Improving Session Search Performance with a Multi-MDP Model

Jia Chen*, Yiqun Liu, Cheng Luo, Jiaxin Mao, Min Zhang, and Shaoping Ma

Department of Computer Science and Technology, Institute for Artificial Intelligence,
Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing 100084, China
yiqunliu@tsinghua.edu.cn
www.thuir.cn

Abstract. To fulfill some sophisticated information needs in Web search, users may submit multiple queries in a search session. Session search aims to provide an optimized document rank by utilizing query log as well as user interaction behaviors within a search session. Although a number of solutions were proposed to solve the session search problem, most of these efforts simply assume that users' search intents stay unchanged during the search process. However, most complicated search tasks involve exploratory processes where users' intents evolve while interacting with search results. The evolving process leaves the static search intent assumption unreasonable and hurts the performance of document rank. To shed light on this research question, we propose a system with multiple agents which adjusts its framework by a self-adaption mechanism. In the framework, each agent models the document ranking as a Markov Decision Process (MDP) and updates its parameters by Reinforcement Learning algorithms. Experimental results on TREC Session Track datasets (2013 & 2014) show the effectiveness of the proposed framework.

Keywords: Session search · Markov Decision Process · Reinforcement Learning.

1 Introduction

Session search is an unfading topic in Information Retrieval (IR) research. It refers to the retrieval and reranking of documents for a search session with multiple queries. When a user tries to complete some complicated tasks, she may need information concerning various aspects. A single query can hardly satisfy such complex information need, so more queries might be submitted later. Research frameworks designed for single-round search processes may not be expert in dealing with these scenarios. To shed light on this research question, researchers in the IR community seek to optimize users' whole-session experiences with context information, e.g. "short-term" query history within a search session. This is the core idea of most existing studies which try to support session search.

* This work is supported by Natural Science Foundation of China (Grant No. 61622208, 61732008, 61532011) and National Key Basic Research Program (2015CB358700).

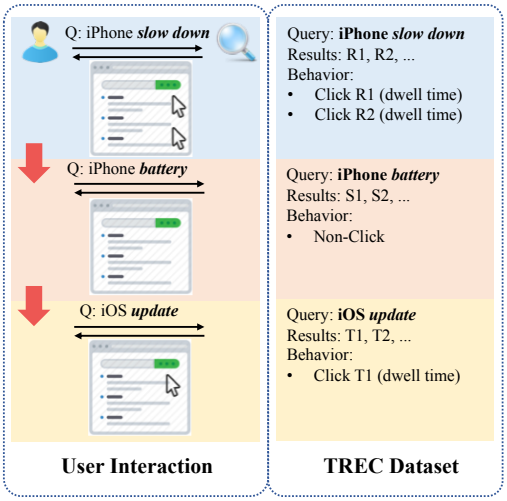

Fig. 1: An example search session which shows multi-round search interactions. (The left column illustrates the interaction process between a user and a search engine. The right column presents the information provided by TREC Session Track corpus, which is one of the most popular benchmarks in session search studies.)

Although this line of research has already gained significant success in supporting session search, we find that they may suffer from the "intent shift problem", i.e. when users are submitting queries and reading search results, their information needs keep evolving because the search can also be regarded as a learning process [7]. Figure 1 presents an example of a search session in which a user found that her iPhone became very slow and issued a query "iPhone slow down". She clicked several results and found that it may be caused by the battery problem. Then she submitted another query "iPhone battery" to the search engine and read a document which said that this issue might be solved by a future iOS update. At last, she checked the news about "iOS update". We can observe that during the example search process, the intent of the user actually shifted from "slow down" to "battery", then ended with "iOS update". This kind of "intent shift" was also observed by previous studies [19, 12]. Traditional session search approaches may fail to deal with this kind of intent shift since their optimization strategies usually implicitly assume that search sessions are focusing on a specific topic/intent.

In a search session, users' interaction processes are modeled as a *trial-and-error* process, which means that users are trying to optimize their search strategies by multi-round iterations. This process can also be regarded as a gaming process in which both users and search engines only know a portion of information. Reinforcement Learning (RL) algorithms can be naturally introduced to solve these problems. For instance, [13] proposes a system called "win-win" search model which regards the search process within a session as a Partially Observable Markov Decision Process (POMDP). It assumes that the users' decision states are determined by exploration and exploitation

processes of search engine users. Many existing studies share similar ideas, i.e. using the queries and results in previous rounds to improve the ranking performance of lately submitted queries.

However, most of the existing models maintain a single RL agent. The static system framework may hurt the performance of document ranking if the queries are rather dissimilar, or in other words focus on very different topics. To settle this problem, we propose a novel RL framework named Multiple Markov Decision Process (Multi-MDP) Model. The basic idea is: to improve the document rank of a certain query, we can not only learn from previously issued queries in the same session but also benefit from those semantically similar ones in other sessions. In our proposed framework, an MDP agent will be trained on a self-clustered query collection. Firstly each query will be distributed to the most appropriate cluster, then its candidate documents will be ranked by the MDP agent trained on this cluster. The system performance is validated on two popular session search benchmarks: TREC Session Track 2013 & 2014. Experimental results demonstrate that Multi-MDP model outperforms existing session search solutions on these datasets.

To summarize, the contributions of this paper are:

- We propose a novel framework based on reinforcement learning named Multi-MDP to improve performance in the context of session search.
- With Multi-MDP, we are able to capture users' evolving information needs by query clustering during the search process.
- Self-adaption techniques are employed to reduce the computational cost of multiple MDP rankers, which greatly improve the efficiency of Multi-MDP.

2 Related Work

2.1 Session search

Search engines are trying to support complex search tasks with useful history information. There has been much work on session search [3, 5, 8, 6, 4]. These approaches can be classified into *log-based* and *content-based* methods.

Considerable work involves studying queries and sessions through session log. [16] improved ranking performances by using query chains and click-through data from logs with a learning-to-rank framework. [18] put forward an optimal rare query suggestion framework by leveraging implicit feedbacks from users in the query logs. The framework also used a random walk model to optimize query correlation. In addition, [20] targeted the identification of *cross-session* tasks by learning inter-query dependencies from user behaviors in logs with a semi-supervised clustering model. Similarly, Kotov et al. proposed methods for modeling and analyzing user search behaviors that extend over multiple search sessions [10].

Less common in session search are content-based methods which directly study the content of queries and documents. Some methods are generated with the aid of Session Tracks at TREC [1, 2]. [5] greatly boosted the search accuracy by formulating structured queries using the nuggets generated from context information rather than

just using the current query. Furthermore, [17] optimized the whole-session relevance by exploring the intrinsic diversity of sessions.

What sets our work apart from previous studies is: rather than looking into the relationship between query and document, we put the emphasis on the semantic similarity between queries. We detect the users' information intents by query clustering and then optimize ranking performances with Reinforcement Learning.

2.2 Reinforcement Learning in IR

In these years, Reinforcement Learning (RL) has been a promising direction in Artificial Intelligence studies. In particular, session search can be modeled as a Markov Decision Process (MDP) or Partially Observable Markov Decision Process (POMDP) based on its *trail-and-error* specialty. Many relevant problems could be solved by a family of reinforcement learning algorithms. A number of MDP/POMDP models have been raised to optimize the system performance in session search.

[6] proposed a novel query change retrieval model, named as QCM, which modeled session search as an MDP process and utilized syntactic editing changes between query change and previously retrieved documents to enhance search accuracy. Zeng and Xu's work regarded the construction of a document ranking as a sequence of decision makings and each of which corresponded to an action of selecting a document for a specific position [21].

Some other researchers exerted POMDP to construct more complicated systems [11]. [22] modeled log-based document re-ranking as a POMDP which can complete the task effectively without document contents. Yuan and Wang's work addressed online advertisement recommendation by employing POMDPs. They also discussed how to utilize the correlation of ads to improve the efficiency of the exploration and increased ads incomes. The win-win search model proposed by [13] considered session search as a dual-agent stochastic game and designed four user decision states in its POMDP.

Our approach models the re-ranking of document lists as a Markov Decision Process (MDP). The essential difference between our model and the previous one is that our framework owns a self-adaption mechanism. Specifically, all of the aforementioned systems mainly focused on optimizing search results with a static user intent while ours starts from the semantic information embedded in queries. It provides users with better search experiences when their information needs are dynamically changing.

3 Multi-MDP Ranking Model Framework

3.1 Main Framework

We introduce a Multi-MDP ranking model to improve the ranking performance in session search. The model owns multiple agents and maintains each agent by training queries with similar semantic features with reinforcement learning algorithms.

To facilitate the presentation of our work, we first briefly introduce the kernel modules of the model. Figure 2 outlines two core parts of the Multi-MDP Ranking Model.

Part I is an automatic clustering module distributing training queries into clusters according to their semantic features. Here we detect the evolving intents of search

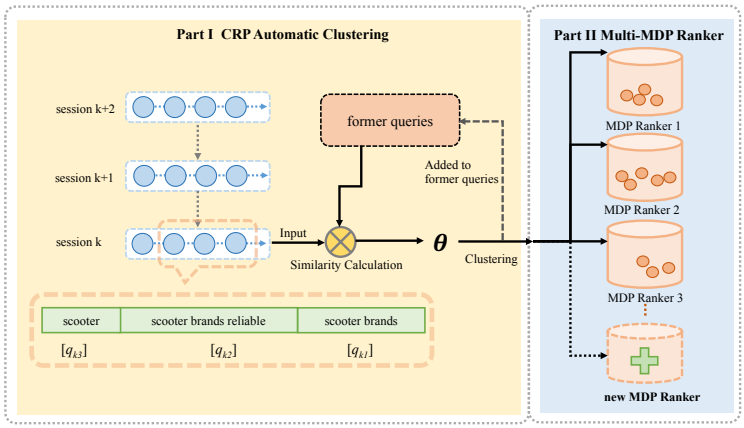

Fig. 2: The Framework of Multi-MDP Ranking Model

users by clustering a sequence of queries on the semantic level. For a specific query, it might be aggregated to an existing cluster or a newly created one. In this process, a hyper-parameter θ is introduced as the threshold of the minimum similarity between two queries within the same agent. We calculate the value of the θ by implementing a greedy search method on the training set. More details of this method will be described in Section 3.2. Then we model query clustering as a **Chinese Restaurant Process (CRP)**. In probability theory, the Chinese Restaurant Process provides a generative description for the Dirichlet Process [15]. It is a discrete-time stochastic process analogous to seating customers at tables in a Chinese restaurant. Note that CRP is introduced in the prevention of explosive growth in the number of ranking clusters in part II.

Part II is a series of ranking agents, each of which models document ranking as an MDP and trains parameters by REINFORCE, a widely-used Policy Gradient algorithm in reinforcement learning. As described in part I, the increase of the agent totality will be under the control of CRP.

System parameters are learned in the training phase. For a query in the test set, the system will choose the most appropriate cluster for it(even the cluster does exist yet and should be created). Then the query will be processed by the MDP ranker which is trained on that cluster. Note that the proposed framework can dynamically detect the instant information needs of users in multiple search rounds (Part I) as well as optimize the document list returned for a specific query at each rank(Part II).

3.2 Chinese Restaurant Process Based Query Clustering

We model the clustering process as a Chinese Restaurant Process (CRP). The reason we choose CRP rather than some other classical clustering methods is that it can dynamically control the total number of clusters with a hyper-parameter ξ. For a new query, the system will calculate the similarity between it and existing clusters first. If the maximal similarity is lower than a pre-set threshold, then the query should not be owned by any existing clusters. So it will be distributed to a newly created one with a probability

determined by CRP. This mechanism efficaciously restrains the total number of rankers within an upper bound and avoids a huge occupation of system space. For a query Q_i (analogous to a customer) and a query cluster (compared to a table), the probability distribution of seating one customer at a table is as follows:

$$Pr(Q_i = r|Q_1, ..., Q_{i-1}) = \begin{cases} \frac{\xi+|B|\alpha}{\xi+i-1} & if\ r\ \in\ new\ ranker, \\ \frac{|b|-\alpha}{\xi+i-1} & if\ r = b, b \in B; \end{cases} \tag{1}$$

where $|B|$ denotes the total number of existing clusters, b is one of the non-empty clusters and $|b|$ denotes the capacity of it. Specifically, we only need to control the probability of creating a new cluster so only the first formula will be utilized. Thus there is no need to discuss the value of $|b|$ here. If we let α = 0, then a single parameter ξ needs to be determined. This can be solved by the expected number of rankers $E(R)$ as shown in Equation 2:

$$E(R) = \sum_{k=1}^{n} \frac{\xi}{\xi + k - 1} = \xi \cdot (\Psi(\xi + n) - \Psi(\xi)), \tag{2}$$

In this equation, *n* refers to the number of queries. ξ is the hyper-parameter while $\Psi(\xi)$ denotes the *digamma* function. To hold system robustness, we set the expected number of rankers $E(R)$ as twice of topic scopes in training set (this can be adjusted according to system condition). By solving the equation, the value ξ can be determined and will be introduced as a parameter in query clustering. In this way, the total number of rankers will converge to an invariant value rather than increase endlessly.

Given that query clustering has a direct impact on the system performance, we should try to find a potentially reasonable threshold, e.g. θ, which can discriminate a query from other dissimilar ones. If the similarity of the two queries is higher than the threshold then we should aggregate them in the same cluster. One of the options is to estimate θ based on the topic labels provided by TREC Session Track training set. A greedy algorithm searching for the value θ follows two heuristic rules:

- if the maximum similarity of the current query q and a previous query $\hat{q}$ is greater than the threshold, then $label_q$ = $label_{\hat{q}}$;
- if the maximum similarity of the current query q and a previous query $\hat{q}$ is lower than the threshold, then $label_q$ is assigned to a new label.

In the training phase, the clustering process is performed as mentioned above. However in testing phase, a query is always distributed to the cluster that is the most similar to it. By comparing the coherence with ground truth labels for various searching values of θ, we choose the one of highest coherence. The ground truth labels are analyzed from the log provided by TREC which classifies each query to a corresponding topic.

3.3 MDP Ranking Agent

Having clustered each query into a specific cluster, the system should optimize document rankings for queries within a cluster. The document ranking process for each

query is formalized as a Markov Decision Process (MDP) [21], where document ranking is considered as a sequence of decision makings, each of which can be defined as selecting one appropriate document at a rank in vertical position. As shown in Figure 3, the proposed MDP ranking process is represented as a tuple $\{S, A, T, R, \pi\}$, respectively denoting system states, actions, transition function, reward and the decision policy. Detailed expositions are described as:

States *S* is a set of different states that describe the system environment. To arrange a rank for a document set, the agent should make a sequence of decisions. One decision can be made according to the position and the candidate document set the agent can choose from. Thus, the state at the time *t* is designed as [*t,* X_t], where X_t represents the remaining candidate document set.

Actions *A* refers to a set of actions that the agent could take at each step. In our model, the agent should decide which document to be selected at the current position. The action set for the state s_t is denoted by $A(s_t)$. At the time *t*, $a_t \in A(s_t)$ selects a document d_i at the ranking position *t* + 1, where *i* is the index of the chosen document.

Transition Function *T(S, A)* is a function that maps a state s_t into a new state s_{t+1} in response to the action at time *t*. When the agent chooses a document d_i from the candidate set and ranks it at the *t*-th position, the document is removed from the candidate set in case of duplication. Equation 3 is the formulation of the transition function.

$$s_{t+1} = T([t, X_t], a_t) = [t + 1, X_t \backslash d_i], \tag{3}$$

Reward *R(S, A)* is the global reward of a rank, which directly represents the gains of users. In this paper, the reward is regarded as a sum of discounted gain at each position from 1 to *N*. It is defined in a manner similar to DCG (a widely-used metric) to directly optimize the evaluation measures and leverages the performances at each rank. Unable to estimate accurate user gains from the insufficient interaction data, we design a reward function in direct promotion of IR metrics.:

$$R(s_t, a_t) = \begin{cases} 2^{L(d_i)} - 1 \; if \; t \; = \; 0, \\ \frac{2^{L(d_i)}-1}{log_2(t+1)} \; if \; t \; \neq \; 0; \end{cases} \tag{4}$$

where $L(d_i)$ is the relevance label of the selected document d_i. The reward at each position will be accumulated at the end to represent the global reward of a rank, which provides references for an agent to update parameters through RL algorithms.

Policy $\pi(a|s) : A \times S \rightarrow (0, 1)$ is a probabilisitic distribution over the possible actions for an agent at the state s_t. Generally, the policy of one agent represents the probabilities of selecting one specific document for a position. This is formalized as Equation 5:

$$\pi(a_t|s_t; w) = softmax(w^T V(d_i)), \tag{5}$$

where $V(d_i)$ denotes the vector representation of d_i, $w \in \mathbb{R}^k$ represents the parameters of each ranking agent with the same dimension *k* as the query/document representations.

Given a training query q, a set of h candidate documents D, the relevance labels for each document L, the initial system state can be denoted as $s_0 = [0, X]$. At each time $t = 0, ..., h - 1$, the agent observes the environment and determines the belief of its state s_t. According to the state and the system policy π, an agent carries out an

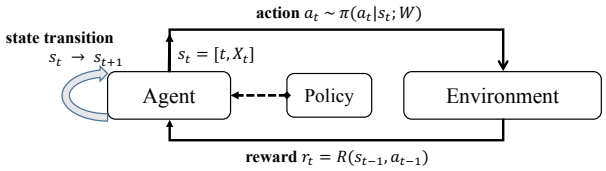

Fig. 3: State Transition in Each MDP Ranking Agent

action a_t and ranks a document at a position. The agent will immediately receive a reward $r_t = R(s_t, a_t)$ on the basis of the relevance label. Then the system transfers to the next state s_{t+1} and tries to take next action a_{t+1}. The process is repeated until all the h documents are selected. We adopt the Monte Carlo Sampling method to generate several ranks and calculate the reward of these ranks. Of all the rank with the highest reward will be recorded for system updates.

In the testing phase, no label is available. An agent will generate a rank according to its learned policy. For each position, the agent will choose a document with maximal probability. The probability is calculated by the product of w and document representation. In order to learn the model parameters w, we use REINFORCE algorithm to update them. According to the reward defined above, the long-term gain of a sampled ranking list μ can be defined as:

$$G(\mu) = \sum_{k=1}^{M} \gamma^{k-1} r_k, \tag{6}$$

where μ is the ranking list sampled from a stochastic ranking process, on the basis of Monte Carlo sampling. Note that if γ = 1, the formula is equivalent to the definition of DCG. According to REINFORCE algorithm, the gradient $\nabla_w J(w)$ can be calculated as follows:

$$\nabla_w J(w) = \gamma^t G_t \nabla_w log \pi_w(a_t|s_t; w), \tag{7}$$

where $J(w)$ is the expected long-term return of the document ranking for a query. Our goal is to optimize this value, enabling the sampled rank to approximate the ideal rank. By comparing the reward received by the agent from different ranks, the agent chooses the one with the highest reward and stores the rank with its reward as an episode into its memory. Every time after a batch of training queries has been processed, the agent updates its parameters by the episodes stored in the memory through equation 7 and clears the memory then.

4 Experiments

4.1 Dataset

To validate the effectiveness of our proposed Multi-MDP framework, we conducted our experiments on TREC Session Tracks 2013-2014 datasets [1, 2]. TREC Session Tracks were one of the core tasks of Text REtrieval Conference[1].

[1] `http://trec.nist.gov`

Table 1: Dataset Statistics for TREC Session Track 2013 and 2014

Data	2013	2014
#topic number	61	60
#session number	133	1,257
#query number	453	3,213
#unduplicated documents	26,664	107,564

As shown in Figure 1, in Session Tracks, the participants were given a collection of search sessions. For each session, the participants were provided with real users' interactions in multiple search rounds except for the last one. The goal is to optimize document rank for the last query with the context information in previous search rounds.

Table 1 provides an overview of the two datasets. For each query, we used the top 100 documents through batch query service as the candidate set on the ClueWeb12 B13 dataset. In addition, queries whose number of documents returned by batch service are less than 100 were filtered. All duplicated documents were removed as well.

4.2 Query Clustering

The clustering process is based on semantic features. For a pair $< q_1, q_2 >$, the similarity can be calculated as follows:

1. The textual similarity of q_1 and q_2 (QT);
2. The proportion of overlapped documents in the top 10 retrieved documents of q_1 and q_2 (DO);
3. The average textual similarity of top-10 retrieved documents of q_1 and q_2 (DT);

These three measurements are respectively denoted as *QT*, *DO* and *DT* in the remaining parts of this paper. Note that for *QT* and *DT*, textual similarity refers to the cosine similarity of two semantic representations. *DT* can be formulated as:

$$sim^{DT}(q_1, q_2) = \frac{1}{100} \sum_{i=1}^{10} \sum_{j=1}^{10} cos(V_{d_{1i}}, V_{d_{2j}}) \tag{8}$$

where d_{1i} refers to the document at the i-th position of q_1 and d_{2j} refers to the document at the j-th position of q_2. V denotes vector representation.

To better capture the implicit information intents embedded in queries, the similarity is calculated based on the *distributed representation* of textual contents, i.e. Another option here is to use the *one-hot* representation of tokens. We choose distributed presentations since they are more likely to capture the semantic level similarity while one-hot representations tend to capture exact matchings of textual units.

According to the three definitions of query similarity and two methods of generating document vector representations (this will be described in Section 4.3), five combinatorial methods can be applied in the traversal process (the result of top 10 document overlapping ratio is equivalent for max-pooling or average-pooling representations, hence a repeated method is removed). The consistency was measured by Cohen's Kappa. Table

Table 2: Best Threshold of Each Method.

Method	Best θ	Kappa	Method	Best θ	Kappa
MAX_QT (2013)	0.95	0.454	MAX_QT (2014)	0.96	0.052
AVG_QT (2013)	0.75	0.335	AVG_QT (2014)	0.77	0.054
DO (2013)	0.60	0.497	DO (2014)	0.90	0.040
MAX_DT (2013)	0.99	0.289	MAX_DT (2014)	0.99	0.032
AVG_DT (2013)	0.99	0.428	AVG_DT (2014)	0.99	0.044

2 shows the best thresholds θ found for each method/dataset. Note that the scale of 2014 dataset is much larger, the clustering consistency results are much lower compared to those of the 2013 dataset. Generally, the similarity between MAX representations is higher so we can also find a relatively higher value of θ in MAX representations.

4.3 Evaluation Settings

We followed the Session Tracks' settings and used the first $N-1$ queries in each session for training. The Test set is consist of the last queries. For both Session Track 2013 & 2014 dataset, the ratio of the training set and test set approximates 4:1. Queries and documents actually contain multiple words. We tried two methods of generating the query-level/document-level representations: max-pooling (MAX) and average-pooling (AVG) of word vectors. All web pages in Session Track 2013 & 2014 dataset were collected as the corpus. The detailed process is described as follows: We first fed the corpus to a word representation model, named as *Global Vectors for Word Representation* (GloVe) [14], to get each word's representation. Long-tailed words were removed and only those with top 100K frequency remain.

To validate the system performance, we compared our model with several baseline systems listed as following:

- *Win-Win*: A framework which models the session search process as Partially Observable Markov Decision Process (POMDP) [13];
- *Lemur*: We submitted each query to Lemur Indri Search Engine[2] through online batch service provided by ClueWeb12 and calculate evaluation metrics of the document lists returned from it;
- *TREC median*: The median TREC system reported by TREC Session Track;
- *TREC best*: The winning run in TREC Session Tracks;
- *QCM SAT*: A variation of QCM proposed by [6];
- *Rocchio*: A session search model which is proposed by [9].

We use nDCG and MAP to evaluate different runs. For nDCG, we choose several different cutoffs to check the performance of our model at different ranks.

[2] `http://lemurproject.org/clueweb12/`

Table 3: Search Performance on TREC Session Tracks 2013. (The best Multi-MDP model, i.e. Multi-MDP^{MAX_QT}, is significantly better than the strongest baseline Win-Win at $p < 0.001$ using a paired t-test.)

System	MAP	nDCG@1	nDCG@3	nDCG@6	nDCG@10
Win-Win	0.1290	–	–	–	0.2026
Lemur	0.1072	0.0592	0.0775	0.0953	0.1123
TREC best	–	–	–	–	0.1952
TREC median	–	–	–	–	0.1521
QCM SAT	0.1186	–	–	–	0.0939
Rocchio	0.1320	–	–	–	0.1060
Multi-MDP^{MAX_QT}	**0.2190**	**0.2561**	**0.2638**	**0.2453**	**0.2556**
Multi-MDP^{AVG_QT}	0.1891	0.2073	0.1956	0.2089	0.2182
Multi-MDP^{MAX_DO}	0.2138	0.2561	0.2339	0.2193	0.2301
Multi-MDP^{AVG_DO}	0.1342	0.1138	0.1171	0.1333	0.1424
Multi-MDP^{MAX_DT}	0.1488	0.1638	0.1627	0.1571	0.1618
Multi-MDP^{AVG_DT}	0.1306	0.1156	0.1075	0.1170	0.1409

Table 4: Search Performance on TREC Session Tracks 2014. (The best Multi-MDP model, i.e. Multi-MDP^{MAX_QT}, is significantly better than the baseline Lemur at $p < 0.001$ using a paired t-test.)

System	MAP	nDCG@1	nDCG@3	nDCG@6	nDCG@10
Lemur	0.1260	0.0916	0.1009	0.1139	0.1251
TREC best	–	–	–	–	0.2099
TREC median	–	–	–	–	0.1170
Multi-MDP^{MAX_QT}	**0.1857**	**0.2271**	**0.2169**	**0.2118**	**0.2107**
Multi-MDP^{AVG_QT}	0.1315	0.1194	0.1063	0.1211	0.1361
Multi-MDP^{MAX_DO}	0.1850	0.2047	0.2036	0.2000	0.1995
Multi-MDP^{AVG_DO}	0.1516	0.1573	0.1630	0.1649	0.1751
Multi-MDP^{MAX_DT}	0.1293	0.1358	0.1388	0.1384	0.1412
Multi-MDP^{AVG_DT}	0.1586	0.1413	0.1544	0.1672	0.1779

4.4 Experimental results

For both TREC 2013 and 2014 dataset, we tried our Multi-MDP Ranking Model with six types, where *QT*, *DO* and *DT* denote three definitions of query similarity and "MAX"/"AVG" refers to the generating methods for query/document presentation.

Table 3 shows the search performance of all systems in TREC Session Tracks 2013. Since our experimental settings are exactly the same as [13], we cite the results they reported for comparison. The result shows that our model significantly outperforms other session search systems. We can see that Multi-MDP^{MAX_QT} is the best among all methods. The nDCG metrics of Multi-MDP models are significantly higher than those of Lemur Search Engine at all ranks. The results validate that our model can better detect evolving intents of users and improve search performance in both whole-session level (high metric value on average) and query level (high performance at each rank).

Table 4 shows the search performance of all systems for TREC Session Tracks 2014. On the 2014 dataset, we can only get the results of TREC Best and TREC Median [2]. Although the dataset scale is much larger, the experimental results show that the Multi-MDP Ranking Model still performs well. Of all the methods, Multi-MDP^{MAX_QT} still owns the best performance, although the results are not statistically significant compared to the TREC Best result.

An interesting finding is that the systems with *QT* usually outperform other systems. It suggests that the query contents are more representative than the document lists or document contents. Query contents suffer from fewer noises than documents thus could express refined user intents which may improve clustering accuracy. Another finding is that the systems with *DO* perform better for a dataset on a larger scale. This might be caused by the smaller threshold of small dataset which owns lower discrimination between two documents.

5 Conclusion

This paper presents a novel session search framework: Multi-MDP Model, which models the document reranking for each query as a Markov Decision Process (MDP) and owns multiple agents. Our model gives a new guide of constructing a self-adaptive framework to fit with the size of the dataset and dynamically detect the instant information intents of search users. The increase of agents is under the control of the Chinese Restaurant Process (CRP) to ensure a low space occupation. In each agent, an MDP-based ranking model is trained for better document ranking performances. The experiments on TREC Session Tracks 2013 and 2014 datasets show that our work is both effective and efficient for session search and outperforms the best systems in TREC.

Our work is content-based and considers the explicit feedbacks of users as the global reward for a rank. We emphasize two main stages: query clustering and model training. Some other factors like click-through data, dwell time from session logs have not been utilized in the system yet. We may have an attempt to make better use of this information to improve the system performance in future.

References

1. Carterette, B., Kanoulas, E., Hall, M., Bah, A., Clough, P.: Overview of the trec 2013 session track. Tech. rep., DELAWARE UNIV NEWARK DEPT OF COMPUTER AND INFORMATION SCIENCES (2013)
2. Carterette, B., Kanoulas, E., Hall, M., Clough, P.: Overview of the trec 2014 session track. Tech. rep., DELAWARE UNIV NEWARK DEPT OF COMPUTER AND INFORMATION SCIENCES (2014)
3. Carterette, B., Kanoulas, E., Yilmaz, E.: Simulating simple user behavior for system effectiveness evaluation. In: Proceedings of the 20th ACM international conference on Information and knowledge management. pp. 611–620. ACM (2011)
4. Feild, H., Allan, J.: Task-aware query recommendation. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. pp. 83–92. ACM (2013)
5. Guan, D., Yang, H., Goharian, N.: Effective structured query formulation for session search. Tech. rep., GEORGETOWN UNIV WASHINGTON DC DEPT OF COMPUTER SCIENCE (2012)
6. Guan, D., Zhang, S., Yang, H.: Utilizing query change for session search. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. pp. 453–462. ACM (2013)

7. Gwizdka, J., Hansen, P., Hauff, C., He, J., Kando, N.: Search as learning (sal) workshop 2016. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. pp. 1249–1250. ACM (2016)
8. Jiang, J., Han, S., Wu, J., He, D.: Pitt at trec 2011 session track. In: TREC (2011)
9. Joachims, T.: A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Tech. rep., Carnegie-mellon univ pittsburgh pa dept of computer science (1996)
10. Kotov, A., Bennett, P.N., White, R.W., Dumais, S.T., Teevan, J.: Modeling and analysis of cross-session search tasks. In: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. pp. 5–14. ACM (2011)
11. Littman, M.L., Kaelbling, L.P., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. In: ARTIFICIAL INTELLIGENCE. pp. 99–134 (1995)
12. Liu, J., Belkin, N.J.: Personalizing information retrieval for multi-session tasks: The roles of task stage and task type. In: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. pp. 26–33. ACM (2010)
13. Luo, J., Zhang, S., Yang, H.: 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. pp. 587–596. ACM (2014)
14. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
15. Pitman, J., et al.: Combinatorial stochastic processes. Tech. rep., Technical Report 621, Dept. Statistics, UC Berkeley, 2002. Lecture notes for St. Flour course (2002)
16. Radlinski, Filip, Joachims, Thorsten: Query chains: learning to rank from implicit feedback pp. 239–248 (2006)
17. Raman, K., Bennett, P.N., Collins-Thompson, K.: Toward whole-session relevance: exploring intrinsic diversity in web search. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. pp. 463–472. ACM (2013)
18. Song, Y., He, L.W.: Optimal rare query suggestion with implicit user feedback. In: International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, Usa, April. pp. 901–910 (2010)
19. Teevan, J., Dumais, S.T., Liebling, D.J.: To personalize or not to personalize: modeling queries with variation in user intent. In: International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 163–170 (2008)
20. Wang, H., Song, Y., Chang, M.W., He, X., White, R.W., Chu, W.: Learning to extract cross-session search tasks. In: Proceedings of the 22nd international conference on World Wide Web. pp. 1353–1364. ACM (2013)
21. Wei, Z., Xu, J., Lan, Y., Guo, J., Cheng, X.: Reinforcement learning to rank with markov decision process (2017)
22. Zhang, S., Luo, J., Yang, H.: A pomdp model for content-free document re-ranking pp. 1139–1142 (2014)