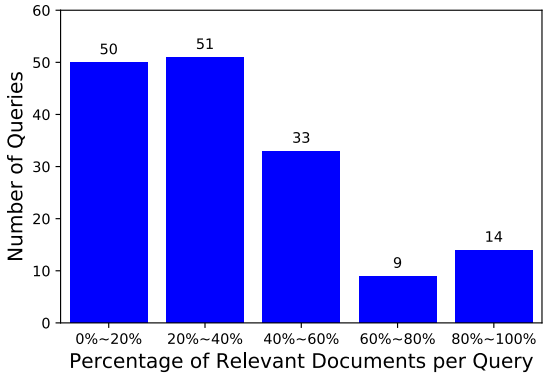

Figure 5: Distribution of training queries w.r.t. different percentages of relevant documents.

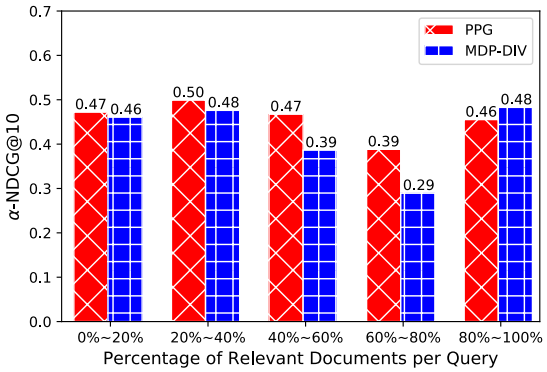

Figure 6: Performance comparison in terms of α-NDCG@10 for different query groups.

whose percentage of relevance documents per query are between 0% and 20%. We can see that the numbers of relevant documents per query really vary from query to query, indicating the high variance in performance among the queries in real data.

Next we evaluated the accuracies of PPG and MDP-DIV in terms of NDCG@10 for each of the query group. The results are reported in Figure 6. We found that the average NDCG@10 of PPG over the groups is higher than MDP-DIV (except the fifth group). Furthermore, it is interesting to see that PPG performs particularly better than MDP-DIV for queries with small numbers of relevant documents (e.g., 0% ~ 20%, 20% ~ 40%, 40% ~ 60%, and 60% ~ 80%). The results indicate that PPG successfully avoids creating a model biased toward easy queries.

6 CONCLUSION

In this paper, we proposed a novel reinforcement learning algorithm tailored for document ranking in IR, called Pairwise Policy Gradient (PPG). In its learning, PPG estimates the gradients based on the intra-query comparisons between pairs of sampled document lists. PPG addressed the two issues in conventional policy gradient: ignoring intra-query relative ordering nature of IR ranking and generating high variance gradient estimations. Theoretical analysis showed that PPG makes unbiased and low variance estimations, leading

to a correct and fast learning. Experimental results showed that PPG outperformed the state-of-the-art baselines on search result diversification and text retrieval. Empirical analysis also showed that PPG converged fast to an optimum and really reduced the estimation variances.

ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China (61872338, 61832017, and 61773362), Beijing Academy of Artificial Intelligence (BAAI2019ZD0305 and BAAI2020ZJ0303), Beijing Outstanding Young Scientist Program (BJJWZYJH012019100020098), the Youth Innovation Promotion Association CAS (2016102), Fundamental Research Funds for the Central Universities, and Research Funds of Renmin University of China (2018030246).

A PROOF OF THE THEOREMS

A.1 Proof of Theorem 4.1

Proof. According to policy gradient theorem([31], Chapter 13.2),

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \cdot \nabla \pi(a|s;\mathbf{w}),$$

where a is the action. Calculating it by sampling two times

$$\nabla J(\theta) \propto \sum_s \mu(s) \left(\sum_a q_\pi(s,a) \cdot \nabla \pi(a|s;\theta) + \sum_b q_\pi(s,b) \cdot \nabla \pi(b|s;\theta) \right).$$

Note that $\sum_a v(s) \nabla \pi(a|s;\theta) = v(s) \nabla \sum_a \pi(a|s;\theta) = v(s) \nabla 1 = 0$, where $v(s) = \sum_a q_\pi(s,a)\pi(a|s;\theta) = \sum_b q_\pi(s,b)\pi(b|s;\theta)$ is value function. The term can be added to ∇J without changing its value:

$$\begin{aligned}\nabla J(\theta) \propto \sum_s \mu(s) \Bigg(& \sum_a q_\pi(s,a) \cdot \nabla \pi(a|s;\theta) \\ & - \sum_b q_\pi(s,b)\pi(b|s;\theta) \cdot \sum_a \nabla \pi(a|s;\theta) \\ & - \sum_a q_\pi(s,a)\pi(a|s;\theta) \cdot \sum_b \nabla \pi(b|s;\theta) + \sum_b q_\pi(s,b) \cdot \nabla \pi(b|s;\theta) \Bigg).\end{aligned}$$

Since the policy is a distribution over the actions, we have $\sum_b \pi(b|s;\theta) = \sum_a \pi(a|s;\theta) = 1$, and the gradient can be written as:

$$\begin{aligned}\nabla J(\theta) &\propto \sum_s \mu(s) \Bigg(\sum_a \sum_b q_\pi(s,a)\pi(b|s;\theta)\nabla \pi(a|s;\theta) \\ &\quad - \sum_a \sum_b q_\pi(s,a)\pi(a|s;\theta)\nabla \pi(b|s;\theta) \\ &\quad - \sum_a \sum_b q_\pi(s,b)\pi(b|s;\theta)\nabla \pi(a|s;\theta) \\ &\quad + \sum_a \sum_b q_\pi(s,b)\pi(a|s;\theta)\nabla \pi(b|s;\theta) \Bigg) \\ &= \sum_s \mu(s) \sum_a \sum_b \big(q_\pi(s,a) - q_\pi(s,b)\big) \\ &\qquad \cdot \big(\pi(b|s;\theta)\nabla \pi(a|s;\theta) - \pi(a|s;\theta)\nabla \pi(b|s;\theta)\big)\end{aligned}$$

□

A.2 Proof of Theorem 4.2

Proof. The difference of these two variance is

$$\begin{aligned}\mathrm{Var}(\mathbf{g}_1) - \mathrm{Var}(\mathbf{g}_2) =& \mathrm{tr}(\mathrm{cov}(\mathbf{g}_1,\mathbf{g}_1)) - \mathrm{tr}(\mathrm{cov}(\mathbf{g}_2)) \\ =& \mathbb{E}_{\mu_\pi,\pi}\left[\|\mathbf{g}_1 - \mathbb{E}_{\mu_\pi,\pi}[\mathbf{g}_1]\|^2\right] \\ & - \mathbb{E}_{\mu_\pi,\pi}\left[\|\mathbf{g}_2 - \mathbb{E}_{\mu_\pi,\pi}[\mathbf{g}_2]\|^2\right] \\ =& \mathbb{E}_{\mu_\pi,\pi}\left[\|\mathbf{g}_1\|^2\right] - \mathbb{E}_{\mu_\pi,\pi}\left[\|\mathbf{g}_2\|^2\right] \\ =& \mathbb{E}_{\mu_\pi,\pi}\left[\|\mathbf{g}_1\|^2 - \|\mathbf{g}_2\|^2\right]\end{aligned}$$

The third equation stands because $\mathbb{E}_{\mu_\pi,\pi}[\mathbf{g}_1] = \mathbb{E}_{\mu_\pi,\pi}[\mathbf{g}_2]$. Note that according to Theorem 4.1 and the policy gradient theorem (Chapter 13.2 of [31]), both $\mathbf{g}_1$ and $\mathbf{g}_2$ are unbiased estimation. According to the definitions of $\mathbf{g}_1$ and $\mathbf{g}_2$, and denoting $q_a = q_\pi(s,a)$, $q_b = q_\pi(s,b)$, $\pi_a = \pi(a|s;\theta)$, and $\pi_b = \pi(b|s;\theta)$ for simplicity:

$$\begin{aligned}\|\mathbf{g}_1\|^2 - \|\mathbf{g}_2\|^2 =& \|(q_a - q_b) \cdot (\nabla \log \pi_a - \nabla \log \pi_b)\|^2 \\ & - \|q_a \cdot \nabla \log \pi_a + q_b \cdot \nabla \log \pi_b\|^2 \\ =& q_a^2\|\nabla \log \pi_b\|^2 - 2q_a^2[\nabla \log \pi_a]^T \nabla \log \pi_b + q_b^2\|\nabla \log \pi_a\|^2 \\ & - 2q_b^2[\nabla \log \pi_a]^T \nabla \log \pi_b - 2q_a q_b\|\nabla \log \pi_a\|^2 \\ & - 2q_a q_b\|\nabla \log \pi_b\|^2 + 2q_a q_b[\nabla \log \pi_a]^T \nabla \log \pi_b.\end{aligned}$$

For any $a \sim \pi(\cdot|s;\theta)$,

$$\begin{aligned}\mathbb{E}_{\mu_\pi,\pi}\left[\nabla \log \pi_a\right] =& \mathbb{E}_{\mu_\pi,\pi}\left[\frac{\nabla \pi_a}{\pi_a}\right] = \mathbb{E}_{\mu_\pi}\left[\sum_a \pi_a \frac{\nabla \pi_a}{\pi_a}\right] \\ =& \mathbb{E}_{\mu_\pi}\left[\sum_a \nabla \pi_a\right] = \mathbb{E}_{\mu_\pi}\left[\nabla \sum_a \pi_a\right] = \mathbb{E}_{\mu_\pi}\left[\nabla 1\right] = \mathbf{0}.\end{aligned}$$

and a and b are independent random variables, it is easy to know

$$\begin{aligned}\mathbb{E}_{\mu_\pi,\pi} q_a^2[\nabla \log \pi_a]^T \nabla \log \pi_b =& [\mathbb{E}_{\mu_\pi,\pi} q_a^2 \nabla \log \pi_a]^T \mathbb{E}_{\mu_\pi,\pi} \nabla \log \pi_b \\ =& [\mathbb{E}_{\mu_\pi,\pi} q_a^2 \nabla \log \pi_a]^T \mathbf{0} = 0,\end{aligned}$$

and similarly

$$\mathbb{E}_{\mu_\pi,\pi} q_b^2[\nabla \log \pi_a]^T \nabla \log \pi_b = 0.$$

Therefore,

$$\begin{aligned}&\mathrm{Var}(\mathbf{g}_1) - \mathrm{Var}(\mathbf{g}_2) = \mathbb{E}_{\mu_\pi,\pi}[\|\mathbf{g}_1\|^2 - \|\mathbf{g}_2\|^2] \\ &= \mathbb{E}_{\mu_\pi,\pi}\left[-2(q_a^2 + q_b^2)[\nabla \log \pi_a]^T \nabla \log \pi_b\right] \\ &\qquad + \mathbb{E}_{\mu_\pi,\pi}[q_b^2\|\nabla \log \pi_a\|^2 + q_a^2\|\nabla \log \pi_b\|^2 - 2q_a q_b\|\nabla \log \pi_a\|^2 \\ &\qquad - 2q_a q_b\|\nabla \log \pi_b\|^2 + 2q_a q_b[\nabla \log \pi_a]^T \nabla \log \pi_b] \\ &= \mathbb{E}_{\mu_\pi,\pi}[q_b^2\|\nabla \log \pi_a\|^2 + q_a^2\|\nabla \log \pi_b\|^2 - 2q_a q_b\|\nabla \log \pi_a\|^2 \\ &\qquad - 2q_a q_b\|\nabla \log \pi_b\|^2 + 2q_a q_b[\nabla \log \pi_a]^T \nabla \log \pi_b] \\ &= \mathbb{E}_{\mu_\pi,\pi}(q_a - q_b)(q_a\|\nabla \log \pi_b\|^2 - q_b\|\nabla \log \pi_a\|^2) \\ &\qquad - \mathbb{E}_{\mu_\pi,\pi} q_a q_b\|\nabla \log \pi_a - \nabla \log \pi_b\|^2 \\ &\le \mathbb{E}_{\mu_\pi,\pi}(q_a - q_b)(q_a\|\nabla \log \pi_b\|^2 - q_b\|\nabla \log \pi_a\|^2),\end{aligned}$$

where the last inequation stands because $q_a \ge 0, q_b \ge 0$ and $\|\nabla \log \pi_a - \nabla \log \pi_b\|^2 \ge 0$. Thus, we have

$$\mathrm{Var}(\mathbf{g}_1) \le \mathrm{Var}(\mathbf{g}_2),$$

if $\mathbb{E}_{\mu_\pi,\pi}\left[(q_a - q_b)(q_b\|\nabla \log \pi_a\|^2 - q_a\|\nabla \log \pi_b\|^2)\right] \ge 0$.

□

REFERENCES

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*. 89–96.

[2] Chris J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report.

[3] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. 2006. Adapting Ranking SVM to Document Retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*. 186–193.

[4] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*. 129–136.

[5] Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*. 335–336.

[6] David Carmel and Elad Yom-Tov. 2010. Estimating the Query Difficulty for Information Retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 2, 1 (2010), 1–89.

[7] 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 the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. ACM, New York, NY, USA, 659–666.

[8] Koby Crammer and Yoram Singer. 2002. Pranking with Ranking. In *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.). MIT Press, 641–647.

[9] Van Dang and W. Bruce Croft. 2012. Diversity by Proportionality: An Election-based Approach to Search Result Diversification. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*. ACM, New York, NY, USA, 65–74.

[10] Jun Feng, Heng Li, Minlie Huang, Shichen Liu, Wenwu Ou, Zhirong Wang, and Xiaoyan Zhu. 2018. Learning to Collaborate: Multi-Scenario Ranking via Multi-Agent Reinforcement Learning. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. 1939–1948.

[11] Yue Feng, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2018. From Greedy Selection to Exploratory Decision-Making: Diverse Ranking with Policy-Value Networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. 125–134.

[12] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2013. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval* 16, 1 (01 Feb 2013), 63–90.

[13] Katja Hofmann, Shimon Whiteson, and Maarten Rijke. 2013. Balancing Exploration and Exploitation in Listwise and Pairwise Online Learning to Rank for Information Retrieval. *Inf. Retr.* 16, 1 (Feb. 2013), 63–90.

[14] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application. In *Proceedings of the 24th SIGKDD (KDD '18)*. 368–377.

[15] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446.

[16] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*. 133–142.

[17] Branislav Kveton, Csaba Szepesvári, Zheng Wen, and Azin Ashkan. 2015. Cascading Bandits: Learning to Rank in the Cascade Model. *CoRR* abs/1502.02763 (2015).

[18] Hang Li. 2014. Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition. *Synthesis Lectures on Human Language Technologies* 7, 3 (2014), 1–121.

[19] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. 2016. Collaborative Filtering Bandits. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. 539–548.

[20] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.* 3, 3 (March 2009), 225–331.

[21] Zhongqi Lu and Qiang Yang. 2016. Partially Observable Markov Decision Process for Recommender Systems. *CoRR* abs/1608.07793 (2016).

[22] Jiyun Luo, Sicong Zhang, and Hui Yang. 2014. Win-win Search: Dual-agent Stochastic Game in Session Search. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 587–596.

[23] Ramesh Nallapati. 2004. Discriminative Models for Information Retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04)*. 64–71.

[24] Harrie Oosterhuis and Maarten de Rijke. 2018. Ranking for Relevance and Display Preferences in Complex Presentation Layouts. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. 845–854.

[25] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. *Inf. Retr.* 13, 4 (Aug. 2010), 346–374.

[26] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning Diverse Rankings with Multi-armed Bandits. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. 784–791.

[27] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning Diverse Rankings with Multi-armed Bandits. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, New York, NY, USA, 784–791.

[28] Rodrygo L.T. Santos, Craig Macdonald, and Iadh Ounis. 2010. Exploiting Query Reformulations for Web Search Result Diversification. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. 881–890.

[29] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *J. Mach. Learn. Res.* 6 (Dec. 2005), 1265–1295.

[30] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and Anxiang Zeng. 2018. Virtual-Taobao: Virtualizing Real-world Online Retail Environment for Reinforcement Learning. *CoRR* abs/1805.10000 (2018).

[31] Richard S. Sutton and Andrew G. Barto. 2016. *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

[32] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. 515–524.

[33] 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 the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. 113–122.

[34] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2016. Modeling Document Novelty with Neural Tensor Network for Search Result Diversification. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. 395–404.

[35] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2017. Adapting Markov Decision Process for Search Result Diversification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. 535–544.

[36] Jun Xu and Hang Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. 391–398.

[37] Hui Yang, Dongyi Guan, and Sicong Zhang. 2015. The Query Change Model: Modeling Session Search As a Markov Decision Process. *ACM Trans. Inf. Syst.* 33, 4, Article 20 (May 2015), 33 pages.

[38] Yisong Yue and Thorsten Joachims. 2008. Predicting Diverse Subsets Using Structural SVMs. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, New York, NY, USA, 1224–1231.

[39] Yisong Yue and Thorsten Joachims. 2009. Interactively Optimizing Information Retrieval Systems As a Dueling Bandits Problem. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. 1201–1208.

[40] Wei Zeng, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Reinforcement Learning to Rank with Markov Decision Process. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. 945–948.

[41] Wei Zeng, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2018. Multi Page Search with Reinforcement Learning to Rank. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '18)*. 175–178.

[42] Sicong Zhang, Jiyun Luo, and Hui Yang. 2014. A POMDP Model for Content-free Document Re-ranking. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 1139–1142.

[43] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. ACM, New York, NY, USA, 95–103. https://doi.org/10.1145/3240323.3240374

[44] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. 1040–1048.

[45] Yadong Zhu, Yanyan Lan, Jiafeng Guo, Xueqi Cheng, and Shuzi Niu. 2014. Learning for Search Result Diversification. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 293–302.

[46] Lixin Zou, Long Xia Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement Learning to Optimize Long-term User Engagement in Recommender Systems. In *Proceedings of the 25th annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '19)*.