Multi Page Search with Reinforcement Learning to Rank

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

ABSTRACT

Web search engines are typically designed to involve multiple pages of search results, and the search users engaging in exploratory search with ad hoc queries are likely to access more than one result pages. The ranking of web pages for such queries should consider additional information other than the original query, e.g., the user clicks on previous result pages. Existing methods that utilize this kind of information usually involve relevance feedback, which uses the feedback information to explore the user's intent. However, due to the limitation of the feedback mechanism, it is difficult to apply existing relevance feedback techniques to state-of-the-art learning to rank models. In this paper, we propose a novel learning to rank model for multi page search in which the user's feedback can be naturally utilized for improving the ranking of next result page. The model, referred to as MDP-MPS, formalizes the ranking of documents in multi page search as a Markov decision process (MDP) in which the search engine corresponds to the agent for constructing the document rankings in the result pages, and the user corresponds to the environment for judging the rankings and providing rewards. The policy gradient algorithm of REINFORCE is adopted for learning the model parameters. Experimental results on OHSUMED dataset showed that our approach outperformed the baselines of traditional relevance ranking model of ListNet and relevance feedback method of Rocchio.

KEYWORDS

Multi Page Search; Reinforcement Learning

ACM Reference Format:
Wei Zeng, Jun Xu*, Yanyan Lan, Jiafeng Guo, Xueqi Cheng. 2018. Multi Page Search with Reinforcement Learning to Rank. In *ICTIR '18: 2018 ACM SIGIR International Conference on the Theory of Information Retrieval, September 14–17, 2018, Tianjin, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3234944.3234977

1 INTRODUCTION

The modern information retrieval interface typically involves multiple pages of search results and users are likely to access more than one page. In many common retrieval scenarios, the search results are split into multiple pages that the user traverses across by clicking 'next page' button. The users examine a Search Engine Result Page(SERP) by browsing the rank list from top to down, click on relevant documents and return to the SERP in the same session. A good multi-page search system should begin by a static method and could continue to adapt the model based on the feedback from the user. Relevance feedback method [7] has been proved very effective for improving retrieval accuracy over the interactive information retrieval tasks. As for the Rocchio algorithm [11], the search engine gets the feedback information from the user, and then add weight of the terms from known relevant documents or minus weight of the terms from known irrelevant documents. Most relevant feedback methods balance the initial query and the feedback information based on a fixed value, rather than apply an adaptive coefficient for each query and each set of feedback.

Learning to rank methods have been widely used for information retrieval, in which all the documents are represented by feature vectors to reflect the relevance of the documents to the query [8]. The learning to rank methods aim to learn a score function for the candidate documents by minimizing a carefully designed loss function. Our work considers the multi page search scenario, and applies relevance feedback techniques to state-of-art learning to rank models. The multi-page search process is an interactive process between the user and the search engine. At each time step, the search engine selects M documents to construct a rank list. The user browses this rank list from top to down, clicks the relevant documents, skips the irrelevant documents, and then clicks the "next page" button for more results. In this paper, we mathematically formulate the multi page search process as an Markov Decision Process. The search engine is treated as the agent, which selects documents from remaining candidate document set to deliver to the user for satisfying the user's information need. The state of the environment is consist of the query, remaining documents, rank position and user's click information. We apply the soft-max policy to balance the exploration and exploitation during training and design the reward on the basis of IR measure metric. A classical policy gradient policy method based on the REINFORCE is applied to optimize the search policy.

In this paper, we propose a technical schema to use user feedback from the top-ranked documents to generate re-ranking for the remaining documents. Compared with existing methods, our method enjoys the following advantages: 1) Formulate the multi-page process as an Markov Decision Process and apply policy gradient to train the search policy which could optimize the search measure metric directly. 2) Apply the Recurrent Neural Network to process the feedback and improve the traditional learning to rank model with the feedback information based on Rocchio.

We use traditional learning to rank method ListNet, RankNet and RankBoost as the initial model to construct the experiments

* Corresponding author: Jun Xu

ICTIR '18, September 14–17, 2018, Tianjin, China

ACM ISBN 978-1-4503-5656-5/18/09…$15.00
https://doi.org/10.1145/3234944.3234977

on OHSUMED dataset, simulate the interaction between the search engine and user based on Dependent Click Model (DCM). The experimental result show that our model can prove the ranking accuracy for traditional learning to rank method and has better generalization ability.

2 RELATED WORK

In interactive information retrial, several methods have been proposed to utilize the relevance feedback information to modified the model. Relevance feedback has been extensively studied and most of the methods take it as a learning problem with s special treatment of query. For example, the well known Rocchio algorithm [11, 12] uses explicit feedback to improve query. The balance parameter is usually set to be a fixed value across all the queries and collections. However, due to the difference in queries and feedback documents, it should be optimized for each query and each set of feedback. In [10], presented a learning approach to adaptively predict the optimal balance coefficient for each query and each collection and given three heuristics to characterize the balance between query and feedback information.

A similar approach was explored by [6], where formulated this problem as a Bayesian sequential model and used a dynamic programming approach for optimizing. In [16], [17] and [15], has adapted Markov Decision Process to model the diverse ranking, session search and relevance ranking process. Online learning to rank methods aim to incrementally learn from user feedback in real time and formulated as reinforcement learning problem in[5]. Different these methods, we take the multi page search scenarios, use the Recurrent Neural Network to repress the feedback information and modified the initial static learning to rank model inspired by the Rocchio algorithm.

3 PROBLEM FORMULATION

3.1 Multi Page Search as MDP

The multi-page search process can be treated as an interactive ranking process between the search engine and the user, just as figure 1 shown. At each time step, the search engine selects M documents from the candidate documents set to construct a search result page and delivers it to the user. The user receives these documents, browses them from top to down, clicks the relevant documents and skips the irrelevant documents. Then, the user will clicks the 'next page' button, and the search engine constructs another search result page from the remaining candidate document by considering the user's click information. This process can be mathematically formulated as a Markov Decision Process(MDP), which can be represented as 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:

State S describes the environment. In multi pages search process, the state can be the ranking position to calculate the reward of the received document, the relevant ranked documents, the irrelevant ranked documents as well as the the remaining documents. A tuple is given to describe the state s_t:

$$[t, X_{\mathrm{r}}^t, X_{\mathrm{ir}}^t, X_{\mathrm{c}}^t] \tag{1}$$

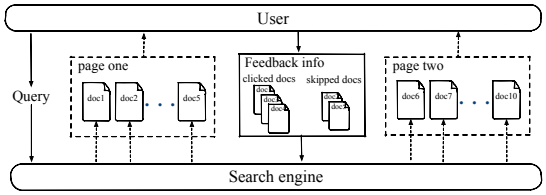

Figure 1: The Multi Page Search process.

where t is the time step, X_{r}^t and X_{ir}^t are the relevant ranked documents and irrelevant ranked documents lists respectively, X_{c}^t is the remaining documents set.

Action A is taken by the agent to influence the state of the environment. At each time step, the search engine selects M documents to construct the search result page. However, there are still an insurmountable number of possible rank lists, even if restrict the search agent to only select M documents. To lower the computational complexity of generating the search result page, we consider the search engine select one document at a time. So, at the time step t, $a_t \in A(s_t)$ represents to select a document $\mathbf{x}_{m(a_t)} \in X_{\mathrm{c}}^t$ 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 the dynamic of the environment, which can be described as a function $\mathcal{T} : S \times A \rightarrow S$ which maps a state s_t into a new state s_{t+1} of the environment in response to the selected action a_t. For multi page search, select a document a_t means removing the document $\mathbf{x}_{m(a_t)}$ out of the candidate set. And once the user clicks the 'next page' button, the search engine gains the judgements over the documents of current page. So the transition function can be described by the following equation:

$$\begin{aligned} s_{t+1} &= \mathcal{T}([t, X_{\mathrm{r}}^t, X_{\mathrm{ir}}^t, X_{\mathrm{c}}^t], a_t) \\ &= \begin{cases} [t+1, X_{\mathrm{r}}^t + \{\mathbf{x}_{\mathrm{r}}\}, X_{\mathrm{ir}}^t + \{\mathbf{x}_{\mathrm{ir}}\}, X_{\mathrm{c}}^t \setminus \{\mathbf{x}_{m(a_t)}\}] & \text{interaction} \\ [t+1, X_{\mathrm{r}}^t, X_{\mathrm{ir}}^t, X_{\mathrm{c}}^t \setminus \{\mathbf{x}_{m(a_t)}\}] & \text{otherwise} \end{cases}, \end{aligned} \tag{2}$$

where interaction means the user clicks the 'next page' button. And $\{\mathbf{x}_{\mathrm{r}}\}$ and $\{\mathbf{x}_{\mathrm{ir}}\}$ is the rank list consisted by the relevant documents and irrelevant documents of current page respectively.

Reward $\mathcal{R}(S, A)$ measures the influence to the environment by the selected action. For multi page search, the reward should be able to evaluate the quality of the chosen document. We design the reward function on the basis of the IR evaluation measures, the promotion of the DCG [3]:

$$\mathcal{R}_{\mathrm{DCG}}(s_t, a_t) = \begin{cases} 2^{y_{m(a_t)}} - 1 & t = 0 \\ \frac{2^{y_{m(a_t)}} - 1}{\log_2(t+1)} & t > 0 \end{cases}, \tag{3}$$

where $y_{m(a_t)}$ is the relevance label of the selected document.

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. We apply a soft-max policy which based on the Gibbs distribution, which could be described as:

$$\pi(a_t|s_t) = \frac{\exp\{f(a_t, s_t)\}}{\sum_{a \in A(s_t)} \exp\{f(a_t, s_t)\}}, \tag{4}$$

where $f(a_t, s_t)$ can be seen as a score function to each documents. The details of this score function $f(a_t, s_t)$ will be described in the next section.

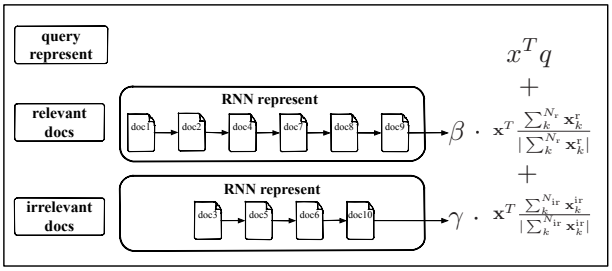

Figure 2: The framework for Multi-Page Search.

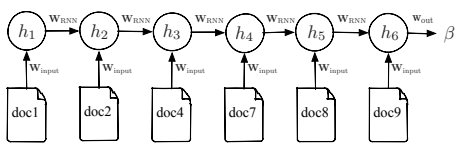

Figure 3: Calculate the coefficient β.

3.2 Involve the feedback

Relevant feedback technology has been proved can effectively improve the rank accuracy. For example, the well known Rocchio algorithm [11, 12] uses explicit feedback to improve the user's query. The formula and variable definitions defined as:

$$Q_{i+1} = \alpha Q_i + \beta \sum_{\text{rel}} \frac{D_i}{|D_i|} - \gamma \sum_{\text{irrel}} \frac{D_i}{|D_i|} \tag{5}$$

where, Q_i is the i-th iteration queries, and D_i represents the document which has been delivered to the user.

Inspired by Rocchio algorithm, we try to improve the traditional learning to rank method based on the relevance feedback technology. In traditional learning to rank settings [9] , the data is represented as $\{q^n, X^n, Y^n\}_{n=1}^N$, where $X^{(n)}=\left\{\mathbf{x}_1^{(n)}, \cdots, \mathbf{x}_{M_n}^{(n)}\right\}$ is the query-document feature set for query q^n, $Y^{(n)}=\left\{y_1^{(n)}, \cdots, y_{M_n}^{(n)}\right\}$ is the relevance label set, and M_n is the number of the candidate documents. The learning to rank methods aim to learn a score function $f(\mathbf{x})$, which is used to compute the relevant score for each candidate document. The search result rank list is generated by sorting the candidate documents by these scores. We try to adopt Rocchio style algorithm to update the score of the remaining documents based on the user click feedback information from previous pages.

$$f_{\text{new}}(\mathbf{x}) = f(\mathbf{x}) + \beta \cdot \mathbf{x}^T \frac{\sum \mathbf{x}_{\text{rel}}}{|\sum \mathbf{x}_{\text{rel}}|} + \gamma \cdot \mathbf{x}^T \frac{\sum \mathbf{x}_{\text{irrel}}}{|\sum \mathbf{x}_{\text{irrel}}|} \tag{6}$$

where $\mathbf{x}_{\text{rel}}$ and $\mathbf{x}_{\text{irrel}}$ represent the query-document feature of the relevant ranked document and irrelevant document respectively.

As equation 6 shown, the score of a remaining document is formed by three part. The first part computes the score based on the traditional learning to rank model, which learned by traditional learning to rank method. The second part, measures the influence from the relevant ranked documents, and the third part, measures the influence from the irrelevant ranked documents. Note that the normalization procedure ensures the equal emphasis of the impact from relevant ranked documents and irrelevant ranked documents.

The parameters, β and γ, are set to describe the influence from the relevant ranked documents and irrelevant documents. In this paper, we apply an recurrent neural network (RNN) to represent the ranked documents and calculate the parameters based on these

Algorithm 1 MDP for Multi Page Rank

Input: Labeled training set $D = \{(q^{(n)}, X^{(n)}, Y^{(n)})\}_{n=1}^N$, learning rate η, discount factor $\hat{\gamma}$, reward function R,
Output: The model parameters $\Theta = \left\{\mathbf{W}_{\text{input}}^{\text{rel}}, \mathbf{W}_{\text{RNN}}^{\text{rel}}, \mathbf{w}_{\text{out}}^{\text{rel}}, \mathbf{W}_{\text{input}}^{\text{irrel}}, \mathbf{W}_{\text{RNN}}^{\text{irrel}}, \mathbf{w}_{\text{out}}^{\text{irrel}}\right\}$

```
Initialize the parameters Θ ← random values
for epoch = 1, 2, 3, ··· do
    ΔΘ = 0
    for each query q do
        Select top M documents from the candidate document set X_C to construct a rank list {x}.
        The user bowsers the rank list, click the relevant documents {x_rel} and skips the irrelevant documents {x_irrel}.
        Initialize s_0 ← [0, {x_rel}, {x_irrel}, X_C \ {x}], and episode E ← ∅
        for i = 0 to T do
            for t = i * M + 1 to (i + 1) * M do
                Sample an action a_t ∈ A(s_t) ~ π(a_t|s_t; w)
                Calculate the reward r_{t+1} ← R(s_t, a_t)
                Append (s_t, a_t, r_{t+1}) at the end of the episode
                if mod(t, M) == 0 then
                    Transit state by the relevant documents {x_rel} and the irrelevant documents {x_irrel} on current page.
                    s_{t+1} ← [t+1, X_r^M+{x_ir}, X_ir^M+{x_irrel}, X_c^t \{x_{m(a_t)}}]
                else
                    State transition s_{t+1} ← [t+1, X_r^M, X_ir^M, X_c^t \{x_{m(a_t)}}]
                end if
            end for
        end for
        for t = 0 to (t + 1) * M do
            G_t ← Σ_{k=t}^{(t+1)*M} γ̂^{k-t} r_t
            ΔΘ ← ΔΘ + η γ̂^{t-1} G_t ∇_Θ log π(a_t|s_t; Θ)
        end for
    end for
    Update the model parameter Θ ← Θ + ηΔΘ
end for
return w
```

represents. Figure 3 shows the details about how to calculate the parameter β based on the relevant ranked documents.

$$\begin{aligned} \mathbf{h}_{t+1} &= \sigma\left(\mathbf{W}_{\text{imput}}^{\text{rel}} \mathbf{x}_{m(a_t)} + \mathbf{W}_{\text{RNN}}^{\text{rel}} \mathbf{h}_t\right) \\ \beta &= \mathbf{W}_{\text{out}}^{\text{rel}} \mathbf{h}_{N_{\text{rel}}} \end{aligned} \tag{7}$$

where N_{rel} is the total number of the relevant ranked documents. The parameter γ is calculated in a similar way.

3.3 Learning with Policy Gradient

In this paper, we calculate the parameter β and γ dynamically on the basis of the relevant ranked documents and irrelevant ranked documents. The parameters of our model can be represented as:

$$\Theta = \left\{\mathbf{W}_{\text{input}}^{\text{rel}}, \mathbf{W}_{\text{RNN}}^{\text{rel}}, \mathbf{w}_{\text{out}}^{\text{rel}}, \mathbf{W}_{\text{input}}^{\text{irrel}}, \mathbf{W}_{\text{RNN}}^{\text{irrel}}, \mathbf{w}_{\text{out}}^{\text{irrel}}\right\} \tag{8}$$

The policy gradient method [13, 14] is used to learn the parameters with the objective to maximize the expected discount cumulative reward, which defined as:

$$L(\Theta) = \mathbb{E}_{\mathcal{E}\sim\pi}[\sum_{k=1}^{M\cdot T} \hat{\gamma}^{k-1} r_k]. \tag{9}$$

where $\mathcal{E}$ is the ranking list sampled by the search engine based on current policy π, M is the number of the documents per search page, T is the number of interaction between the user the search engine, $\hat{\gamma}$ is the discount rate which set as 1 to guarantee the consistency over the objective and IR measure metric.

According to REINFORCE algorithm, the gradient can be

$$\nabla_\Theta L(\Theta) = \hat{\gamma}^t G_t \nabla_\Theta \log \pi_\Theta(a_t|s_t;\Theta), \tag{10}$$

Table 1: Ranking accuracies based on ListNet.

Method	NDCG@1	NDCG@10	NDCG@15	NDCG@20
ListNet	0.4943	0.4438	0.4383	0.4401
Rocchio	0.4943	0.4438	0.4419	0.4405
MDP-MPS	0.4943	0.4438	**0.4436**	**0.4453**

Table 2: Ranking accuracies based on RankBoost.

Method	NDCG@1	NDCG@10	NDCG@15	NDCG@20
RankBoost	0.5199	0.4521	0.4440	0.4449
Rocchio	0.5199	0.4521	0.4438	0.4359
MDP-MPS	0.5199	0.4521	**0.4459**	**0.4452**

Table 3: Ranking accuracies based on RankNet.

Method	NDCG@1	NDCG@10	NDCG@15	NDCG@20
RankNet	0.5039	0.4347	0.4305	0.4335
Rocchio	0.5039	0.4347	0.4345	0.4335
MDP-MPS	0.5039	0.4347	**0.4365**	**0.4358**

which is further be estimated with Monte-Carlo sampling. Algorithm 1 shows the procedure of optimize the parameters. At each iteration, an episode 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 $\nabla_\Theta \log \pi(a_t|s_t;\Theta)$, scaled by the step size η, the discount rate $\hat{\gamma}^t$, and the long-term return of the sampled episode starting from position t, denoted as G_t : $G_t = \sum_{k=t}^{M\cdot T} \hat{\gamma}^{k-t} r_t$.

4 EXPERIMENTS

We simulate the user's interaction by an evaluation setup proposed in [5]. This setup combine datasets with explicit relevance judgements that are typically used for supervised learning to rank with recently developed click models. This click model is based on the Dependent Click Model (DCM), a generalization of the cascade model. The model posits that users travers result lists from top to button, examining each document as it is encountered. We use the perfect model, where all relevant documents are clicked and no non-relevant documents are clicked.

The experiments are constructed over the learning to rank dataset "OHSUMED". We take the classical learning to rank methods ListNet [2] RankNet [1] and RankBoost [4] as the initial methods and set $M = 10$ as it provides reasonable and meaningful results within the scope of our test data set and is reflective of actual search system. The grid search strategy is applied to find the best parameters $\beta \in [-5, 5]$ and $\gamma \in [-5, 5]$ based on the validation data set.

The table 1 and 2 give us the experiment results of our method as take the ListNet and RankBoost as the initial learning to rank model representatively. And the boldface indicates the highest score among all runs. From the results, we can see that our methods can outperform the Rocchio algorithm in both ListNet and RankBoost. It should note that, Rocchio algorithm would decrease the performance of the initial RankBoost method, as for different queries satisfy different balance coefficient. However, our method MDP-MPS gain a commensurate performance, which mean our method has better generalization ability.

5 CONCLUSION

The paper formulates the ranking process in multi page search as an Markov Decision Process and optimizes the model parameters with policy gradient. Specifically, on the basis of a traditional learning to rank model, the traditional relevance feedback model Rocchio is used to adapt it for the task of multi page search in which a Recurrent Neural Network is used to represent the user feedback information. Experimental results based on the OHSUMED dataset showed that the proposed MDP-MPS model, initialized with the document ranking generated by ListNet and RankBoost, can outperformed the corresponding baselines, respectively. The experimental results showed the effectiveness of MDP-MPS for multi page search. The results also showed that MDP-MPS has better generalization ability in the multi page search task.

6 ACKNOWLEDGMENTS

This work was funded by the National Key R&D Program of China under Grants No. 2016QY02D0405, the 973 Program of China under Grant No. 2014CB340401, the National Natural Science Foundation of China (NSFC) under Grants No. 61773362, 61425016, 61472401, 61722211, and 20180290, and the Youth Innovation Promotion Association CAS under Grants No. 20144310, and 2016102.

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*. 89–96.
[2] 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*. 129–136.
[3] Charles LA 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*. ACM, 659–666.
[4] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of machine learning research* 4, Nov (2003), 933–969.
[5] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2011. Balancing exploration and exploitation in learning to rank online. In *European Conference on Information Retrieval*. 251–263.
[6] Xiaoran Jin, Marc Sloan, and Jun Wang. 2013. Interactive exploratory search for multi page search results. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 655–666.
[7] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. 2007. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)* 25, 2 (2007), 7.
[8] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.
[9] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. 2007. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop*, Vol. 310.
[10] Yuanhua Lv and ChengXiang Zhai. 2009. Adaptive relevance feedback in information retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*. 255–264.
[11] Joseph John Rocchio. 1971. Relevance feedback in information retrieval. *THE SMART RETRIEVAL SYSTEM: Experiments in Automatic Document Processing* (1971), 313–323.
[12] Gerard Salton and Chris Buckley. 1990. Improving retrieval performance by relevance feedback. Vol. 41. Wiley Online Library, 288–297.
[13] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.
[14] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.
[15] Zeng Wei, 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*. 945–948.
[16] 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*. 535–544.
[17] Hui Yang, Dongyi Guan, and Sicong Zhang. 2015. The query change model: Modeling session search as a markov decision process. *ACM Transactions on Information Systems (TOIS)* 33, 4 (2015), 20.