From Greedy Selection to Exploratory Decision-Making: Diverse Ranking with Policy-Value Networks

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

ABSTRACT

The goal of search result diversification is to select a subset of documents from the candidate set to satisfy as many different subtopics as possible. In general, it is a problem of subset selection and selecting an optimal subset of documents is NP-hard. Existing methods usually formalize the problem as ranking the documents with *greedy* sequential document selection. At each of the ranking position the document that can provide the largest amount of additional information is selected. It is obvious that the greedy selections inevitably produce suboptimal rankings. In this paper we propose to partially alleviate the problem with a Monte Carlo tree search (MCTS) enhanced Markov decision process (MDP), referred to as M^{2}Div. In M^{2}Div, the construction of diverse ranking is formalized as an MDP process where each action corresponds to selecting a document for one ranking position. Given an MDP state which consists of the query, selected documents, and candidates, a recurrent neural network is utilized to produce the policy function for guiding the document selection and the value function for predicting the whole ranking quality. The produced raw policy and value are then strengthened with MCTS through *exploring the possible rankings* at the subsequent positions, achieving a better search policy for decision-making. Experimental results based on the TREC benchmarks showed that M^{2}Div can significantly outperform the state-of-the-art baselines based on greedy sequential document selection, indicating the effectiveness of the exploratory decision-making mechanism in M^{2}Div.

KEYWORDS

Diverse ranking; Markov decision process; Monte Carlo tree search

ACM Reference Format:
Yue Feng, Jun Xu*, Yanyan Lan, Jiafeng Guo, Wei Zeng, Xueqi Cheng . 2018. From Greedy Selection to Exploratory Decision-Making: Diverse Ranking with Policy-Value Networks. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3209979

* Corresponding author: Jun Xu

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3209979

1 INTRODUCTION

One important goal in many information retrieval tasks involves providing search results that covers a wide range of topics for a search query, called search result diversification [1]. The goal of search result diversification can be formalized as selecting a minimal subset of documents from the candidate set to cover as many different subtopics as possible. Since the novelty of a document depends on the other selected documents, selecting an optimal subset of documents amounts to the problem subset selection and its complexity is in general NP-hard.

Typical approaches treat search result diversification as ranking the documents based on their relevance as well as the novelty. Greedy sequential document selection has been widely adopted to construct the diverse document ranking, that is, the document ranking is constructed step by step. At each step the ranking model selects one document from the candidate set for the current ranking position. Usually, the document with the maximal amount of additional utility, a.k.a. marginal relevance, is selected.

A number of diverse ranking algorithms have been developed under the greedy document selection framework. Different algorithms utilize different criteria to estimate the additional utility a candidate document can provide. For example, the representative approach of maximal marginal relevance (MMR) [3] uses the sum of the query-document relevance and the maximal document distance (referred to as marginal relevance) as the utility. xQuAD [25] defines the utility so as to explicitly account for the relationship between documents retrieved for the original query and the possible subqueries. In recent years, machine learning based methods have been proposed for conducting diverse ranking [23, 31, 32, 34, 36, 39]. The relational learning to rank (R-LTR) [39] and its variations [31, 32, 34] define the utilities based on the relevance features and the novelty features. MDP-DIV adapted the Markov decision process (MDP) to model the document ranking process. The utility of a document is estimated based on the MDP state, which consists of the query, the preceding documents, and the remaining candidates [33].

The greedy sequential document selection simplifies the ranking process and can accelerate the online ranking. However, the rankings produced by greedy document selection are inevitably suboptimal. At each ranking position, the greedy selection mechanism only considers the possibilities at the current ranking position (i.e., estimates the utility of each candidate document if it were selected). Thus, greedy document selection will select the locally optimal document at each ranking position. However, a sequence of the locally optimal documents cannot lead to the globally optimal diverse ranking, because the utilities of the documents are not independent. The selection of a document at one position will change

the utilities of the remaining candidate documents and thereafter affects the subsequent decisions. In general the ranking algorithm need to explore the whole ranking space if the optimal ranking is mandatory. However, this is usually infeasible in real applications because of the huge space size: there exist $N!$ different rankings for N documents.

Inspired by the success and methodology of the AlphaGo [27] and AlphaGo Zero [28] for the Game of Go, in this paper we propose to enhance the MDP model for diverse ranking [33] with the Monte Carlo tree search (MCTS), for alleviating the suboptimal ranking problem. The new ranking model, referred to as M^{2}Div (stands for MCTS enhanced MDP for Diverse ranking), makes use of an MDP to model the sequential document selection process of diverse ranking. At each time step (corresponding to a ranking position), based on the user query and the preceding document ranking, a recurrent neural network (RNN) is used to produce the policy (a distribution over the candidate documents) for guiding the document selection and the value for estimating the whole document ranking quality (e.g., in terms of α-NDCG@M). To alleviate the problem of suboptimal diverse ranking, in stead of greedily selecting a document with the predicted raw policy, M^{2}Div conducts an exploratory decision making: an MCTS is conducted to explore the possible document rankings at the subsequent positions, resulting a strengthened search policy for conducting the real document selection at current position. Since it has explored more future possible document rankings, the search policy has higher probability to select a globally optimal document than the predicted raw policy. Moving to the next iteration, the above process is continued until the candidate set is empty.

Reinforcement learning is used to train the model parameters. In the training phase, at each training iteration and for each training query, an MCTS guided by the current policy function and value function is conducted at each ranking position. The MCTS produces a search policy for the document selection. Then the model parameters are adjusted to minimize the loss function. The loss function consists of two terms: 1) the squared error between the predicted value and the final quality of the whole document ranking in terms of α-NDCG@M; and 2) the cross entropy of the predicted raw policy and the search policy for document selection. Stochastic gradient descent is utilized for conducting the optimization.

To evaluate the effectiveness of M^{2}Div, we conducted experiments on the basis of TREC benchmark datasets. The experimental results showed that M^{2}Div can significantly outperform the state-of-the-art diverse ranking approaches that using greedy sequential decision making, including the heuristic based diverse ranking methods of MMR and xQuAD, and the machine learning based diverse ranking methods of PAMM and MDP-DIV. We analyzed the results and showed that the exploratory decision-making mechanism in M^{2}Div does help to improve the ranking performances.

2 RELATED WORK

2.1 Search result diversification

It is a common practice to formalize the construction of a diverse ranking list in search as a process of greedy sequential decision making. Existing research focus on designing effective criteria to estimate the utility a document can provide. Carbonell and Goldstein [3] proposed the maximal marginal relevance criterion, which is a linear combination of the query-document relevance and the document novelty, to select the document. xQuAD [24] directly models different aspects of a query and estimates the utility as the relevance of the retrieved documents to each identified aspects. Hu et al. [13] proposed a utility function that explicitly leverages the hierarchical intents of queries and selects the documents that maximize diversity in the hierarchical structure. See also [2, 4, 7, 9–11, 22, 29]

Machine learning techniques have been applied to construct diverse ranking, also adopting the greedy sequential decision making as the basic framework. The key problem becomes how to automatically learn the utility function on the basis of training queries. Some researchers define the utility as a linear combination of the handcrafted relevance features and novelty features [23, 31, 34, 39]. The novelty term in the utility function can be modeled with the deep learning model of neural tensor networks [32]. Jiang et al. used recurrent neural networks and max-pooling to model subtopic information explicitly with the attention mechanism [14, 15]. Xia et al. [33] proposed to model the dynamics of the document utility with MDP and learning the model parameters with policy gradient. Other learning approaches please refer to [18, 21, 23, 34, 35, 37].

2.2 Reinforcement learning for IR

The reinforcement learning has been widely used in variant IR applications. For example, in [20], a win-win search framework based on partially observed Markov decision process (POMDP) is proposed to model session search as a dual-agent stochastic game. In the model, the state of the search users are encoded as a four hidden decision making states. In [38], the log-based document re-ranking is modeled as a POMDP. [33] and [30] propose to model the process of constructing a document ranking with MDP, for the ranking tasks of search result diversification and relevance ranking, respectively. Muti-armed bandit, another type of reinforcement learning model, is also widely applied to rank learning. For example, [23] proposes two online learning bandit algorithms to learn a diverse ranking of documents based on users clicking behaviors. [37] formalizes the interactively optimizing of information retrieval systems as a dueling bandit problem and [16] proposes cascading bandits to identify K most attractive document for users. See also [12]

Reinforcement learning models are also used for building recommender systems. For example, [26] designs 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 [19] proposes POMDP-Rec, a neural-optimized POMDP algorithm, for building a collaborative filtering recommender system.

In this paper, we also adopt the reinforcement learning model of MDP to formalize the diverse ranking process in search result diversification.

3 MDP AND MCTS

We employ Markov decision process and Monte Carlo tree search to model diverse ranking process.

3.1 Markov decision process

MDP provides a mathematical framework for modeling the sequential decision making process with the agent-environment interface. The key components of an MDP include:

States $\mathcal{S}$ is a set of states. For instance, in this paper we define the state as a tuple consists of the query, the preceding document ranking, and the candidate documents.

Actions $\mathcal{A}$ is a discrete set of actions that an agent can take. The actions available may depend on the state s, denoted as $\mathcal{A}(s)$.

Policy p describes the behaviors of an agent, which is a probabilistic distribution over the possible actions. **p** is usually optimized to maximize the long term return.

Transition T is the state transition function $s_{t+1} = T(s_t, a_t)$ which specifies a function that maps a state s_t into a new state s_{t+1} in response to the action selected a_t.

Value: state value function $V : \mathcal{S} \rightarrow \mathbb{R}$ is a function that predicts the long term return of the whole episode, on the basis of the current state s under the policy **p**.

An MDP model is running as follows: the agent and environment interact at each of a sequence of discrete time steps, $t = 0, 1, 2, \cdots$. At each time step t the agent receives some representation of the environment's state, $s_t \in \mathcal{S}$, and on that basis selects an action $a_t \in \mathcal{A}(s_t)$. One time step later, in part as a consequence of its action, the agent finds itself in a new state $s_{t+1} = T(s_t, a_t)$. Usually the goal of reinforcement learning is to achieve maximum long term return, that is, to maximize the value $V(s_0)$.

3.2 Monte Carlo tree search

The decisions made by an MDP (e.g., selecting the most confident actions according to the policy) may get suboptimal results. Theoretically the system has to explore the whole space of decision sequences to get a global optimal result. However, this is usually not feasible. MCTS is a tool to conduct heuristic search inside the whole space, more likely to produce a better decision sequence than that of produced by the greedy decisions.

Given the time step t, the policy function **p**, and the state value function V, MCTS aims at searching a strengthened policy for making better decisions. The MCTS consists of four phases: selection, expansion, simulation, and back-propagation:

Selection: Starting at the root node R, MCTS recursively selects the optimal child nodes until a leaf node L is reached.

Expansion: If L is not a terminal node (i.e. it does not end the episode) then create one or more child nodes for L (each corresponds a possible action) and select one child node C according to the predicted policy.

Simulation/Evaluation: MCTS runs a simulation from C until a result is achieved. In the AlphaGo Zero [28] the simulation is replaced with the value function for accelerating the tree search. That is, the result of simulation is estimated by the value function.

Back-propagation: Update the statistics stored in the current move sequence with the simulation or estimated result.

The MCTS outputs a search policy $\boldsymbol{\pi}$ over the actions, which is utilized to choose the action at time step t. The iteration is continued until the whole episode is generated.

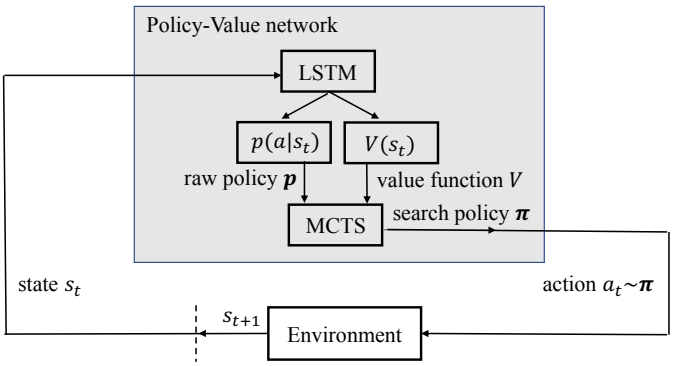

Figure 1: The agent-environment interaction of M^{2}Div.

4 DIVERSE RANKING WITH POLICY-VALUE NETWORKS

In this section, we introduce the proposed M^{2}Div model, which makes use of MDP and MCTS for modeling the diverse ranking process and for strengthening the policy for selecting documents at each of the MDP iteration, respectively. The agent-environment interaction of M^{2}Div is illustrated in Figure 1. Each of the MDP time step corresponds to a ranking position. At time step t ($t = 0, 1, \cdots$), the policy-value network receives the environment state s_t and makes use of an LSTM to produce the representation of the state s_t. After that, guided by the current policy function **p** and value function V, an MCTS search is executed. The output of MCTS is a strengthened new search policy $\boldsymbol{\pi}$. The action a_t is then selected according to the strengthened policy $\boldsymbol{\pi}$, which chooses a document from the candidate set and places it to the ranking position $t + 1$. Moving to the next time step $t + 1$, the system finds itself in a new state s_{t+1} and the process is repeated until all of the documents are ranked.

4.1 MDP formulation of diverse ranking

Suppose we are given a query $\mathbf{q}$, which is associated with a set of retrieved documents $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_M\} \subseteq \mathcal{X}$, where both the query $\mathbf{q}$ and the documents $\mathbf{x}_i$ are represented with their preliminary representations, i.e., the vectors learned by the doc2vec model [17], and $\mathcal{X}$ is the set of all possible documents. The goal of diverse ranking is to construct a model that can rank the documents so that the top ranked documents cover a wide range of subtopics of a search query.

The construction of a diverse ranking can be considered as a process of sequential decision making with an MDP in which each time step corresponds to a ranking position. The states, actions, transition function, value function, and policy function of the MDP are set as:

States $\mathcal{S}$**:** We design the state at time step t as a triple $s_t = [\mathbf{q}, \mathcal{Z}_t, X_t]$, where $\mathbf{q}$ is the preliminary representation of the user issued query; $\mathcal{Z}_t = \{\mathbf{x}_{(n)}\}_{n=1}^{t}$ is the sequence of t preceding documents, where $\mathbf{x}_{(n)}$ is the document ranked at position n; X_t is the set of candidate documents. At the beginning ($t = 0$), the state is initialized as $s_0 = [\mathbf{q}, \emptyset, X]$, where $\emptyset$ is the empty sequence and X contains all of the M retrieved candidate documents.

Actions $\mathcal{A}$: At each time step t, the $\mathcal{A}(s_t)$ is the set of actions the agent can choose, each corresponds to a document from X_t. That is, the action $a_t \in \mathcal{A}(s_t)$ at the time step t selects a document $\mathbf{x}_{m(a_t)} \in X_t$ for the ranking position $t+1$, where $m(a_t)$ is the index of the document selected by a_t.

Transition T: The transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is defined as follows:

$$\begin{aligned} s_{t+1} = T(s_t, a_t) &= T([\mathbf{q}, \mathcal{Z}_t, X_t], a_t) \\ &= \left[\mathbf{q}, \mathcal{Z}_t \oplus \{\mathbf{x}_{m(a_t)}\}, X_t \setminus \{\mathbf{x}_{m(a_t)}\}\right], \end{aligned} \tag{1}$$

where $\oplus$ appends $\mathbf{x}_{m(a_t)}$ to $\mathcal{Z}_t$ and $\setminus$ removes $\mathbf{x}_{m(a_t)}$ from X_t. At each time step t, based on state s_t the system chooses an action a_t. Then, the system moves to time step $t+1$ and the system transits to a new state s_{t+1}: first, the query $\mathbf{q}$ is kept unchanged; second, the selected document is appended to the end of $\mathcal{Z}_t$, generating a new document sequence; finally, the selected document at step t is removed from the candidate set: $X_{t+1} = X_t \setminus \{\mathbf{x}_{m(a_t)}\}$. Thus, the number of actions the agent can choose at $t+1$ is reduced by one.

Value function V: The state value function $V : \mathcal{S} \rightarrow \mathbb{R}$ is a scalar evaluation, estimating the quality of the whole document ranking (an episode) based on the input state. The value function is learned so as to approximate a predefined evaluation measure (e.g., α-NDCG@M).

In this paper, we make use of the long short term memory (LSTM) to summarize the input state s as a real vector, and then define the value function as nonlinear transformation of the weighted sum of the LSTM outputs:

$$V(s) = \sigma\left(\langle \mathbf{w}, \text{LSTM}(s) \rangle + b_v\right), \tag{2}$$

where $\mathbf{w}$ and b_v are the weight vector and the bias to be learned during training, and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the nonlinear sigmoid function. The deep neural network model LSTM: $\mathcal{S} \rightarrow \mathbb{R}^L$ maps a state to a real vector where L is the number of dimensions. Given $s = [\mathbf{q}, \mathcal{Z} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\}, X_t]$, where $\mathbf{x}_k (k = 1, \cdots, t)$ is the document ranked at k-th position and represented with its doc2vec embedding. LSTM outputs a representation $\mathbf{h}_k$ for position k as:

$$\begin{aligned} \mathbf{f}_k &= \sigma(\mathbf{W}_f \mathbf{x}_k + \mathbf{U}_f \mathbf{h}_{k-1} + \mathbf{b}_f), \\ \mathbf{i}_k &= \sigma(\mathbf{W}_i \mathbf{x}_k + \mathbf{U}_i \mathbf{h}_{k-1} + \mathbf{b}_i), \\ \mathbf{o}_k &= \sigma(\mathbf{W}_o \mathbf{x}_k + \mathbf{U}_o \mathbf{h}_{k-1} + \mathbf{b}_o), \\ \mathbf{c}_k &= \mathbf{f}_k \circ \mathbf{c}_{k-1} + \mathbf{i}_k \circ \tanh(\mathbf{W}_c \mathbf{x}_k + \mathbf{U}_c \mathbf{h}_{k-1} + \mathbf{b}_c), \\ \mathbf{h}_k &= \mathbf{o}_k \circ \tanh(\mathbf{c}_k), \end{aligned}$$

where $\mathbf{h}$ and $\mathbf{c}$ are initialized with the query $\mathbf{q}$:

$$[\mathbf{h}_0, \mathbf{c}_0] = [\sigma(\mathbf{V}_h \mathbf{q}), \sigma(\mathbf{V}_c \mathbf{q})],$$

operator "$\circ$" denotes the element-wise product and "σ" is applied to each of the entries; the variables $\mathbf{f}_k, \mathbf{i}_k, \mathbf{o}_k, \mathbf{c}_k$ and $\mathbf{h}_k$ denote the forget gate's activation vector, input gate's activation vector, output gate's activation vector, cell state vector, and output vector of the LSTM block, respectively. $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o, \mathbf{U}_f, \mathbf{U}_i, \mathbf{U}_o, \mathbf{V}_h, \mathbf{V}_c, \mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o$ are weight matrices and bias vectors need to be learned during training. The output vector and cell state vector at the t-th cell are concatenated as the output of LSTM:

$$\text{LSTM}(s) = \left[\mathbf{h}_t{}^T, \mathbf{c}_t{}^T\right]^T. \tag{3}$$

Policy function p: The policy $\mathbf{p}(s)$ defines a function that takes the state as input and output a distribution over all of the possible actions $a \in \mathcal{A}(s)$. Specifically, each probability in the distribution is a normalized soft-max function whose input is the bilinear product of the LSTM defined in Equation (3) and the selected document:

$$p(a|s) = \frac{\exp\left\{\mathbf{x}_{m(a)}^T \mathbf{U}_p \text{ LSTM}(s)\right\}}{\sum_{a' \in \mathcal{A}(s)} \exp\left\{\mathbf{x}_{m(a')}^T \mathbf{U}_p \text{ LSTM}(s)\right\}},$$

where $\mathbf{U}_p$ is the parameter in the bilinear product. Thus, the policy function $\mathbf{p}(s)$ is:

$$\mathbf{p}(s) = \langle p(a_1|s), \cdots, p(a_{|\mathcal{A}(s)|}|s) \rangle. \tag{4}$$

4.2 Strengthening policy with MCTS

Selecting documents greedily with the predicted raw policy $\mathbf{p}$ in Equation (4) may lead to suboptimal rankings because the policy only summarizes the historical information and has no idea on how the action a_t will influent the future decisions. Let's use an example to show the limitation of greedy selection. Suppose that the query q has 6 subtopics, and the 3 retrieved candidate documents d_1, d_2, and d_3 cover the subtopics of $\{1, 2, 3, 6\}$, $\{1, 2, 5\}$, and $\{1, 2, 6\}$, respectively. The greedy selection with raw policy prefers d_1 for the first position, as it covers the most number of subtopics. Thus, the document ranking constructed with the greedy policy could be $[d_1, d_2, d_3]$ and S-recall@2 = $\frac{5}{6}$. However, if the ranking algorithm could explore (part of) the whole ranking space, it will found a better document ranking in terms of S-recall@2: $[d_2, d_3, d_1]$ with S-recall@2 = 1. The example clearly indicate that greedy selection could lead to suboptimal rankings.

To alleviate the issue, following the practices in AlphaGo [27] and AlphaGo Zero [28], we propose to conduct lookahead searches in the ranking space with MCTS. Specifically, at each ranking position t, an MCTS search is executed, guided by the current policy function $\mathbf{p}$ and value function V, and output a strengthened new search policy $\boldsymbol{\pi}$. It is believed that the search policy $\boldsymbol{\pi}$ will select a much better action (document) for the ranking position than that of selected by the raw policy $\mathbf{p}$ in Equation (4). This is because the lookahead MCTS tries to explore the whole ranking space and can partially alleviate the suboptimal ranking problem.

Figure 2 illustrates the MCTS process and Algorithm 1 shows the details. Each node of the tree corresponds to an MDP state. The tree search algorithm takes a root node s_R, number of search times K, value function V, policy function $\mathbf{p}$, human labels J, and the evaluation measure R as inputs. Note that J and R are only used at the training time. The algorithm iterate K times and outputs a strengthened search policy $\boldsymbol{\pi}$ for selecting a document for root node s_R. Suppose that each edge $e(s, a)$ (the edge from state s to the state $T(s, a)$) of the MCTS tree stores an action value $Q(s, a)$, visit count $N(s, a)$, and prior probability $P(s, a)$. The raw policy $\mathbf{p}(s_R)$ at the root state s_R is strengthened with the following steps:

Selection (line 3 to line 7 of Algorithm 1): Each of the K iterations starts from the root state s_R and iteratively selects the documents that maximize an upper confidence bound. Specifically, at each time step t of each simulation, an action a_t is selected from

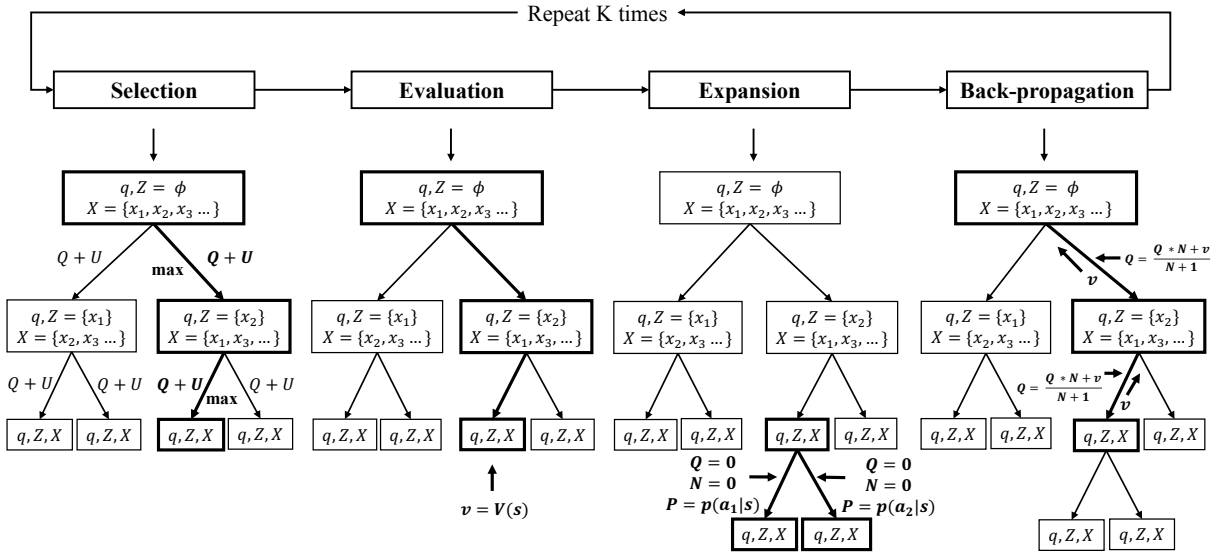

Figure 2: The MCTS process for calculating the strengthened search policy $\boldsymbol{\pi}$.

state s_t so as to maximize action value plus a bonus:

$$a_t = \arg\max_a (Q(s_t, a) + \lambda U(s_t, a)), \tag{5}$$

where $\lambda \geq 0$ is the tradeoff coefficient, and the bonus $U(s_t, a)$ is defined as

$$U(s_t, a) = p(a|s_t) \frac{\sqrt{\sum_{a' \in \mathcal{A}(s_t)} N(s_t, a')}}{1 + N(s_t, a)},$$

where $p(a|s_t)$ is the predicted probability by the policy function $\mathbf{p}(s_t)$, $\mathcal{A}(s_t)$ is the set of available actions (documents) at state s_t. $U(s_t, a)$ is proportional to the prior probability but decays with repeated visits to encourage exploration.

Evaluation and expansion (line 8 to line 19 of Algorithm 1): When the traversal reaches a leaf node s_L, the node is evaluated either with the value function $V(s_L)$ or with the predefined performance measure if the node is the end of an episode and the human labels are available. Specifically, at the test phase or online ranking phase, there is no human label information available. s_L will be evaluated with the value function $V(s_L)$. At the training phase, if s_L is not the end of the episode, $V(s_L)$ is used to conduct the evaluation. If s_L is the last node of the episode (cannot be expanded), it is evaluated with the true performance measure (e.g., α-NDCG@M) at the position. Line 10 and line 18 of Algorithm 1 shows the evaluation details. Please note that following the practice in AlphaGo Zero program, we use the value function instead of rollouts for evaluating a node.

Then, the leaf node s_L may be expanded (line 11 to line 16 of Algorithm 1). Each edge from the leaf position s_L (corresponds to each action $a \in \mathcal{A}(s_L)$) is initialized as: $P(s_L, a) = p(a|s_L)$ (Equation (4)), $Q(s_L, a) = 0$, and $N(s_L, a) = 0$. In this paper all of the available actions of s_L are expanded.

Back-propagation and update (line 20 to line 28 of Algorithm 1): At the end of evaluation, the action values and visit counts of all traversed edges are updated. For each edge $e(s, a)$, the prior probability $P(s, a)$ is kept unchanged, and $Q(s, a)$ and $N(s, a)$ are updated:

$$\begin{aligned} Q(s, a) &\leftarrow \frac{Q(s, a) \times N(s, a) + V(s_L)}{N(s, a) + 1}, \\ N(s, a) &\leftarrow N(s, a) + 1. \end{aligned} \tag{6}$$

Calculate the strengthened search policy (line 29 to line 32 of Algorithm 1): After iterating K times, the strengthened search policy $\boldsymbol{\pi}$ for the root node s_R can be calculated according to the visit counts $N(s_R, a)$ of the edges starting from s_R:

$$\pi(a|s_R) = \frac{N(s_R, a)}{\sum_{a' \in \mathcal{A}(s_R)} N(s_R, a')}, \tag{7}$$

for all $a \in \mathcal{A}(s_R)$.

4.3 Training with reinforcement learning

The model has parameters $\boldsymbol{\Theta} = \{\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o, \mathbf{U}_f, \mathbf{U}_i, \mathbf{U}_o, \mathbf{U}_p, \mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o, \mathbf{V}_h, \mathbf{V}_c, \mathbf{w}\}$ to learn. In the training phase, suppose we are given N labeled training queries $\{(q^{(n)}, X^{(n)}, J^{(n)})\}_{n=1}^{N}$, where $J^{(n)}$ denotes the human labels on the documents, in the form of a binary matrix. $J^{(n)}(i, j) = 1$ if document $\mathbf{x}_i^{(n)}$ contains the j-th subtopic of $q^{(n)}$ and 0 otherwise.

Algorithm (2) shows the training procedure. First, the parameters $\boldsymbol{\Theta}$ is initialized to random weights in $[-1, 1]$. At each subsequent iteration, for each query, a ranking of documents is generated with current parameter setting. At each ranking position t, an MCTS search is executed, using previous iteration of value function and policy function, and a document $\mathbf{x}_{m(a_t)}$ is selected according to the search probabilities $\boldsymbol{\pi}_t$.

The ranking terminates when the candidate is empty or the ranking exceeds the maximum length defined by the predefined evaluation measure R. These sampled documents consist a permutation of documents $\boldsymbol{\tau}$, which is then evaluated with the evaluation measure to give a ground-truth return:

$$r = R(\boldsymbol{\tau}, J).$$

Algorithm 1 TreeSearch

Input: root state s_R, value function V, and policy function $\mathbf{p}$, number of search times K, trade-off parameter λ, human labels J, and performance evaluation function R
Output: Tree search probabilities $\boldsymbol{\pi}$

for $k = 0$ **to** $K - 1$ **do**
  $s_L \leftarrow s_R$
  {Selection}
  while s_L is not a leaf node **do**
    $a \leftarrow \arg\max_{a \in \mathcal{A}(s_L)} Q(s_L, a) + \lambda \cdot U(s_L, a)$ {Equation (5), using the P, Q, and N stored in the corresponding edges}
    $s_L \leftarrow$ child node pointed by edge (s_L, a)
  end while
  {Evaluation and expansion}
  if s_L can be expanded **then**
    $v \leftarrow V(s_L)$ {simulate v with value function V}
    for all $a \in \mathcal{A}(s_L)$ **do**
      Expand a new edge e which connects to a new node $s = [\mathbf{q}, s_L.\mathcal{Z} \oplus \{\mathbf{x}_{m(a)}\}, s_L.X \setminus \{\mathbf{x}_{m(a)}\}]$
      $e.P \leftarrow p(a|s_L)$ {initialize the prior probability}
      $e.Q \leftarrow 0$
      $e.N \leftarrow 0$
    end for
  else
    $v \leftarrow \begin{cases} V(s_L) & R = \text{NULL or } J = \emptyset \\ R(s_L.\mathcal{Z}, J) & \text{otherwise} \end{cases}$ {Case 1: predict with V at test phase (R =NULL or $J = \emptyset$); Case 2: directly set to the ground truth (e.g., α-NDCG@M) at the training phase}
  end if
  {Back-propagation}
  while $s_L \neq s_R$ **do**
    $s \leftarrow$ parent of s_L
    $e \leftarrow$ edge from s to s_L
    $e.Q \leftarrow \frac{e.Q \times e.N + v}{e.N+1}$ {Equation (6)}
    $e.N \leftarrow e.N + 1$
    $s_L \leftarrow s$
  end while
end for
{calculate tree search probabilities}
for all $a \in \mathcal{A}(s_R)$ **do**
  $\pi(a|s_R) \leftarrow \frac{e(s_R,a).N}{\sum_{a' \in \mathcal{A}(s_R)} e(s_R,a').N}$ {$e(s, a)$ is the edge from s to the state by taking action a}
end for
return $\boldsymbol{\pi}$

Here R can be any diverse ranking evaluation measure such as α-NDCG@M etc. The data generated at each time step $E = \{(s_t, \boldsymbol{\pi}_t)\}_{t=1}^{T}$ and the final evaluation r are utilized as the ground-truth signals in training for adjusting the value function. The model parameters are adjusted to minimize the error between the predicted value $V(s_t)$ and the evaluation of the ranking in terms of the chosen evaluation measure, and to maximize the similarity of the raw policy $\mathbf{p}(s_t)$ to the search policy $\boldsymbol{\pi}_t$. Specifically, the parameters $\boldsymbol{\Theta}$ are adjusted by gradient descent on a loss function ℓ that sums over

Algorithm 2 Train diverse ranking model

Input: Labeled training set $D = \{(\mathbf{q}^{(n)}, X^{(n)}, J^{(n)})\}_{n=1}^{N}$, learning rate η, number of search steps K, MCTS trade-off parameter λ, and evaluation measure R
Output: $\boldsymbol{\Theta}$

1: Initialize $\boldsymbol{\Theta} \leftarrow$ random values in $[-1, 1]$
2: **repeat**
3: **for all** $(\mathbf{q}, X, J) \in D$ **do**
4: $s \leftarrow [\mathbf{q}, \emptyset, X]$
5: $M \leftarrow |X|$
6: $E = ()$ {empty episode}
7: **for** $t = 0$ **to** $M - 1$ **do**
8: $\boldsymbol{\pi} \leftarrow$ TreeSearch$(s, V, \mathbf{p}, K, \lambda, J, R)$ {Algorithm (1): tree search using s as root, with current $\boldsymbol{\Theta}$}
9: $a = \arg\max_{a \in \mathcal{A}(s)} \pi(a|s)$ {select the best document}
10: $\tau(t+1) \leftarrow m(a)$ {document $\mathbf{x}_{m(a)}$ is ranked at $t+1$}
11: $E \leftarrow E \oplus \{(s, \boldsymbol{\pi})\}$
12: $s \leftarrow [\mathbf{q}, s.\mathcal{Z} \oplus \{\mathbf{x}_{m(a)}\}, s.X \setminus \{\mathbf{x}_{m(a)}\}]$
13: **end for**
14: $r \leftarrow R(\boldsymbol{\tau}, J)$ {evaluating the generalized ranking}
15: $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \eta \frac{\partial \ell(E,r)}{\partial \boldsymbol{\Theta}}$ {Update parameters. ℓ is defined in Equation (8)}
16: **end for**
17: **until** converge
18: **return** $\boldsymbol{\Theta}$

the mean-squared error and cross-entropy losses, respectively:

$$\ell(E, r) = \sum_{t=1}^{|E|} \left((V(s_t) - r)^2 + \sum_{a \in \mathcal{A}(s_t)} \pi_t(a|s_t) \log \frac{1}{p(a|s_t)} \right). \quad (8)$$

Figure 3 illustrates the construction of the loss given a training query. The model parameters are trained by back propagation and stochastic gradient descent. Specifically, we use AdaGrad [8] on all parameters in the training process.

Please note that following the practices in [28], the search tree constructed at the t-th iteration (line 8 to line 12 of Algorithm 2) is reused at subsequent steps: the child node corresponding to the selected document (chosen action) becomes the new root node; the subtree below this child is retained along with all its statistics, while the remainder of the tree is discarded.

4.4 Online ranking

The construction of a diverse ranking for an online query is shown in Algorithm 3. Given a user query $\mathbf{q}$, a set of M retrieved documents X, the system state is initialized as $s_0 = [\mathbf{q}, \mathcal{Z}_0 = \emptyset, X_0 = X]$. Then, at each of the time steps $t = 0, \cdots, M - 1$, the agent receives the state $s_t = [\mathbf{q}, \mathcal{Z}_t, X_t]$ and searches the policy $\boldsymbol{\pi}$ with MCTS, on the basis of the value function V and policy function $\mathbf{p}$. Then, it chooses an action a according $\boldsymbol{\pi}$, selects the document $\mathbf{x}_{m(a)}$ from the candidate set, and places it to the rank $t + 1$. Moving to the next step $t + 1$, the state becomes $s_{t+1} = [\mathbf{q}, \mathcal{Z}_{t+1}, X_{t+1}]$. The process is repeated until the candidate set becomes empty.

Considering that the MCTS is time consuming and may be infeasible in some online ranking tasks, the online ranking algorithm can also skip the tree search and directly use the raw policy for

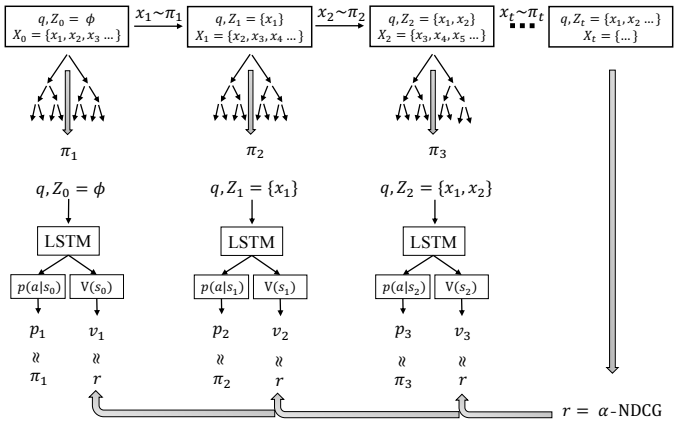

Figure 3: Construction of loss function for a training query.

Algorithm 3 Diverse ranking with Monte Carlo tree search

Input: query $\mathbf{q}$, documents $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$, number of search steps K, MCTS trade-off parameter λ, value function V, and policy function $\mathbf{p}$

Output: Permutation of documents $\boldsymbol{\tau}$

1: $s \leftarrow [\mathbf{q}, \Phi, X]$
2: $M \leftarrow |X|$
3: **for** $t = 0$ **to** $M - 1$ **do**
4: $\boldsymbol{\pi} \leftarrow \begin{cases} \text{TreeSearch}(s, V, \mathbf{p}, K, \lambda, \emptyset, \text{NULL}) & \text{with MCTS,} \\ \mathbf{p}(s) & \text{without MCTS} \end{cases}$
{Case 1: ranking with search policy. No human labels and evaluation function are available at the test/online ranking phase; Case 2: ranking with raw policy for accelerating online ranking speeds.}
5: $a \leftarrow \arg\max_{a \in \mathcal{A}(s)} \pi(a|s)$
6: $\tau(t+1) \leftarrow m(a)$ {document $\mathbf{x}_{m(a)}$ is ranked at $t+1$}
7: $[\mathbf{q}, \mathcal{Z}, X] \leftarrow s$
8: $s \leftarrow [\mathbf{q}, \mathcal{Z} \oplus \{\mathbf{x}_{m(a)}\}, X \setminus \{\mathbf{x}_{m(a)}\}]$
9: **end for**
10: **return** $\boldsymbol{\tau}$

ranking, denoted with "without MCTS" in the line 4 of Algorithm 3. In the experiments, we observed that the M^{2}Div without MCTS can still outperform the baselines. This is because the training process of M^{2}Div can generate high quality episodes to train the model parameters with the help of MCTS, which leads to more accurate policy function $\mathbf{p}$.

4.5 Difference with AlphaGo Zero

M^{2}Div is inspired by the AlphaGo Zero program. It enjoys a number of merits from AlphaGo Zero, including the shared neural network for estimating policies and values, lookahead MCTS for strengthening the raw policy, and the compounded loss that simultaneously optimizes the value function and the policy function etc. However, M^{2}Div has made a number of fundamental modifications for search result diversification.

First, the formalizations of the tasks are different. AlphaGo Zero formalizes the playing of Go as an alternating Markov game where each action corresponds a move, the states are the raw board positions, the value function approximates the probability of winning, and the next state depends not only on the chosen action but also on the move by the opponent. M^{2}Div, on the other hand, formalizes the ranking of documents as planning with an MDP where each action corresponds to a document selection. The MDP states consists of the query, the preceding document ranking, and the remaining candidates. The value function approximates a diverse ranking evaluation measure (e.g., α-NDCG@M). The state transition is fully determined by the current state and the chosen action.

Second, the supervision signals for learning the model parameters are different. AlphaGo Zero uses the results of self-play (-1 for loss, 0 for draw, and 1 for win) as the supervision signals and the value function is fitted to the results. The task of diverse ranking, however, is not a Markov game. It is difficult (also unnecessary) to execute the self-play. In the training phase, M^{2}Div resorts to the human labels and the predefined evaluation measure to generate supervision information. Specifically, the α-NDCG@M of each generated episode is calculated and used as the ground-truth to fit the value function (the first part of Equation (8)). In this way, M^{2}Div drives the training process so as to directly optimize the evaluation measure of α-NDCG@M. Note that α-NDCG@M can be replaced with any other diverse ranking evaluation measures.

Third, the shared deep neural networks for calculating the policies and values are different. AlphaGo Zero makes use of a residual network which takes the raw board position as its inputs and outputs the a probability distribution over moves, and a probability of the current player winning in position. The raw boards can be considered as some fixed sized images. In M^{2}Div, the MDP states consist of the queries and sequences of documents etc. Residual network cannot handle the state data as the queries/documents are raw texts, and the lengths of the document sequences vary in different time steps. To address the issue, M^{2}Div first represents the query and document sequence as the state vector of an LSTM (Equation 3). The policy and the value are then calculated on the basis of the representations outputted by the LSTM.

5 EXPERIMENTS

We conducted experiments to test the performances of M^{2}Div using a combination of four TREC benchmark datasets: TREC 2009 ~ 2012 Web Track datasets (WT2009, WT2010, WT2011, and WT2012).

5.1 Experimental settings

In our experiments, for effective training of the model parameters and following the practices in [33], we combined four TREC datasets and constructed a new dataset with 200 queries and in total about 45,000 labeled documents. Each query includes several subtopics identified by the TREC assessors. The document relevance labels are made at the subtopic level and the labels are binary[1].

All the experiments were carried out on the ClueWeb09 Category B data collection[2], which is comprised of 50 million English web documents. Porter stemming, tokenization, and stop-words removal (using the INQUERY list) were applied to the documents as preprocessing. We conducted 5-fold cross-validation experiments.

[1] WT2011 has graded judgements and we treat them as binary in the experiments.
[2] http://boston.lti.cs.cmu.edu/data/clueweb09

We randomly split the queries into five even subsets. At each fold, three subsets were used for training, one was used for validation, and one was used for testing. The results reported were the average over the five trials.

The TREC official diversity evaluation metrics of α-NDCG [6] and ERR-IA [5] were used in the experiments, including. They measure the diversity of a result list by explicitly rewarding diversity and penalizing redundancy observed at every rank. Following the default settings in official TREC evaluation program, the parameter α in these evaluation measures are set to 0.5.

We compared M^{2}Div with several state-of-the-art baselines in search result diversification:

MMR [3]: a heuristic approach in which the document is selected according to maximal marginal relevance.

xQuAD [24]: a representative method which explicitly models different aspects underlying original query in form of subqueries.

PM-2 [7]: a method of optimizing proportionality for search result diversification.

We also compared MDP-DIV with the learning methods:

SVM-DIV [36]: a learning approach which utilizes structural SVMs to optimize the subtopic coverage.

R-LTR [39]: a learning approach developed in the relational learning to rank framework.

PAMM [31]: a learning approach that directly optimizes diversity evaluation measure using structured Perceptron.

NTN-DIV [32]: a learning approach which automatically learns novelty features based on neural tensor networks.

MDP-DIV [33]: a state-of-the-art learning approach which uses an MDP for modeling the diverse ranking process. Following the practice in [33], we configured the reward function in MDP-DIV as α-DCG and the discounting parameter $\gamma = 1$.

M^{2}Div, and the baselines of MDP-DIV and NTN-DIV need preliminary representations of the queries and the documents as their inputs. Following the practices in [33], in the experiments we used the query vector and document vector generated by the doc2vec [17] to represent the document. Doc2vec model was trained on all of the documents in Web Track dataset and the number of vector dimensions were set to 100. For training the model, we used the distributed bag of words model[3]. The learning rate was set to 0.025 and the window size was set to 8.

The M^{2}Div also has some parameters to tune: the training evaluation measure function R was set to α-NDCG@5, making the value function V to approximate α-NDCG@5. In the training phrase, the MCTS also was set to control the maximal length of the search depth less than 5. That is, the condition at line 9 of Algorithm 1 is true if the length of the tree depth is less than 5 and the candidate set is not empty. In all of the experiments, the learning rate η, the number of search times K, the tree search trade-off parameter λ, and the number of LSTM hidden units h were tuned based on the validation set and set to $\eta = 0.01, K = 5000, \lambda = 3.0$, and $h = 5$.

In online ranking phase, M^{2}Div has two versions: selecting the documents with the raw policy $\mathbf{p}$ or with the MCTS strengthened policy $\boldsymbol{\pi}$. These two versions are respectively denoted as "M^{2}Div(without MCTS)" and "M^{2}Div(with MCTS)". The source code of M^{2}Div is available at https://github.com/sweetalyssum/M2DIV.

[3]http://radimrehurek.com/gensim/tutorial.html

Table 1: Performance comparison of all methods on TREC web track datasets.

Method	α-NDCG@5	α-NDCG@10	ERR-IA@5	ERR-IA@10
MMR	0.2753	0.2979	0.2005	0.2309
xQuAD	0.3165	0.3941	0.2314	0.2890
PM-2	0.3047	0.3730	0.2298	0.2814
SVM-DIV	0.3030	0.3699	0.2268	0.2726
R-LTR	0.3498	0.4132	0.2521	0.3011
PAMM(α-NDCG)	0.3712	0.4327	0.2619	0.3029
NTN-DIV(α-NDCG)	0.3962	0.4577	0.2773	0.3285
MDP-DIV(α-DCG)	0.4189	0.4762	0.2988	0.3494
M^{2}Div(without MCTS)	0.4386*	0.4835	0.3435*	0.3668*
M^{2}Div(with MCTS)	**0.4424***	**0.4852**	**0.3459***	**0.3686***

5.2 Experimental results

Table 1 reports the performances of our approach and all of the baseline methods in terms of diversity performance metrics including α-NDCG@5, α-NDCG@10, ERR-IA@5, and ERR-IA@10. Boldface indicates the highest scores among all runs. From the results we can see that, in terms of the four diversity evaluation metrics, "M^{2}Div (with MCTS)" and "M^{2}Div (without MCTS)" outperformed all of the baseline methods, including the heuristic method of MMR, xQuAD, PM-2 and learning methods of SVM-DIV, R-LTR, PAMM(α-NDCG), NTN-DIV(α-NDCG), and MDP-DIV(α-DCG). We conducted significant testing (t-test) on the improvements of our approaches over the best baseline MDP-DIV(α-DCG). The results indicate that most of the improvements are significant (p-value < 0.05 and denoted with '*' in Table 1).

From the results we can see that "M^{2}Div (without MCTS)", which did not conduct MCTS at the online time, still outperformed all of the baselines including "MDP-DIV(α-DCG)", indicating that the MCTS conducted at the training time can generate better episodes to estimate the model parameters, achieving better raw policy $\mathbf{p}$ for ranking. Note that "M^{2}Div(with MCTS)", which conducted MCTS at the online time, further improved the ranking accuracies and performed the best among all of the methods. The results indicate that the MCTS can improve the raw policies $\mathbf{p}$ at both the training and the online ranking.

5.3 Discussion

In this section, we conducted experiments to investigate how M^{2}Div works and why it can outperform the baselines, using the experimental results on the first fold of the data as example.

5.3.1 Effects of Monte Carlo tree search. One key step in M^{2}Div is the MCTS which outputs the search policy $\boldsymbol{\pi}$. It is very likely that the search policy $\boldsymbol{\pi}(s)$ are better than raw policy $\mathbf{p}(s)$ in terms of choosing optimal documents.

We conducted experiments to show the effectiveness of the MCTS in online ranking. Specifically, based on the trained M^{2}Div model, we tracked the online ranking process for query number 148 ("martha stewart and imclone"). The red real curve in Figure 4 shows the α-NDCG values of the document ranking generated by $\boldsymbol{\pi}$. The blue dashed lines in Figure 4 show the α-NDCG values at

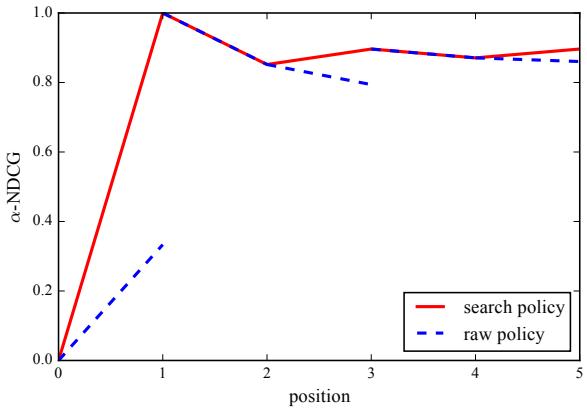

Figure 4: α-NDCG at the top 5 positions for the ranking generated by the search policy π (red real curve) for query number 148. Blue dashed lines show the α-NDCG values if the documents were selected by the raw policy p.

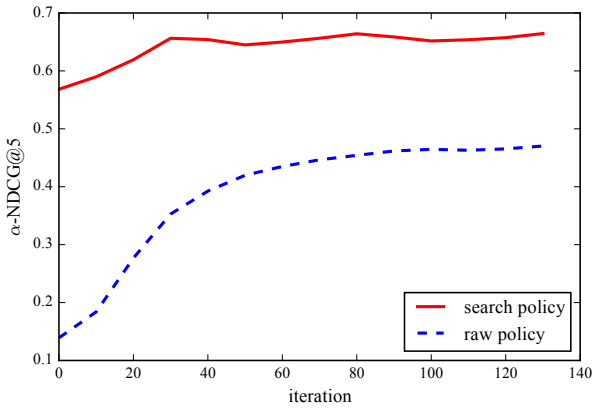

Figure 5: Performance curves of the document rankings generated with the raw policy p and with the search policy π, w.r.t. the training iterations. The curves illustrate the averaged performances over all of the training queries.

these 5 positions if the corresponding document was chosen by the raw policy **p** (note the preceding documents are still selected by $\boldsymbol{\pi}$). From the results, we can see that $\boldsymbol{\pi}$ improved **p** at the positions of 1, 3, and 5, showing the effectiveness of MCTS at the online ranking.

We also conducted experiments to show the effectiveness of the MCTS in offline training. Specifically, at the end of each training iteration, based on current ranking model, we tested the performances of the document ranking constructed with the raw policy **p** and with the MCTS search policy $\boldsymbol{\pi}$, in term of α-NDCG@5 on the training queries. Figure 5 shows the averaged α-NDCG@5 scores among all of the training queries at each training iterations. It is not surprise that the document rankings generated by the search policy $\boldsymbol{\pi}$ are superior to that of generated by the raw policy **p**, at all of the training iterations. The results indicate that the lookahead MCTS can generate better diverse document rankings for training the model parameters, at all of the training iterations. The results also partially explained why "M^{2}Div(without MCTS)", which did not conduct MCTS at the online time, can still outperform the baselines. The MCTS conducted at the training time produced better diverse document rankings for training the model parameters.

5.3.2 Ranking with policy or with value? In "M^{2}Div(without MCTS)", we rely on the raw policy **p** for ranking the documents. In principle, the value function V can also be used for ranking, that is, at each state s a document (action) $\hat{a}$ is selected if $\hat{a} = \arg\max_{a \in \mathcal{A}(s)} V(T(s,a))$. We conducted experiments to show the performances of these two approaches and Figure 6 shows the test performance curves of the document rankings with the policy function and with the value function. From the results, we can see that the document rankings generated by the raw policy **p** are more stable and better than that of generated by the value function V. One possible reason for the phenomenon is that the ranking performance measure α-NDCG@5 are not easy to estimate accurately, especially at the early stages of the ranking procedure. This is why **p** in stead of V is adopted in "M^{2}Div(without MCTS)".

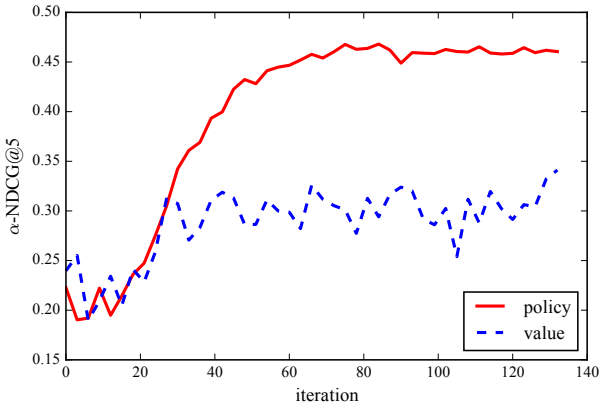

Figure 6: Performance curves of the document rankings generated with the policy function and with the value function, w.r.t. training iterations. The curves illustrates the averaged performances over all of the test queries.

6 CONCLUSION

In this paper we have proposed a novel approach to learning diverse ranking model for search result diversification, referred to as M^{2}Div. In contrast to existing methods that greedily select a locally optimal document at each of the ranking positions, M^{2}Div conducts an exploratory decision making with the lookahead MCTS and thus can select a better document. MDP is used to model the ranking process and reinforcement learning is utilized to train the model parameters. M^{2}Div offers several advantages: ranking with both the shared policy function and the value function, high accuracy in ranking, and flexibility of trading-off between accuracy and online ranking speed. Experimental results based on the TREC benchmark

datasets show that M^2Div can significantly outperform the state-of-the-art baselines including the heuristic methods of MMR, xQuAD and learning methods of R-LTR, PAMM, and MDP-DIV.

7 ACKNOWLEDGMENTS

This work was funded by the 973 Program of China under Grant No. 2014CB340401, the National Key R&D Program of China under Grants No. 2016QY02D0405, 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] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying Search Results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM '09)*. 5–14.

[2] Sumit Bhatia. 2011. Multidimensional Search Result Diversification: Diverse Search Results for Diverse Users. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. ACM, New York, NY, USA, 1331–1332.

[3] 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.

[4] Ben Carterette and Praveen Chandar. 2009. Probabilistic Models of Ranking Novel Documents for Faceted Topic Retrieval. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. 1287–1296.

[5] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected Reciprocal Rank for Graded Relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. 621–630.

[6] 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.

[7] 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.

[8] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* 12 (July 2011), 2121–2159.

[9] Sreenivas Gollapudi and Aneesh Sharma. 2009. An Axiomatic Approach for Result Diversification. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*. 381–390.

[10] Shengbo Guo and Scott Sanner. 2010. Probabilistic Latent Maximal Marginal Relevance. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. ACM, 833–834.

[11] Jiyin He, Vera Hollink, and Arjen de Vries. 2012. Combining Implicit and Explicit Topic Representations for 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, 851–860.

[12] 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.

[13] Sha Hu, Zhicheng Dou, Xiaojie Wang, Tetsuya Sakai, and Ji-Rong Wen. 2015. Search Result Diversification Based on Hierarchical Intents. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*. ACM, New York, NY, USA, 63–72.

[14] Zhengbao Jiang, Zhicheng Dou, Xin Zhao, Jian-Yun Nie, Ming Yue, and Ji-Rong Wen. 2018. Supervised Search Result Diversification via Subtopic Attention. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[15] Zhengbao Jiang, Ji-Rong Wen, Zhicheng Dou, Wayne Xin Zhao, Jian-Yun Nie, and Ming Yue. 2017. Learning to Diversify Search Results via Subtopic Attention. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 545–554.

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

[17] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. 1188–1196.

[18] Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. 2009. Enhancing Diversity, Coverage and Balance for Summarization Through Structure Learning. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*. ACM, New York, NY, USA, 71–80.

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

[20] 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.

[21] Lilyana Mihalkova and Raymond Mooney. 2009. Learning to Disambiguate Search Queries from Short Sessions. In *Machine Learning and Knowledge Discovery in Databases*. Lecture Notes in Computer Science, Vol. 5782. Springer.

[22] Filip Radlinski and Susan Dumais. 2006. Improving Personalized Web Search Using Result Diversification. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*. ACM, New York, NY, USA, 691–692.

[23] 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.

[24] 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.

[25] Rodrygo L. T. Santos, Jie Peng, Craig Macdonald, and Iadh Ounis. 2010. *Explicit Search Result Diversification through Sub-queries*.

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

[27] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[28] 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* 550, 7676 (2017), 354.

[29] Xiaojie Wang, Zhicheng Dou, Tetsuya Sakai, and Ji-Rong Wen. 2016. Evaluating Search Result Diversity Using Intent Hierarchies. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 415–424.

[30] 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 on Research and Development in Information Retrieval (SIGIR '17)*. 945–948.

[31] 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.

[32] 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.

[33] 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.

[34] Jun Xu, Long Xia, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Directly Optimize Diversity Evaluation Measures: A New Approach to Search Result Diversification. *ACM Trans. Intell. Syst. Technol.* 8, 3, Article 41 (Jan. 2017), 26 pages.

[35] Hai-Tao Yu, Adam Jatowt, Roi Blanco, Hideo Joho, Joemon Jose, Long Chen, and Fajie Yuan. 2017. A concise integer linear programming formulation for implicit search result diversification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 191–200.

[36] 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.

[37] 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.

[38] 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.

[39] 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.