Designing States, Actions, and Rewards for Using POMDP in Session Search

Jiyun Luo, Sicong Zhang, Xuchu Dong, and Hui Yang

Department of Computer Science, Georgetown University
37th and O Street NW, Washington DC, 20057, USA
jl1749,sz303,xd47@georgetown.edu,huiyang@cs.georgetown.edu

Abstract. Session search is an information retrieval task that involves a sequence of queries for a complex information need. It is characterized by rich user-system interactions and temporal dependency between queries and between consecutive user behaviors. Recent efforts have been made in modeling session search using the Partially Observable Markov Decision Process (POMDP). To best utilize the POMDP model, it is crucial to find suitable definitions for its fundamental elements – *States*, *Actions* and *Rewards*. This paper investigates the best ways to design the states, actions, and rewards within a POMDP framework. We lay out available design options of these major components based on a variety of related work and experiment on combinations of these options over the TREC 2012 & 2013 Session datasets. We report our findings based on two evaluation aspects, retrieval accuracy and efficiency, and recommend practical design choices for using POMDP in session search.

Keywords: Session Search, POMDP, State, Action, Reward.

1 Introduction

Information Retrieval (IR) tasks are concerned with finding relevant documents to fulfill user's information needs. Session search, as defined in the TREC (Text REtrieval Conference) Session tracks is an information retrieval task that involves multiple queries and multiple search iterations to achieve a complex information need [11, 12]. In a session, a user keeps formulating queries until he or she gets satisfied with the information need [12], bored, or frustrated [2]. Session search is a challenging research area that is characterized by rich user-system interactions, complex information needs, and temporal dependency between queries and between user behaviors.

In a session, a user interacts with the search engine to explore the information space: the user continuously reformulates queries, clicks on documents, and examines documents. This is a trial-and-error setting. Classic ad-hoc retrieval models emphasize on handling one-shot query and treating each queries in a session independently [16]. Classic relevance feedback models, such as Rocchio [9], although modeling feedbacks from the user, also treat each query in a session independently: the user feedbacks are for a particular query. The continuity

of queries in a sequence during a session has not yet been studied much. This places unique challenge on session search for new statistical retrieval models that is able to handle the dynamics present in the task.

The family of Reinforcement Learning (RL) algorithms [6] matches well with the trial-and-error setting present in session search: the algorithm learns from repeated, varied attempts which are continued until success. The learner (also known as agent) learns from its dynamic interactions with the world, rather than from a labeled dataset as in supervised learning. In such a setting, a stochastic model assumes that the system's current state depend on the previous state and action in a non-deterministic manner [15]. Among various models in the RL family, Partially Observable Markov Decision Processes (POMDP) [19] has been applied recently on IR problems including session search [14], document re-ranking [8, 22], and advertisement bidding [20]. In a POMDP, hidden information can be modeled as hidden states, while visible signals in the process can be modeled as observations or actions.

States, actions, and *reward functions* are the fundamental elements in a POMDP framework. The following principles are usually referred to when defining these elements in a POMDP framework:

- *States*: What changes with each time step?
- *Actions*: How does our system change the state?
- *Rewards*: How can we measure feedback or effectiveness?

Given the recent work on applying POMDP to session search, what is missing is a study that evaluates the design for *States, Actions,* and *Rewards*. In this paper, we strive to answer the research question – *what are the best design options to model session search using POMDP*. We use search effectiveness and search efficiency as two evaluation aspects to help select the best design under different circumstances.

However, there are only a few existing approaches that use POMDP to study IR problems. We hence expand the targeted group of approaches to a wider range of methods, including MDP [5], exploratory online learning [6] and decision theories [18], to study how they define the three major components for session search. Therefore, not all methods studied in this paper are based on POMDP, but they all share the idea of using states, actions, and rewards. We would like to find out the promising designs of those elements for our task.

In the remainder of this paper, after briefly presenting the POMDP framework (Section 2), we lay out available options for *states, actions,* and *rewards* in using POMDP for session search (Section 3). We then experiment on combinations of various options over the TREC Session 2012 & 2013 datasets [11, 12] and report our findings on the impacts of various settings in terms of search accuracy and efficiency (Section 4). Finally, we recommend design choices for using POMDP in session search (Section 5) and conclude the paper (Section 6).

2 Using a POMDP framework

Partially Observable Markov Decision Process (POMDP) can be represented as a tuple of $(S, M, A, R, \gamma, O, \Theta, B)$, , which consists of states S, state transition function M, actions A, reward function R, discount factor γ (usually between 0 and 1), observations O, observation function Θ, and belief states B. In a POMDP model, the states are hidden from the agent. The agent can only observe symbols (observations) emitted according to hidden states. At the same time, the agent forms its beliefs on the hidden states, which is an estimated probability distribution over the state space. Once the agent obtains a new observation, its belief will be updated accordingly. A detailed version of using POMDP in session search can be found in [14].

The goal of a POMDP is to find an optimal policy which maximizes the expected reward value, also known as the value function. Let $R(b, a)$ be the reward for an action a based on the current belief b. The value function can be expressed by the Bellman equation [1, 10].

$$V^*(b) = \max_a \left[R(b, a) + \gamma \sum_{o'} P(o'|a, b) V^*(b') \right] \tag{1}$$

The notation $P(o'|a, b)$ represents the probability of observing o after taking action a with belief b. Let $b(s)$ denote the belief on being in state s. The new belief b' on the next state is calculated by updating b as follows:

$$b'(s') = \eta \times \Theta(o|s', a) \sum_{s \in S} P(s'|s, a) b(s) \tag{2}$$

In Eq. 2, probability function P is the transition function, and the notation $\Theta(o|s', a)$ stands for the probability to observe o given state s' and action a. Here, we use η as the normalizing constant.

There are standard algorithms, including QMDP and MC-POMDP, to solve problems formalized by POMDPs [10]. Value iteration is used in QMDP by treating the value function as a mapping from beliefs to real numbers. MC-POMDP algorithm is applicable to continuous POMDP problems. However, many approaches can only be applied to problems of very small scales. Littman et al's Witness Algorithm is a more practical approach to obtain solutions to POMDP problems [13].

Solutions to the POMDP framework for session search can be obtained by using these approaches. Our aim in this paper is not how to get a solution. When applying POMDP to session search, the definitions of the states, actions, and rewards are flexible but critical to search accuracy and efficiency. In the following sections, we focus on studying the design choices of these elements.

3 Design Choices: States, Actions, and Rewards

In this section, we summarize the existing research work to enumerate the available design choices for a POMDP model in the context of session search. These

choices are discussed in three categories: states, actions and rewards. Some of the existing work mentioned in this section are not based on POMDP. However, they all share the idea of using states, actions, and rewards. Hence they are still valuable to our study.

3.1 States

State definition is essential in modeling session search by a POMDP. As we can see, related research in similar tasks have proposed a variety of state definitions. They include *queries* [5, 6], *document relevance* [8, 22], and *relevance vs. exploration decision making states* [14]. We group them into two design options:

(S1) Fixed number of states. Using a predefined fixed number of states can easily characterize certain properties of the session based on the current state. For instance, Zhang et al. used two binary relevance states, "Relevant" and "Irrelevant" to represent the decision-making states that the user considers the previously returned documents are relevant or not [22]. A more complete formulation of the decision-making states was presented in Luo et al. [14], where a cross-product of two decision-making dimensions – "whether the previously retrieved documents are relevant" and "whether the user desires to explore" – forms four hidden states which reflect the current status of the search process.

(S2) Varying number of states. Some approaches choose to model session search using a varying or even infinite number of states. A popular approach is to model queries in a session as states (Hofmann et al. [6] and Guan et al. [5]). In this design, the number of states changes according to session length, i.e., the number of queries in a session. There are also abstract definitions of states. For instance, Jin et al. used relevance score distribution as the states[8], which leads to an infinite number of real valued states.

As discussed above, all state definitions are used to characterize the current status of the search process. Using fixed number of states tends to reflect more specific features while using varying number of states may have more abstract characterization of the search process. Hence, we would like to point out that *state definition is an art*, which depends on the needs of the actual IR task.

3.2 Actions

It is worth noting that, as Luo et al. [14] pointed out, the user and the search engine are two autonomous agents in a session. For session search, typical user actions include: *Add query terms; Remove query terms; Keep query terms; Click on documents;* and *SAT click on documents* (click and read the documents for a long period of time). Typical search engine actions include: *increase/decrease/keep term weights; switch on or switch off query expansion; adjust the number of top documents used in Pseudo Relevance Feedback (PRF)* and *consider the ranked list itself as actions*. Here we focus on the search engine actions. Existing search engine actions in related work are grouped into:

(A1) Technology Selection. Some approaches use a meta-level modeling of actions. They don't focus on details in a single search method but on implementing multiple search methods (termed as *search technologies*), and selecting the best search technology to use. An action using technology selection can be *switching on or switching off the technology*, or *adjusting parameters in the technology*. Example technologies include query expansion and pseudo relevance feedback (PRF). To illustrate, Luo et al. selected the number of top retrieved documents to be included in PRF [14].

(A2) Term Weight Adjustment. Another idea to model search engine actions focuses on term weight adjustments. This group of actions enables the search engine to directly adjust individual terms' weights. Typical weighting schemes include *increasing term weights, decreasing term weights*, or *keeping term weights unchanged*. Guan et al. proposed four types of term weighting scheme (*theme terms, novel added terms, previously-retrieved added terms*, and *removed terms*) as actions according to the query changes detected between adjacent search iterations [5].

(A3) Portfolio A more straightforward type of search engine actions is using the document lists. We follow the naming used in [8] and call this type of actions *portfolio*. Here a ranked list of documents is a *portfolio* and is treated as a single action. The space of the document permutation is the action space, where each document ranking permutation is a different action.

These actions are in fact what a search engine can do for document retrieval. Hence, we say that *actions are essentially options in your search algorithm.*

3.3 Rewards

A clear goal is key to any success. In order to estimate the benefits from an action, we need to evaluate the reward R of taking the action at state s. Similar to the loss (risk) function in supervised learning, a reward function can guide the search engine throughout the entire dynamic process of session search. Since session search is a document retrieval task, it's natural that *the reward function is about document relevance*. Notably, the difference between session search and one-shot query search lies in that session search aims to optimize a *long term reward*, which is an expectation over the overall rewards in the whole session, while one-shot query search doesn't have to do that. We group reward functions in related work into:

(R1) Explicit Feedback. Rewards directly generated from user's relevance assessments are considered as explicit feedback. Both Jin et al. [8] and Luo et al. [14] calculated the rewards using nDCG [7], which measures the document relevance for an entire ranked list of documents with ground truth judgments.

(R2) Implicit Feedback. Other approaches used implicit feedback obtained from user behavior as rewards. For instance, Hofmann et al. used user click information as the reward function in their online ranking algorithm [6] and Zhang et al. used clicks and dwell time as reward for document re-ranking [22].

4 Experiments

In this section, we aim to examine the design choices for using POMDP in session search. As we lay out in the previous section, there are two options for states, three for actions, and two for rewards, which result in a total of $2\times3\times2 = 12$ combinations. For example, the search system proposed by [14] used a combination of $S_1A_1R_1$, which means "Fixed number of states", "Technology Selection" as the actions, and "Explicit Feedback" as the reward. We report our findings on the search accuracy and search efficiency for those design options.

4.1 Task and Datasets

We evaluate a number of systems, each of which represents a combination of design choices as mentioned in Section 3. The session search task is the same as in the recent TREC 2012 and 2013 Session Tracks [11, 12]: to retrieve 2000 relevant documents for the last query in a session. Session logs, including queries, retrieved URLs, Web page titles, snippets, clicks, and dwell time, were generated by the following process. Search topics were provided to the user. The user was then asked to create queries and perform search using a standard search engine provided by TREC. TREC 2012 contains 297 queries in 98 sessions, while TREC 2013 contains 442 queries in 87 sessions. An example search topic is "You just learned about the existence of long-term care insurance. You want to know about it: costs/premiums, companies that offer it, types of policies, ..." (TREC 2013 Session 6).

We use the evaluation scripts and ground truth provided by TREC for evaluation. The metrics are mainly about search accuracy, including nDCG@10, nERR@10, nDCG, and MAP [12]. We also report the retrieval efficiency in Wall Clock Time, CPU cycles and the Big O notation. The dataset used for TREC 2012 is ClueWeb09 CatB, containing 50 million English Web pages crawled in 2009. The dataset used for TREC 2013 is ClueWeb12 CatB, containing 50 million English Web pages crawled in 2012. Spam documents are removed according to the Waterloo spam scores [3]. Duplicated documents are also removed.

4.2 Systems

Among the 12 combinations mentioned in Section 3, $S_1A_2R_2$, $S_1A_3R_1$, $S_2A_1R_2$, $S_2A_2R_2$ and $S_2A_3R_2$ are not discussed in this paper because we have not yet found a realistic way to implement them. We evaluate the remaining seven choices. For $S_2A_1R_1$, we implement two versions of it. The first is UCAIR, a re-implementation of Shen et al.'s work [18]. However, this system has only one action. To have a fair comparison with other systems, we create another $S_2A_1R_1$ system to include more actions. In total, we implement eight POMDP systems:

$\mathbf{S_1A_1R_1}$(win-win) This is a re-implementation of Luo et al.'s system [14]. Its configuration is "S_1 Fixed number of states" + "A_1 Technology Selection" + "R_1 Explicit Feedback". Its search engine actions include six retrieval technologies:

(1) increasing weights of the added query terms; (2) decreasing weights of the added query terms; (3) QCM [5]; (4) PRF (Pseudo Relevance Feedback) [17]; (5) Only use the last query in a session; and (6) Equally weights and combines all unique query terms in a session. The system employs 20 search engine actions in total and uses nDCG@10 as the reward.

$S_1A_1R_2$ This is a variation of $S_1A_1R_1$(win-win). Its configuration is "S_1 Fixed number of states" + "A_1 Technology Selection" + "R_2 Implicit Feedback". This system also uses 20 actions. Unlike win-win, its rewards are SAT Clicks (documents that receive user clicks and the time of user dwelling on is greater than 30 seconds [4]).

$S_1A_2R_1$ This system's configuration is "S_1 Fixed number of states" + "A_2 Term Weight Adjustment" + "R_1 Explicit Feedback". Specifically, the states in this approach are"Exploitation" and "Exploration". The term weights are adjusted similarly to Guan et al. [5] based on query changes. For example, if the user is currently under "Exploitation" and adds terms to the current query, we let the search engine take an action to increase the weights for the added terms.

$S_1A_3R_2$ This system's configuration is "S_1 Fixed number of states" + "A_3 Portfolio" + "R_2 Implicit Feedback". It contains a single state, which is the current query. It uses the last query in a session to retrieve the top X documents as in [21] and then re-ranks them to boost the ranks of the SAT Clicked documents. The actions are portfolios, i.e., all possible rankings for the X documents. For each ranked list D_i, the system calculates a reward and selects the ranked list with the highest reward.

$S_2A_1R_1$(UCAIR) This is a re-implementation of Shen et al.'s work [18]. Its configuration is "S_2 Varying number of states" + "A_1 Technology Selection" + "R_1 Explicit Feedback". Every query is a state. Query expansion and re-ranking are the two search technologies. In UCAIR, if a previous query term occurs frequently in the current query's search results, the term is added to the current query. The expanded query is then used for retrieval. After that, the re-ranking phase is performed based on the combination of each SAT Click's snippet the expanded query.

$S_2A_2R_1$(QCM) This is a re-implementation of Guan et al.'s system in [5]. Its configuration is "S_2 Varying number of states"+ "A_2 Term Weight Adjustment" + "R_1 Explicit Feedback". In QCM, every query is a state. The search engine actions are term weight adjustments. QCM's actions include increasing theme terms' weights, decreasing added terms' weights, and decreasing removed terms' weights. The term weights of each query is also discounted according to an reinforcement learning framework in [5].

$S_2A_1R_1$ This system's configuration is "S_2 Varying number of states"+ "A_1 Technology Selection" + "R_1 Explicit Feedback". It is built on the basis of $S_2A_2R_1$(QCM). Its search engine actions are two: QCM with or without spam detection. The spam detection is done by using Waterloo's spam scores. The rest settings are the same as in QCM.

Table 1. Search accuracy on TREC 2012 and TREC 2013 Session Tracks.

Approach (2012)	nDCG@10	nDCG	MAP	nERR@10
$S_1A_1R_1$(win-win)	0.2916	0.2875	0.1424	0.3368
$S_2A_1R_1$	0.2658	0.2772	0.1307	0.3105
$S_1A_1R_2$	0.2222	0.2733	0.1251	0.2464
$S_2A_2R_1$(QCM)	0.2121	0.2713	0.1244	0.2302
$S_2A_1R_1$(UCAIR)	0.2089	0.2734	0.1225	0.2368
$S_1A_3R_2$	0.1901	0.2528	0.1087	0.2310
$S_1A_2R_1$	0.1738	0.2465	0.1063	0.1877
$S_2A_3R_1$(IES)	0.1705	0.2626	0.1184	0.1890
Approach (2013)	nDCG@10	nDCG	MAP	nERR@10
$S_1A_1R_1$(win-win)	0.2026	0.2609	0.1290	0.2328
$S_2A_1R_1$	0.1676	0.2434	0.1132	0.1914
$S_2A_2R_1$(QCM)	0.1316	0.1929	0.1060	0.1547
$S_2A_1R_1$(UCAIR)	0.1182	0.1798	0.0927	0.1360
$S_2A_3R_1$(IES)	0.1076	0.1851	0.0966	0.1133
$S_1A_3R_2$	0.0987	0.1538	0.0761	0.1064
$S_1A_1R_2$	0.0964	0.2159	0.0689	0.1041
$S_1A_2R_1$	0.0936	0.1499	0.0740	0.0995

$S_2A_3R_1$(IES) This is a re-implementation of Jin et al.'s work [8]. Its configuration is "S_2 Varying number of states" + "A_3 Portfolio" + "R_1 Explicit Feedback". This system uses the top K documents as pseudo relevance feedback to re-rank the retrieved documents. It assumes each document's true relevance score is a random variable following a multi-variable normal distribution $\mathcal{N}(\boldsymbol{\theta}, \Sigma)$. $\boldsymbol{\theta}$ is the mean vector and is set as the relevance score calculated directly by [21]. The Σ is approximated using document cosine similarity. IES also uses Monte Carlo Sampling and a greedy algorithm called "Sequential Ranking Decision" to reduce the action space.

4.3 Search Accuracy

Table 1 shows the search accuracy of the above systems using TREC's effectiveness metrics for both datasets. The systems are decreasingly sorted by nDCG@10 in the table.

As we can see, $S_1A_1R_1$ (win-win) outperforms all other systems in both datasets. For example, in TREC 2012, $S_1A_1R_1$ (win-win) shows 37.5% improvement in nDCG@10 and 46.3% in nERR@10 over $S_2A_2R_1$ (QCM), a strong state-of-the-art session search system which uses a single search technology [5]. The improvements are statistically significant ($p < 0.05$, t-test, one-sided). It also shows 6.0% nDCG and 14.5% MAP improvements over QCM, however they are not statistically significant. Another system $S_2A_1R_1$, which also uses technology selection, improves 25.3% in nDCG@10 and 34.9% in nERR@10 over QCM, too. The improvements are statistically significant ($p < 0.05$, t-test, one-sided).

Table 2. Efficiency on TREC 2012 and 2013 Session Track. O(L) is the time complexity of conducting a Language Modeling retrieval. l is the number of alternative actions. K is the top K ranks. O(X) is the time complexity of re-ranking X documents. Z is the sample size of feedback documents.

Approach	TREC 2012		TREC 2013		BigO
	Wall Clock	CPU cycle	Wall Clock	CPU cycle	
$S_2A_3R_1$(IES)	9.7E4s	2.6E14	8.0E4s	2.2E14	$O(L+KZX^3)$
$S_1A_1R_2$	3.2E4s	8.6E13	1.8E4s	4.8E13	$O(lL)$
$S_1A_1R_1$(win-win)	3.1E4s	8.4E13	1.3E4s	3.5E13	$O(lL)$
$S_2A_1R_1$	6.6E3s	1.8E13	8.6E3s	2.3E13	$O(lL)$
$S_2A_2R_1$(QCM)	2.2E3s	5.8E12	1.9E3s	5.2E12	$O(L)$
$S_2A_1R_1$(UCAIR)	1.8E3s	4.8E12	0.8E3s	2.0E12	$O(L)$
$S_1A_2R_1$	1.1E3s	3.0E12	0.4E3s	1.0E12	$O(L)$
$S_1A_3R_2$	0.8E3s	2.2E12	0.3E3s	0.8E12	$O(L+X)$

It suggests that "A_1 Technology Selection", the meta-level search engine action, is superior to a single search technology, for example, term weight adjustment in QCM. Moreover, $S_1A_1R_1$(win-win) performs even better than $S_2A_1R_1$, where the former uses more search technologies than the latter. We therefore suggest that using more alternative search technologies can be very beneficial to session search.

4.4 Search Efficiency

In this section, we report the efficiency of these systems using a hardware support of 4 CPU cores (2.70 GHz), 32 GB Memory, and 22 TB NAS. Table 2 presents the wall clock running time, cpu cycles, as well as the Big O notation for each system. The systems are decreasingly ordered by wall clock time , which is measured in seconds.

All approaches, except $S_2A_3R_1$(IES), are able to finish within 1 day. Moreover, the experiment shows that $S_1A_3R_2$, $S_1A_2R_1$, $S_2A_1R_1$ (UCAIR), $S_2A_2R_1$(QCM) and $S_2A_1R_1$ are quite efficient and finished within 2.5 hours. $S_1A_1R_1$(win-win) and $S_1A_1R_2$ also show moderate efficiency and finished within 9 hours.

$S_2A_3R_2$(IES) is the slowest system, which took 27 hours to finish. We investigate the reasons behind its slowness. Based on Algorithm 1 in IES [8], the system first retrieves X documents using a standard document retrieval algorithm [21], then the algorithm has three nested loops to generate top K results by re-ranking. The first loop enumerates each rank position and its time complexity is $O(K)$. The second loop iterates over each retrieved document, thus its time complexity is $O(X)$. Inside the second loop, it first samples Z documents from the top K documents, then runs the third loop. The third loop enumerates each sample and has a time complexity of $O(Z)$. Inside the third loop, there is a matrix multiplication calculation for every retrieved document, which alone

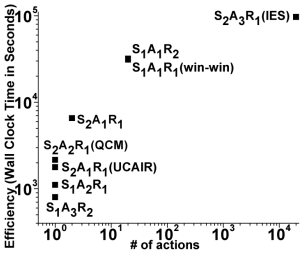

Fig. 1. Efficiency vs. # of Actions on TREC 2012.

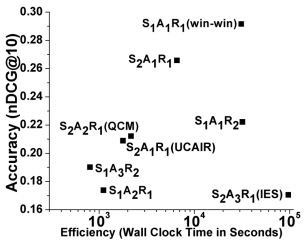

Fig. 2. Accuracy vs. Efficiency on TREC 2012.

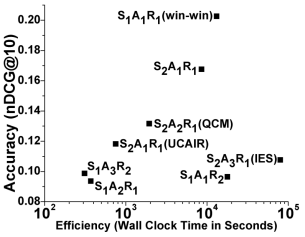

Fig. 3. Accuracy vs. Efficiency on TREC 2013.

attributes to a time complexity of $O(X^2)$. Therefore, IES's total time complexity is $O(KZX^3)$, which makes IES computationally demanding.

We also look into the time complexity of other systems and present their Big O notations in Table 2. We notice that $S_2A_2R_1$(QCM), $S_2A_1R_1$(UCAIR) and $S_1A_2R_1$ only perform one document retrieval, hence their time complexity is $O(L)$. $S_1A_1R_2$, $S_1A_1R_1$(win-win) and $S_2A_1R_1$ conduct l document retrievals, hence their time complexity is $O(lL)$. $S_1A_3R_2$ performs one document retrieval and one document re-ranking, hence its time complexity is $O(L + X)$. Their time complexities range from linear, e.g. $O(L)$ or $O(X)$, to quadratic, e.g. $O(lL)$, which suggests that these systems are efficient.

We see an interesting association between efficiency and the number of actions used in a system. Figure 1 shows that in TREC 2012, the systems' running time increases monotonically as the number of actions increases. It suggests that besides time complexity, the number of actions used in POMDP is another important factor in deciding its running time. We do not observe similar association between actions and accuracy for the systems under evaluation.

4.5 Tradeoff between Accuracy and Efficiency

Based on the search accuracy and efficiency results, we observe a trade-off between them, which is presented in Figures 2 and 3. They show that accuracy tends to increase when efficiency decreases. This is because systems with higher accuracy tend to be more computationally demanding. For instance, $S_1A_1R_1$(win-win) could achieve better accuracy but worse efficiency than $S_2A_1R_1$. We also find that $S_2A_1R_1$ (UCAIR) strikes a good balance between search accuracy and efficiency. With a simple feedback mechanism based on the vector space model, this system reaches high efficiency while can still achieve quite good nDCG@10. Overall, $S_1A_1R_1$(win-win) gives impressive accuracy with a fair degree of efficiency.

5 Our Recommendations

Giving the TREC Session task and typical computational resource as described in Section 4.4, our recommendation is the following. If more emphasis is put on accuracy rather than efficiency, we recommend $S_1A_1R_1$ (win-win) [14], whose settings are "Fixed number of states", "Technology Selection", and "Explicit Feedback" as the reward, for its highest search accuracy (Tables 1 and 2). If more emphasis is put on efficiency, e.g. with a limit of finishing the experiments within 1 hour, our recommendation will be $S_2A_2R_1$(QCM) [5], whose settings are "Varying number of states", "Term Weight Adjustment" as actions, and "Explicit Feedback" as the reward, for its high accuracy within the time constraint. In addition, we also recommend $S_2A_1R_1$ (UCAIR) [18], which is the runner-up in search accuracy among runs finishing within 1 hour, while only taking half as much time as QCM.

We have noticed that the number of actions heavily influences the search efficiency. Specifically, using more actions may benefit the search accuracy, while hurts the efficiency. For instance, with a lot of action candidates, $S_1A_1R_1$ (win-win) outperforms other runs in accuracy. However, the cost of having more actions in the model is that it requires more calculations and longer retrieval time. Therefore, we recommend a careful design of the number of total actions, when creating a new POMDP model, to balance between accuracy and efficiency.

6 Conclusion

This paper aims to provide guidelines for using POMDP models to tackle session search. Based on an extended set of IR algorithms that share the use of state, action and reward, we evaluate the various design options in designing suitable states, actions and reward functions for session search. The design options are evaluated against two major factors, search accuracy and search efficiency. We experiment and report our findings on the TREC 2012 and 2013 Session Track datasets. Finally, we make recommendations for a typical session search task for IR researchers and practitioners to use POMDP in session search.

From our experiments, we have learned that a model with more action options tends to have better accuracy but worse efficiency. It once again proves the importance of managing a good balance between accuracy and efficiency. We hope our work can motivate the use of POMDP and other reinforcement learning models in session search and provide a general guideline for designing *States*, *Actions*, and *Rewards* in session search.

7 Acknowledgments

The research is supported by NSF grant CNS-1223825, DARPA grant FA8750-14-2-0226, and a sponsorship from the China Scholarship Council. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

1. R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
2. L. B. Chilton and J. Teevan. Addressing people's information needs directly in a web search result page. In *WWW '11*, pages 27–36.
3. G. V. Cormack, M. D. Smucker, and C. L. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Inf. Retr.*, 14(5):441–465, Oct. 2011.
4. S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.*, 23(2):147–168.
5. D. Guan, S. Zhang, and H. Yang. Utilizing query change for session search. In *SIGIR '13*, pages 453–462.
6. K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in learning to rank online. In *ECIR'11*, pages 251–263.
7. K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4), Oct. 2002.
8. X. Jin, M. Sloan, and J. Wang. Interactive exploratory search for multi page search results. In *WWW '13*, pages 655–666.
9. T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *ICML '97*, pages 143–151.
10. L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.
11. E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2012 session track. In *TREC'12*.
12. E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2013 session track. In *TREC'13*.
13. M. L. Littman. The witness algorithm: Solving partially observable Markov decision processes. Technical report, Providence, RI, USA, 1994.
14. J. Luo, S. Zhang, and H. Yang. Win-win search: Dual-agent stochastic game in session search. In *SIGIR '14*.
15. J. R. Norris. *Markov Chains*. Cambridge University Press, 1998.
16. S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, Apr. 2009.
17. G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Readings in information retrieval*, 24:5, 1997.
18. X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM '05*, pages 824–831.
19. E. Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted cost. *Operations Research*, 26(2):282–304, 1978.
20. S. Yuan and J. Wang. Sequential selection of correlated ads by POMDPs. In *CIKM '12*, pages 515–524.
21. C. Zhai and J. Lafferty. Two-stage language models for information retrieval. In *SIGIR '02*, pages 49–56.
22. S. Zhang, J. Luo, and H. Yang. A POMDP model for content-free document re-ranking. In *SIGIR '14*.