MarlRank: Multi-agent Reinforced Learning to Rank

Shihao Zou*
szou2@ualberta.ca
University of Alberta

Zhonghua Li
lizhonghua3@huawei.com
Huawei

Mohammad Akbari
m.akbari@ucl.ac.uk
University College London

Jun Wang
jun.wang@cs.ucl.ac.uk
University College London

Peng Zhang
pzhang@tju.edu.cn
Tianjin University

ABSTRACT

When estimating the relevancy between a query and a document, ranking models largely neglect the mutual information among documents. A common wisdom is that if two documents are similar in terms of the same query, they are more likely to have similar relevance score. To mitigate this problem, in this paper, we propose a multi-agent reinforced ranking model, named MarlRank. In particular, by considering each document as an agent, we formulate the ranking process as a multi-agent Markov Decision Process (MDP), where the mutual interactions among documents are incorporated in the ranking process. To compute the ranking list, each document predicts its relevance to a query considering not only its own query-document features but also its similar documents' features and actions. By defining reward as a function of NDCG, we can optimize our model directly on the ranking performance measure. Our experimental results on two LETOR benchmark datasets show that our model has significant performance gains over the state-of-art baselines. We also find that the NDCG shows an overall increasing trend along with the step of interactions, which demonstrates that the mutual information among documents helps improve the ranking performance.

CCS CONCEPTS

• **Information systems → Learning to rank**; **Novelty in information retrieval**.

KEYWORDS

multi-agent reinforcement learning, learning to rank

ACM Reference Format:
Shihao Zou, Zhonghua Li, Mohammad Akbari, Jun Wang, and Peng Zhang. 2019. MarlRank: Multi-agent Reinforced Learning to Rank. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3358075

*This work was done when Shihao was with UCL and interned at Huawei.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358075

1 INTRODUCTION

Learning to rank, which learns a model for ranking documents (items) based on their relevance scores to a given query, is a key task in information retrieval (IR) [18]. Typically, a query-document pair is represented as a feature vector and many classical models, including SVM [8], AdaBoost [18] and regression tree [2] are applied to learn to predict the relevancy. Recently, deep learning models has also been applied in learning to rank task. For example, CDSSM [13], DeepRank [12] and their variants aim at learning better semantic representation of the query-document pair for more accurate prediction of the relevancy. Basically, most of previous approaches emphasize on the learning models with only query-document features considered. They do not investigate the effectiveness of global mutual information among documents to improve the ranking performance, which will be the focus of this paper.

Reinforcement learning has been applied in a few IR tasks, such as session search [11, 20], where a user's behavior in one session affects documents ranking in the next session. This can be formulated as an MDP and optimized using reinforcement learning. Recent works on reinforcement learning to rank model the ranking process as an MDP [6, 17], where, at each time step, an agent selects one document given its current observation (ranking position and unranked documents), and the reward is the improvement of NDCG. But essentially, this contextual information is limited as documents are passively selected and not allowed to interact with each other, and the interactions among documents can be considered as the mutual information among documents which helps to improve ranking performance.

Depart from prior approaches, we consider both query-document and document-document relevancy for ranking. Our hypothesis comes from a common wisdom that if two documents are very similar in terms of the same query, they are likely to have similar relevance scores. We give a motivational example to illustrate our intuition in Sec 2.2. Inspired by [10], we construct the ranking process as an interactive multi-agent environment where each document is treated as an agent. Specifically, our multi-agent environment comprises a query and a set of documents (agents). The ranking process is an interactive MDP among these agents. At each time step, each agent simultaneously presents its relevance score based on the observation of its own query-related information and the local information of its neighbor agents, and then a global ranking performance reward is given. After several time steps of interaction, all the agents achieve an equilibrium state. Their common objective is to find an optimal policy that jointly maximizes the expected accumulative reward.

The contributions of this paper are as follows: (1) we propose to leverage both query-document and document-document relevancy for ranking in IR; (2) we formulate the learning to rank problem as a multi-agent reinforced learning framework. To the best of our knowledge, the proposed framework is the first work that models document ranking as an interactive multi-agent reinforced learning process and optimizes the model directly on the ranking performance measure of the retrieval task. The experimental results show that our model outperforms most strong baselines and it is interesting to find NDCG increases along with time steps, which attests the interactions among documents help improve the ranking performance.

2 BACKGROUND

2.1 Related Work

Typically, ranking models can be classified into three categories: point-wise, pair-wise [1, 3, 8] and list-wise [4, 18, 19]. Point-wise ranking models $p(d|q;\theta)$ assume that documents are independent of each other for predicting the relevance, where θ means the parameter of the ranking model. On the contrary, pair-wise models regard a pair of documents as an instance and the pair of documents' relevance orders as categories to perform classification-based ranking, referred as $p(d_i|q,d_j;\theta)$ where d_i and d_j are a pair of documents. Empirical results show that pair-wise ranking models outperform point-wise ones as, intuitively, pair-wise ranking process takes mutual information between a pair of documents in the ranking process. As for list-wise models, they formulate a ranking list (either a document pair [19] or a documents list [4]) as a permutation distribution so that loss functions become continuous and differentiable. Then list-wise models can be directly optimized on ranking performance measures. However, in essence, list-wise models mainly focus on the reformulation of loss functions and, like point-wise and pair-wise models, treat each document or each pair of documents dependently to do ranking.

Traditional learning to rank models mainly focus on the minimization of classification errors, including RankSVM [8], RankBoost [7] and RankNet [1]. These models formulate ranking as a classification problem, either point-wise or pair-wise. Instead of optimizing the classification loss many previous works design a variety of differentiable loss functions which are directly based on the IR performance measure. For example, LambdaRank [3] speeds up RankNet by directly constructing the first derivative (lambda) of the implicit non-continuous cost (such as NDCG, MAP, etc.) with respect to the model parameters. Following this line, LambdaMART [2] uses multiple additive regression tree, which shows better ranking performance. Another line of work is based on list-wise loss. ListNet [4] maps a list of document scores to a differentiable loss defined on the permutation probability distribution. AdaRank [18] revises the non-continuous list ranking accuracy loss to a continuous exponential upper bound. Besides, SVMNDCG [5] and PermuRank[19] directly optimize the upper bound defined on the pairs between the perfect and imperfect permutations. Our work departs from them in that our model uses ranking performance measure as the reward and we can optimize the model directly based on this ranking performance measure via policy gradient.

Reinforcement learning has shown great success in many tasks, including game [14] and recommender system [21]. In IR, there are mainly two lines of works which are formulated as an MDP. One line is session search. In [11, 20], a collaborative search process between the user and the search engine is defined as an MDP. The other line mainly focuses on reinforcement learning to rank. The ranking process is formulated as an MDP. [17] At each time step, the agent selects one document in the ranking list until all the documents are selected and ranked. A further exploratory algorithm based on Monte Carlo tree search is proposed by [6]. In addition, IRGAN [16] uses policy gradient to optimize the generator. Our model is different from that of [6, 17] whose agent is only for the selection of documents. By regarding each document as an agent in a collaborative environment, we include mutual information among documents in our model to do ranking.

2.2 A Motivational Example

A motivational example is presented to illustrate how our intuition helps improve the ranking performance. In Fig. 1, given a query, we assume there are six documents with three non-relevant $\{d_1, d_2, d_3\}$ and three relevant ones $\{d_4, d_5, d_6\}$. The initial predictions at step 0 are given in Tab. 1, and we assume the naive policy is the average of the target document and its top-2 similar documents' previous actions (relevance scores). For example, the action of document d_4 at step 1 is the average of previous actions by document 4, 5 and 6, $a_1^4 = (a_0^4 + a_0^5 + a_0^6)/3 = 0.63$. We can see from Tab. 1 that document d_4 finds its neighbor documents d_5 and d_6, so it believes that it should have similar scores with these two neighbors and updates its prediction from 0.1 to 0.63. Finally, NDCG@3 increases after one step of interaction and remains the best ranking performance afterwards.

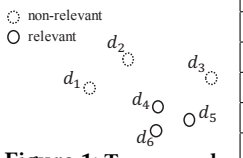

Figure 1: Toy example.

Table 1: Relevance score (action) given by each document.

step t	0	1	2	3
d_1 ($r=0$)	0	0.37	0.46	0.52
d_2 ($r=0$)	1.0	0.37	0.46	0.52
d_3 ($r=0$)	0	0.33	0.53	0.6
d_4 ($r=1$)	0.1	0.63	0.63	0.63
d_5 ($r=1$)	0.9	0.63	0.63	0.63
d_6 ($r=1$)	0.9	0.63	0.63	0.63
NDCG@3	0.3	1	1	1

3 FRAMEWORK

3.1 Reinforced Learning to Rank

Given a query q, we have N documents to rank, $\left(q, \{(d_i, y_i)\}_{i=1}^N\right)$, in which document d_i is represented as a query-document feature vector as in [9] and y_i is its relevance label. Now we formally define the Reinforced Learning to Rank problem as a multi-agent MDP of $\mathcal{M} =< \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \pi, \gamma >$:

$\mathcal{S} = \left\{q, \{(d_i, y_i, a^i)\}_{i=1}^N\right\}$ is a set of environment states, where each document is regarded as an agent, and a^i is the action of the i-th agent. $\mathcal{O} = \{O^1, ..., O^N\}$ is a set of observations for each agent. As the environment is fully observable, the observation of each agent is equivalent to the environment state, $o_t^i = s_t$, where $o_t^i \in O^i$ and $s_t \in \mathcal{S}$.

$\mathcal{A} = \{\mathcal{A}^1, ..., \mathcal{A}^N\}$ is a set of actions for each agent. The action is defined as the discrete relevance level for the query q. For agent i, it uses the policy $\pi_\theta(a_t^i|o_t^i)$ to decide an action at time step t, where θ is the parameter of the policy π. To make it simple, we assume all the agents use the same policy to choose their actions given their own observations.

$\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \times ... \times \mathcal{A}^N \mapsto \mathcal{S}$ is the state transition function.

$\mathcal{R} : \mathcal{S} \times \mathcal{A}^1 \times ... \times \mathcal{A}^N \mapsto \mathbb{R}$ is the reward function, which is defined as

$$r_t(a_t^1, ..., a_t^N) = \begin{cases} 0 & t < T \\ NDCG_T - NDCG_{best} & t = T \end{cases}, \quad (1)$$

where T is the final time step and $NDCG_{best}$ means the best NDCG value, which is 1 if no less than one document is relevant to the query and 0 if no document is relevant. That is to say, only at the final step, a non-zero reward is given. The reason for our reward definition is given in Eq. (1).

The return at time step t is defined as the accumulative reward in the future, $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k(a_k^1, ..., a_k^N)$. The objective is find an optimal policy to maximize the expected return:

$$J(\theta) = \arg\max_\theta \mathbb{E}_\pi \Big[\sum_{k=t}^{T} \gamma^{k-t} r_k(a_k^1, ..., a_k^N) \Big], \quad (2)$$

where γ is the discounted factor and T is the total time steps. Note that according to Eq. (1), the objective of Eq. (2) becomes

$$\begin{aligned} J(\theta) &= \arg\max_\theta \mathbb{E}_\pi \Big[\gamma^{T-t} (NDCG_T - NDCG_{best}) \Big] \\ &\propto \arg\max_\theta \mathbb{E}_\pi \big[NDCG_T \big], \end{aligned} \quad (3)$$

which is consistent with the goal of learning to rank task.

3.2 MarlRank

The proposed model, MarlRank, includes similarity and policy two modules as is shown in Fig. 2. The similarity module is used to find the top-k neighbor (similar) documents. We use $\{(d_{i,n}, s_{i,n}, a_{t-1}^{i,n})\}_{n=1}^{k}$ to denote the set of document d_i's local information, where $s_{i,n}$ means the similarity between document d_i and d_n and $a_{t-1}^{i,n}$ means previous actions of neighbor document d_n. Then, we define the observation o_t^i for d_i at time step t as

$$o_t^i = concat\big[d_i,\ \{a_{t-1}^{i,n}\}_{n=1}^{k},\ \{s_{i,n}\}_{n=1}^{k},\ \frac{1}{k}\sum_{n=1}^{k} s_{i,n} d_{i,n}\big], \quad (4)$$

whose first term is d_i's own query-document feature vector and the other three terms are the local information of d_i, including previous actions, similarities and weighted sum of neighbor document's feature vector. Then given the observation o_t^i, the policy module decides an action a_t^i. To make it clear, we will use $\pi_\theta(a_t^i|o_t^i)$ to denote the two modules in the following part.

Besides, we also add a small individual reward $r_t^i(a_t^i, y_i)$ as a regularization term for each agent at time step t. The individual reward is positive when the agent's action equals to its label, and negative otherwise. Afterwards, we can update the model according to REINFORCE[15] as

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \big[\nabla_\theta \log \pi_\theta(a_t^i|o_t^i) \ (R_t + r_t^i(a_t^i, y_i)) \big]. \quad (5)$$

Our algorithm is summarized in Algorithm 1.

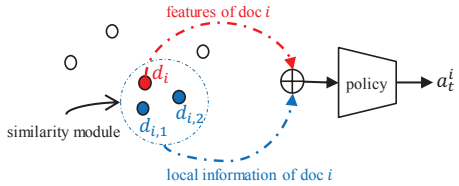

Figure 2: MarlRank: blue dashed circle is the similarity module, red and blue dashed arrow means doc i's own features and its local information respectively. $d_{i,1}$ and $d_{i,2}$ are the top-2 similar documents of d_i.

Algorithm 1 MarlRank

Input: policy π_θ; training dataset $D = \big\{(q, \{(d_i, y_i)\}_{i=1}^{N})\big\}$

```
Initialize π_θ with random weights θ
Pre-train π_θ using D
repeat
    for (q, {(d_i, y_i)}_{i=1}^N) in D do
        Sample a trajectory {(o_t^i, a_t^i)} using policy π_θ(a_t^i|o_t^i)
        Obtain a final reward by Eq. (1)
        Compute return R_t and individual reward r_t^i(a_t^i, y_i)
    end for
    Collect samples {( o_t^i, a_t^i, R_t + r_t^i(a_t^i, y_i) )}
    Update θ via REINFORCE according to Eq. (5)
until π_θ converges
```

4 EXPERIMENTS

4.1 Experiment setup

We conduct experiments on two LETOR benchmark datasets [9], MQ2007 and OHSUMED. Each dataset consists of queries, documents (query-document feature vectors) and relevance labels. Following LETOR, we conduct 5-fold cross-validation and calculate the average as the final test result. The relevance labels in both datasets consist of relevant (2), partially-relevant (1) and non-relevant (0).

Our focus in this paper is on the effectiveness of global mutual information among documents to improve the ranking performance, rather than on the similarity models or sophisticated neural ranking models. So our model has only one-layer MLP for similarity module and two-layer MLP for policy module with each layer 100 hidden units. The model outputs a probabilistic distribution over three relevance levels. We first pre-train our model to make it give better initial relevance prediction via supervised learning, and then train it via policy gradient. The individual reward used in OHSUMED is $0.001, 0.003, 0.004$ for the equal case of three relevance levels respectively and -0.001 for non-equal case, and $0.001, 0.003, 0.008$ and -0.001 in MQ2007. After one episode of sampling, the return is normalized to standard normal distribution. γ is set to 0.95 and learning rate is set to 4×10^{-7} in both datasets. The initial previous action a_{-1}^i is set to 0 for all agents. In test and validation, the maximum time step is 10, though we find that the top-10 ranking order no longer changes after about 4 steps for most queries.

Several classical methods are used as baselines in Tab. 2 and 3, whose results are obtained via RankLib[1]. To make a fair comparison, RankNet is the same three-layer MLP with each layer 100

[1] https://sourceforge.net/p/lemur/wiki/RankLib/

Table 2: Test results on MQ2007. (*, ♯, † and ‡ mean a significant improvement over AdaRank, RankNet, ListNet and LambdaMART using Wilcoxon signed-rank test $p < 0.05$.)

	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankNet[1]	0.3753	0.3802	0.3858	0.4173
RankBoost[7]	0.3968	0.3978	0.4046	0.4333
AdaRank[18]	0.3751	0.3901	0.4013	0.4300
ListNet[4]	0.3870	0.3888	0.3936	0.4207
LambdaMART[2]	0.4154	0.4138	**0.4197**	0.4478
MDPRank[17]	0.4033	0.4059	0.4113	0.4350
MarlRank	**0.4254**$^{*\sharp\dagger\ddagger}$	**0.4139**$^{\sharp}$	0.4179$^{\sharp\dagger}$	**0.4489**$^{*\sharp\dagger}$

Table 3: Test results on OHSUMED.

	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankNet[1]	0.4667	0.4338	0.4356	0.4309
RankBoost[7]	0.5043	0.4739	0.4563	0.4385
AdaRank[18]	0.5078	**0.4913**	0.4647	0.4478
ListNet[4]	0.4595	0.4325	0.4284	0.4181
LambdaMART[2]	0.4563	0.4523	0.4374	0.4299
MDPRank[17]	0.5363	0.4885	0.4695	**0.4591**
MarlRank	**0.5521**$^{*\sharp\dagger\ddagger}$	0.4771$^{\sharp\dagger}$	**0.4698**†	0.4551$^{\sharp\dagger\ddagger}$

hidden units as our model. Though LambdaMART has better implementations such as LightGBM, we should note that our model is a simple one and our focus is to prove that mutual information among documents is helpful to improve ranking.

4.2 Experiment results

Experimental results are shown in Tab. 2 and 3. We can find that MarlRank outperforms most of baselines in terms of NDCG@1, 3, 5 and 10. In MQ2007, MarlRank has significant performance gains on NDCG@1, 5 and 10 over RankNet, AdaRank and ListNet. Our model also exceeds MDPRank on all four NDCG measures. This may lie in the fact that we include more mutual documents information in the ranking process. In OHSUMED, our model shows significant improvement on three baselines in terms of NDCG@1 and 10, while MDPRank is better on NDCG@3 and 10. The reason could be the much fewer amount of training data in OHSUMED than in MQ2007, which makes the model in OUSUMED perform a little worse than that in MQ2007. Fig. 3 shows how the NDCG measures change over time when evaluating MarlRank on both datasets. It is worth to notice from Fig. 3 that all four NDCG measures (computed after each time step on test set) increase along with the increase of time step for both datasets, which demonstrates that the interactions (mutual information) among documents help improve the ranking performance. While on OHSUMED all measures shows an overall increasing trend with slight fluctuations in the middle especially for NDCG@1,3, and 5, this is probably because of the imbalanced or inadequate positive/negative documents for different queries.

5 CONCLUSION

In this paper, we propose MarlRank, a multi-agent framework for the task of learning to rank. By considering each document as an agent, we effectively integrate the mutual information among documents in generating the final ranking list. The experimental results demonstrate the performance improvement of MarlRank over multiple state-of-the-art baselines and also validate the effectiveness of applying document mutual information in such an interactive way in the ranking task. For the future work, more sophisticated approach should be studied to decide neighbor documents from not only similarity but also diversity, novelty, etc.

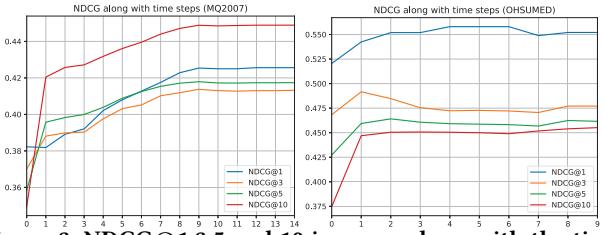

Figure 3: NDCG@1,3,5 and 10 increase along with the time step, demonstrating the interactions (mutual information) among documents help improve the ranking performance.

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 *ICML*.
[2] Chris J.C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview.
[3] Christopher J Burges, Robert Ragno, and Quoc V Le. 2007. Learning to rank with nonsmooth cost functions. In *NeurIPS*.
[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 *ICML*.
[5] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. 2008. Structured Learning for Non-smooth Ranking Losses. In *KDD*.
[6] 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 *SIGIR*.
[7] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *JMLR* (2003).
[8] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support vector learning for ordinal regression. (1999).
[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 *SIGIR*.
[10] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *NeurIPS*.
[11] Jiyun Luo, Sicong Zhang, and Hui Yang. 2014. Win-win Search: Dual-agent Stochastic Game in Session Search. In *SIGIR*.
[12] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval. In *CIKM*.
[13] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *WWW Companion*.
[14] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of Go without human knowledge. *Nature* (2017).
[15] Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. MIT press Cambridge.
[16] 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 *SIGIR*.
[17] Zeng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Reinforcement learning to rank with Markov decision process. In *SIGIR*.
[18] Jun Xu and Hang Li. 2007. Adarank: a boosting algorithm for information retrieval. In *SIGIR*.
[19] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. [n. d.]. Directly optimizing evaluation measures in learning to rank. In *SIGIR*.
[20] Sicong Zhang, Jiyun Luo, and Hui Yang. 2014. A POMDP Model for Content-free Document Re-ranking. In *SIGIR*.
[21] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *WWW*.