Learning to Reinforce Search Effectiveness

Jiyun Luo, Xuchu Dong, Hui Yang
Department of Computer Science, Georgetown University
{jl1749,xd47}@georgetown.edu, huiyang@cs.georgetown.edu

ABSTRACT

Session search is an Information Retrieval (IR) task which handles a series of queries issued for a search task. In this paper, we propose a novel reinforcement learning style information retrieval framework and develop a new feedback learning algorithm to model user feedback, including clicks and query reformulations, as reinforcement signals and to generate rewards in the RL framework. From a new perspective, we view session search as a cooperative game played between two agents, the user and the search engine. We study the communications between the two agents; they always exchange opinions on "whether the current stage of search is relevant" and "whether we should explore now." The algorithm infers user feedback models by an EM algorithm from the query logs. We compare to several state-of-the-art session search algorithms and evaluate our algorithm on the most recent TREC 2012 to 2014 Session Tracks. The experimental results demonstrates that our approach is highly effective for improving session search accuracy.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval—*Information Search and Retrieval*

Keywords

Dynamic Information Retrieval Modeling; Reinforcement Learning; Stochastic Game; Session Search

1. INTRODUCTION

Session search is an Information Retrieval (IR) task which extends classic ad-hoc retrieval by handling more than one queries in a retrieval task. This form of search happens everyday when people search online using a series of queries for a complex information need, such as planning a trip to Paris or purchasing a new home. The process of submitting multiple queries in a session to accomplish a complex information need is called *session search.*

ICTIR'15, September 27–30, 2015, Northampton, MA, USA.

DOI: http://dx.doi.org/10.1145/2808194.2809468.

A session usually starts with the user issuing a query to the search engine. The user then receives a list of ranked documents that are ordered by the decreasing relevance to the query. The user then goes through the list and skims the snippets, clicks on some interesting ones and spends a fair amount of time reading them. This is a complete cycle of an ad-hoc retrieval process. Here in session search, we call one such information retrieval cycle a "search iteration" and it repeats in the session. In the next search iteration, the user either reformulates the previous query or writes a new one to start another search. The loop stops when the user's information need is satisfied or the user abandons the search [8]. As a result, there are a series of search iterations in a session – which include a series of queries $q_1, ..., q_n$, a series of returned documents $D_1, ..., D_n$, and a series of clicks $C_1, ..., C_n$, some of which are examined by the user for a long time and thus potentially highly relevant documents (They are called SAT-clicks, or *satisfactory clicked documents* [10]).

Session search has obtained increasing attentions in the IR community, for instance, in the recent dynamic information retrieval modeling approaches [34, 35] and the TExt Retrieval Conference (TREC) 2010-2014 Session Tracks [19, 20]. The current approaches include (1) extending existing IR techniques, such as using learning to rank and using large scale query logs, from one-shot query ad-hoc retrieval to session search, and (2) emerging efforts in applying reinforcement learning (RL) to session search. For instance, [13] proposed a Markov Decision Process (MDP) approach for session search, where they attributed document retrieval to a decision-making of term-weight adjustments, which they modeled as the actions that a search engine could take.

Our paper belongs to the RL-style session search algorithms, with its focus on modeling the two-way communication between the user and the search engine.

There are rich user and search engine interactions occurred during session search. We believe they are valuable resource and we should take advantage of that to improve search results effectiveness. For instance, Figure 1 illustrates the interactions and the messaging between agents in a session. The example is from TREC 2014 Session 52, which searches for *the efficiency, technology, and environment effect of hydropower.* At iteration 1 ($t = 1$), the user enters the first query "hydropower." Note that there is no browsing information at iteration 1. The search engine sends out its ranked documents as its message. At iteration 2 ($t = 2$), the user sends out the browsing information, including clicked documents (some of them are SAT-clicked[1]) in the browse

[1]In this work, we consider a document with dwell time longer

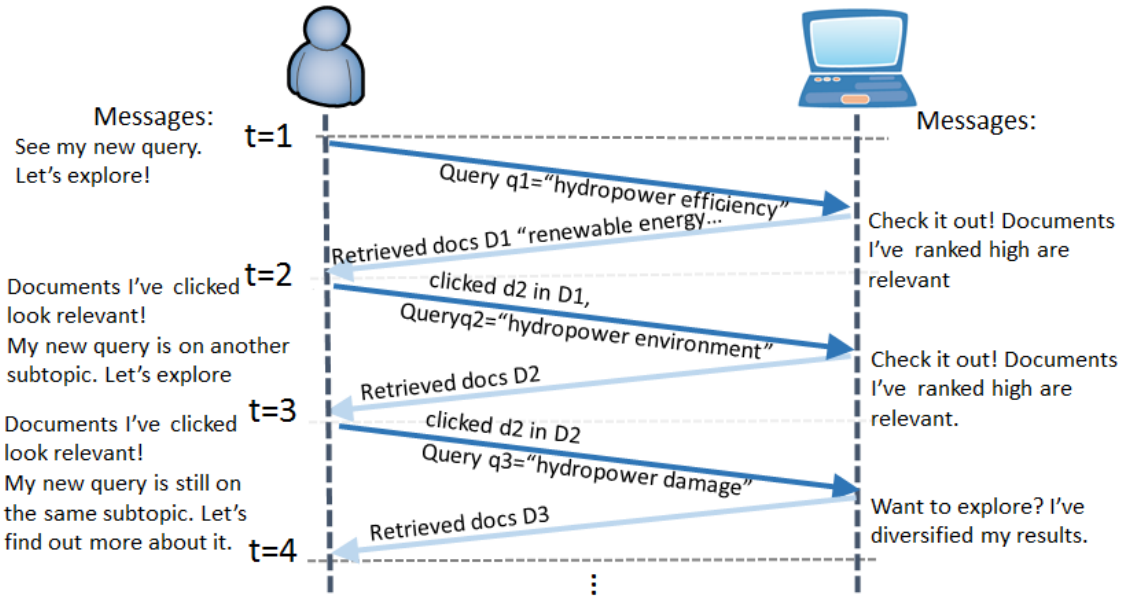

Figure 1: Interactions in Session Search. (Example is from TREC'14 Session 52. d_n is the n^{th} document in a ranked list D at iteration t.)

phase, and then sends out a reformulated query in the query phase. Both the browsing information and the query reformulations are messages (feedback) that the user sends to the search engine. The search engine then sends out a new set of documents in the rank phase for this iteration. The loop continues until the search is done.

In this paper, we propose to look at the interactions in session search from a new perspective: we view session search as a cooperative game played between the user and the search engine. In our point of view, the interactions between the two agents show the following characteristics:

Common goal. The two parties (agents) in session search, the user and the search engine, share a common goal, which is to find documents that can satisfy the ultimate information need. Throughout the process, the two agents cooperate and explore together, seeking a better understanding of the information need and how to satisfy it by finding relevant documents.

Exploration in the unknown information space. The user and the search engine interact in a complex search system to realize a goal: satisfying a ultimate information need. The search process is guided by a hidden, complex, sometimes vague information need, which is always in the user's mind, and may evolve and become clearer as the search develops. The user and the search engine work together step by step, exploring an unknown information space, to find the relevant documents to satisfy the information need. In the process, every search iteration could be generalization, specification, slight change, or completely change to the previous query. All these search attempts from the user, accumulate knowledge and information that allows the user to better understand the ultimate search goal, and the change and dynamics presented in the search process.

Equal partners. We view both the user and the search engine as autonomous agents who are *equal partners* in their cooperation. Our view is different from prior work on interactive search where users are the only main target in the study, They cooperate to achieve a common goal, which is to fulfill the information need for this session. The roles of the user and the search engine are not that one is teaching and another is learning. In this joint exploration, neither of them may have a clear picture of how to approach the information need. Sometimes neither of them even knows how to describe the information need. For instance, users may have difficulty articulating a query that precisely describes the information need, especially when the search task is complex. Therefore, we would like to point out that both parties are equal, and that they take turns in exploring the environment and communicating with each other to reach the goal.

Cooperative exploration. Both the user and the search engine can make decisions on relevance, as well as on when and where to explore the information space. A user can issue a new query from one subtopic to another. This drift drives the search process into a new direction. Similarly, the search engine can also drive the search process into new paths, by introducing diversity in the search results that it recommends. In our opinion, both agents are not only able to tell each other what documents they find to be relevant from their own points of views, but are also able to suggest to each other new directions for exploration.

In this paper we propose a new concept, universal messages, which are exchanged in session search between the two agents. Both the user and the search engine communicate and act using these messages. Basically, the two agents use *documents* and other actions, such as queries, to express their opinions on two things: (1) which documents are *relevant*, and (2) whether they would like to inform the other party to *explore* other subtopics as a team. The is to say, there is a two-way feedback messages that the user the and the search engine talk to each other every time when they communicate – *whether an agent judges the previously re-*

than 30 seconds are satisfactory clicks (SAT-Clicks).

Table 1: Messages Sent from the Senders

Messages	"These are Relevant!"	"Let's Explore!"
User:	"Hey, after reading the documents, those documents that I have clicked or SAT-clicked are potentially relevant. Please pay attention to those documents that I have clicked"	"I have written a query that *shifts* to new topics. I would like us to explore some new topics together."
Search Engine:	"Hello, the documents that I've ranked high for you are supposed to be relevant. The more relevant I think they are, the higher that I rank them. Please take a look."	I have a method to recommend novel documents to you as well. When I do it, I introduce some documents on new topics to increase *diversity*. Please take a look at those documents that I bring to the top positions in the ranking list but may seem not directly related to your current query; they are what I recommend on new topics and hope we both to explore on them."

trieved/recommended documents by the other agent are relevant and *whether an agent would like to lead both of them to explore and move to the next sub information need.*

Table 1 shows our interpretation of the meaning of their actions. We use this intuition to model the two-way feedback between the user and the search engine.

In this work, we propose an RL framework based on direct policy learning in partially observable stochastic game (POSG) to model the feedback among the two agents, aiming to improve the search effectiveness in sessions. We first propose a context bandit formulation of the two way communication between the two agents along two dimensions – relevance and novelty. We then use an Expectation Maximization framework [7] to learn the reward function for the reinforcement learning. Through learning a probabilistic reward function for each agent, we model the interactions between two equal partners in a uniform framework. Session search is further modeled as a multi-agent policy learning problem. We apply our algorithm to the TREC Session Tracks [19, 20]. TREC Session track is an evaluation for session search systems which has been organized by NIST annually from 2010 to 2014. The experiment results show that our approach are highly effective and outperforms the state-of-the-art session search systems.

The remainder of this paper are organized as follows. Section 2 presents the related work. Section 4 models the interactions between the agents. Section 5 presents the EM algorithm. Section 6 evaluates the approach. Section 7 fosters a discussion and Section 8 concludes the paper.

2. RELATED WORK

In this paper, we propose a novel session search framework based on a partially observable stochastic game (POSG), which belongs to the family of reinforcement learning (RL). Reinforcement Learning has been successfully applied to robotics [23, 28] and artificial intelligence in general [17], resource-bounded information extraction [18], and natural language grounding [1, 3, 5, 6, 9, 12, 21, 24, 33]. The most relevant previous work comes from grounding the actions of a reinforcement learning agent in natural language descriptions [3, 5, 33].

The two basic model-based algorithms for RLs are value iteration [2] and policy iteration [14]. However, in the research field of reinforcement learning, model-free algorithms for solving problems modeled by Markov Decision Processes (MDPs) are more popular [31]. Temporal-difference learning and Q-learning are two elementary approaches falling in this category.

RL algorithms can be computationally demanding. Therefore, for problems with a large state space or action space, generalization is often used to reduce the state-value or state-action mappings [30]. Branavan et al. proposed a Monte-Carlo search algorithm utilizing non-linear value function approximation for very large decision-making problems provided with external textual sources, such as game manuals[4]. They employ a four-layer neural network as a non-linear approximation to the state-action value function.

Our work is closely related to multiple-agent cooperative reinforcement learning [22, 25, 32]. Most multi-agent RL algorithms are proposed for deterministic environments, such as minmax-Q to learn an optimal policy in a zero-sum Markov game [25] and projecting central Q-table for distributed Q-learning [22]. In this paper, we pay more attention to cooperative multi-agent RL. Through experiments in the simulated prey/hunter game tasks, Tan found that cooperative agents can learn faster and better than independent agents, especially in joint tasks. Georgila et al. proposed an approach to learn the system policy and the simulated user policy concurrently in dialogue policies [11].

Zhang et al. introduced a supervision mechanism into large scale multi-agent RL systems [36]. In such a mechanism, all the agents are divided into various clusters. In one cluster, there is a supervisor agent and all the other are worker agents. Agents in the same or different clusters can communicate through specific protocols. Suggestion from a supervisor agent is used to improve the action policy of a worker agent. The supervision mechanism is utilized to accelerate the whole learning process. In their approach, the policy is represented by a probability distribution function of log-linear fashion with parameters and features. It needs a large amount of features extracted from the textual instructions, the target environment and candidate actions. In the training phase, based on samples drawn from the interaction with the target environment, the parameters are estimated by running the policy gradient algorithm under the goal of maximizing the expected future reward. It can adopt various reward functions. In case of zero-one rewards, this approach is equivalent to stochastic gradient ascent with a maximum likelihood objective. In the testing phase, suitable actions are chosen according to the mode of the policy

distribution.

Tan studied three types of cooperations among multiple agents, sharing instantaneous information, sharing historical episodes, and sharing learned policies [32].

Littman presented an algorithm called minmax-Q to learn an optimal policy in a zero-sum Markov game with two competitive agents [25]. This algorithm expressed the value function in a min-max form, and used an iterative procedure similar to Q-learning to solve this value function approximately.

For the reinforcement learning problem of multiple independently acting agents in deterministic environments, Lauer and Riedmiller constructed a distributed Q-learning algorithm [22]. Based on the assumption that all the agents cooperate towards one optimal goal, they projected the central Q-table into q-tables stored individually in the agents. Additionally, a mechanism of policy coordination between agents was used to guarantee the optimality of the final policy. However, such a method is not valid in stochastic environments.

For the problem of learning dialogue policies, Georgila et al. proposed an approach to learn the system policy and the simulated user policy concurrently, by using multi-agent RL techniques [11]. The two multi-agent RL algorithms, PHC (Policy Hill-climbing) and PHC-WoLF (Win or Learn Fast Policy Hill Climbing), showed better performance than the single-agent algorithm of Q-learning in experiments on a resource allocation negotiation scenario.

The work most similar to ours is [29]. In their work, Peskin et al. proposed a gradient descent method to find optimal policies in a partially observable multiple agent stochastic game. Their method used a finite state controller to construct policies with memory. In their model for a cooperative multi-agent environment, all the agents are treated identically, which means that the actions of different agents could happen in parallel. That is not suitable for session search, where there is a strict order of turns of the user's actions and the search engine's actions.

We are also inspired by [26], which introduced two Bayesian policy learning algorithms to learn human trainer's feedback strategies. One algorithm assumes a fixed universal feedback strategies for all human trainers, and the other infers trainer strategies from the training history. In their methods, human feedback was classified into three categories: explicit positive feedback, explicit negative feedback, and lack of feedback. Our approach differs from them by focusing on implicit positive and negative feedback. Because modern search engines rarely ask users to input explicit feedback. Another difference is that we study the user feedback from the perspectives of relevance and novelty rather than the perspectives of reward and punishment. Our work is the first to apply multiple-agent reinforcement learning on sessions search.

3. PROBLEM SETTING

Our framework is built on top of Partially Observable Stochastic Games (POSGs) [29], which is the multi-agent version of the Partially Observable Markov Decision Process (POMDP) [17]. A POSG is represented by a tuple, $<S, G, T, R>$, which indicates states, agents, transitions from state to state, and rewards. G is a set of agents, each represented by a tuple, $<A, O, B>$, indicating an agent's actions, observations, and beliefs over the states. The agents take inputs from the environment by making observations and output actions, which in turn influence the environment.

In a POMDP, hidden information can be modeled as hidden states, while visible signals can be modeled as observations or actions. In particular, states S is a (finite) set of discrete states. Actions A is a (finite) set of discrete actions. Observations O is a (finite) set of discrete symbols that an agent observes about the states. Reward $r = R(s_t = s, a_t = a, s_{t+1} = s')$ is the immediate reward when transitioning from state s to s' by taking action a at time t. In session search, the relationships between the two agents are cooperative. However, they are not identical; they differ in the actual actions and reward functions. We therefore do not use identical payoff models.

Observations We consider the received messages by the search engine agent as its observations to the user's behavior. The observations include *user clicks* and *user's query changes*, where query changes [13] are the syntactic editing changes between two adjacent queries q_{i-1} and q_i. They include positive query change ($+\Delta q$), that is the added terms in the new query q_i, and negative query change ($-\Delta q$), that is the deleted terms from the old query q_{i-1}. The user-side observations are retrieved documents D_t by the search engine at time step t.

Actions and Observation-Action Pairs We consider the action of the search engine agent as to generating a ranked document list. The search engine agent's action leads to *a ranked list of documents and their snippets.* (o_t, a_t) is a set of observation-action pairs. In this paper, o_t indicates at time t that we can observe how the user has browsed the previously retrieved search results, clicked the documents, and reformulated the query the the current search iteration. a_t indicates at time t, the search engine selects search options, executes the search algorithms and provides a ranked list of search results.

History We define *history* to keep track of the dynamic changes of states, observations, actions, and rewards during a search session. In this work, we treat the states as unknown in our partially observable model, the history only records a series of observations, actions and rewards generated at each step from search iteration 1 to the current iteration t. In this paper, we use 'iteration' and 'time step' interchangeably to refer to search iteration t. The *history* up to time t is

$$h_t = (o_1, a_1, r_1, o_2, a_2, r_2, \ldots, o_t, a_t, r_t).$$

Optimization In order to solve an RL problem, we need to learn its best policy. In a POMDP, a policy π is a function used to decide which actions for the agent to take at each time step, given a history of state and action pairs since time 0. By taking the selected action, the goal is to maximum a reward function, which, in the most widely used infinite horizon discounted model, is defined as the expected discounted long-term cumulative reward:

$$V_\theta(s_0) = E\big(\sum_{t=0}^{\infty} \gamma^t r_t | s_0\big)$$

which is also known as the *value function*, with γ as a discount factor that discounts the expected future rewards, since they have not been realized yet and could be inflated now. In this paper, we directly learn the policy from obser-

vation to actions

$$\pi : O \rightarrow A$$

and skip defining the states and beliefs.

4. MODELING TWO-WAY FEEDBACK

We are fascinated by the complexity of how users generate feedback to the search engine. We also would like to take the challenge to model the search engine as an autonomous agent who behaves as an equal partner to the user; and we model its feedback to the user as well. The two-way feedback is modeled as a contextual bandit for the proposed framework.

As we have discussed in the introduction section, user's query can be considered as a piece of short text that attempts to describe the information need. Documents or ranked document list returned by search engine can be considered as a 'big chunk' of text that the search engine used to illustrate what in its mind to consider how to describe the information need or what are relevant to the information need.

By showing the retrieved documents to the user, which is like showing search engine's understanding of the information need, the search engine complete its turn of contribute to drive the exploration process, one step closer to the information need. The user can also show his or her feedback via clicking documents or spending a certain amount of time to examine the snippets, title or documents. The explicit or implicit feedback can be captured. They are also text message sending from the user expressing a ranked list of text where the top ranked documents are more relevant and the lower ranked ones are less relevant. It is user's way of expression for what he or she believes that can describes the information need and relevant to the information need.

Hence, here we model the messages, i.e., the communication between the user and search engine here. The messages are about two dimensions: 1) "relevant dimension" – whether an agent thinks the returned documents by the other agent are relevant, and 2) "exploration dimension" – whether an agent would like to drive both agents to explore another subtopic.

To model the various forms of feedback in session search, we define a decision-making distribution $\mathcal{J}$, which is a distribution over the agents' decision-making states. We assume that an agent judges a set of ranked retrieved documents as relevant with probability

$$P(relevant) = 1 - \varepsilon$$

and, the agent judges that it would like to explore with probability

$$P(explore) = \mu$$

At time step t, given the observations o, action a, and target policy π^*, the decision-making distribution $\mathcal{J}$ is defined as:

$$p(\mathcal{J} = RE|o, a, \pi^*) = (1 - \varepsilon)\mu \quad (1)$$

$$p(\mathcal{J} = NRE|o, a, \pi^*) = \varepsilon\mu \quad (2)$$

$$p(\mathcal{J} = RNE|o, a, \pi^*) = (1 - \varepsilon)(1 - \mu) \quad (3)$$

$$p(\mathcal{J} = NRNE|o, a, \pi^*) = \varepsilon(1 - \mu) \quad (4)$$

Here, $\mathcal{J}$ can take four values: 'RE' means 'relevant' and 'explore'; 'NRE' means 'non-relevant' and 'explore'; 'RNE' means 'relevant' and 'non-exploratory'; 'NRNE' means 'non-relevant' and 'non-exploratory.'

Note that the best policy $\pi^*(o_t) = a_t$. The four formulas sum to one, and the highest value among the four indicates the most likely decision-making state. The corresponding value will be used to compose the immediate reward function to guide the reinforcement learning framework.

4.1 Feedback from the User

In our opinion, user feedback is presented in two forms in session search: clicks to indicate relevance, and query change to indicate both relevance and the desire to explore.

Here we use the time step t to index the search iterations. For the query q_t at the t^{th} search iteration, the set of retrieved documents for it is denoted as D_t. The set of documents that are previously retrieved for the previous query at the $(t-1)^{th}$ search iteration is called the previously retrieved documents and denoted as D_{t-1}. It is believed that the previously retrieved documents are very influential in the current run of search, too. They probably influence how the user will write the current query since the user have read some of the documents in D_{t-1} or at least have skimmed the snippets in the list.

The first dimension of judgment is related to *document relevance*.

At search iteration t, if the set of previously returned documents, D_{t-1} leads to one or more SAT clicks, the current state is likely to be relevant; otherwise it is non-relevant. The intuition is that if while examining D_{t-1}, if the user clicks some documents, it is a good indicator that the user considers the set of the documents that were retrieved by the search engine is relevant at this moment, i.e., at time t. Here we model the user clicks as feedback to previous run of search results.

Hence, we model the user's underlying judgment on relevance as:

$$\varepsilon = 1 - \frac{\#\text{ of SAT-Clicked documents} \in D_{t-1}}{\#\text{ of returned documents} \in D_{t-1}} \quad (5)$$

where $P(non\text{-}relevant) = \varepsilon$ represents the probability of "non-relevant" cases in the previously retrieved documents D_{t-1}. $P(relevant) = 1 - \varepsilon$ represents the probability of finding any "relevant" documents in D_{t-1}.

Another dimension of judgment is related to *exploitation vs. exploration*.

Here we study the changes between every consecutive queries as "query change" and denote it as Δq. For the changes between the current query q_t to the previous query q_{t-1}, their query changes at time t is denoted as Δq_t. It is the syntactic difference between the two queries, not the semantic differences. We use it for its simplicity and efficiency to calculate in an online algorithm.

Moreover, the query changes can be grouped into a few categories. They are added terms to the current query, removed query terms from the old query (not in the new query any more), and kept terms in both the current and the old query. We are especially interested in the added terms and removed terms, because they suggest the user's query intent to some degree. We denote them as positive query change $+\Delta q_t$ or negative query change $-\Delta q_t$.

Continue on the idea of the second dimension of judgment.

Table 2: TREC 2014 Session 1011

Information need:

You would like to provide your friend with relevant information about: different ways to quit smoking, programs available to help quit smoking, benefits of quitting smoking, second effects ...

query 1: quit smoking
query 2: quit smoking hypnosis
query 3: side effects quit smoking

The idea is that given that previously retrieved documents D_{t-1} is the message from the search engine and the current query change Δq_t is the message from the user, if an added query term $+\Delta q_t$ appeared in D_{t-1}, it is very likely that the user will stay with the same subtopic from iteration $t-1$ to t. It is a state that we call 'exploitation.'

On the other hand, if the added term $+\Delta q_t$ did not appear in D_{t-1}, it is quite likely that the user has moved to the next subtopic from iteration $t-1$ to t. It is a state that we call 'exploration.'

We model the user's underlying judgment on exploration as:

$$\mu = 1 - \frac{\#\text{ query changes} \in D_{t-1}}{\#\text{ of permutations of query terms} \in q_t} \quad (6)$$

where $P(exploration) = \mu$ represents the probability of "exploratory" cases at time t, and $P(exploitation) = 1 - \mu$ represents the probability of "exploitation" at time t.

Use an example to show our formulations. In the second iteration of Session 1011 in Table 2, there are 3 terms in the query "quit smoking hypnosis". The only added term is "hypnosis". It appears in "clueweb12-0010wb-76-01490", a SAT-clicked document in the previously set of retrieved documents.

In addition, there are 10 documents displayed to the user in the first 10 search results (SERP). Only one out of the 10 receives a SAT-click.

Therefore, for this iteration,we have $\epsilon = 1 - \frac{1}{10} = 0.1$ and $\mu = 1 - \frac{1}{3!} = 0.83$.

4.2 Feedback from the Search Engine

Likewise, the search engine's feedback is also presented in two forms: One to indicate what the search engine thinks are *relevant*, and anther to indicate what the search engine desires to explore in the process.

For the first dimension, we consider that the search engine uses the top-scored or top-ranked documents ranked by itself, to indicate relevance.

It means that we can use the similarity scores that the search engine assigns to each retrieval result to indicate the search engine's judgment on relevance. The assumption is that across different queries, query length normalization is taken care of, and the search engine is able to measure query-to-document similarity in a relatively consistent manner. However, in the datasets that we experiments on, such similarity scores are not available. Therefore, instead of using the actual ranking scores, we use the Maximum Likelihood Estimation (MLE) of relevant documents among the top ranked documents to present search engine's judgment on this dimension.

At the t^{th} search iteration, we model the search engine's underlying judgment on relevance by leveraging the probability that a top returned document is indeed relevant, which means the document appears in the ground truth as a relevant document.

The estimation of ε for search engine is

$$\varepsilon = 1 - \frac{\#\text{ of relevant documents in top } n \text{ retrieved}}{n} \quad (7)$$

Here, $P(NR) = \varepsilon$ represents the overall probability of "non-relevant" cases at time t. $P(R) = 1 - \varepsilon$ represents the probability of finding any "relevant" documents at time t in the top returned documents. n is empirically set to 5 in our experiments.

The second dimension is about *exploration*. Here we consider that the search engine introduces diversity in the retrieval results, to indicate the desire to explore.

To express its feedback/opinion on whether to *exploit or explore*, the search engine show different degrees of diversity in its returned results. The idea is that the search engine can return diversified results – i.e. multiple topic clusters in its results – to promote "exploration." We can detect this by observing the word distribution and indirectly measuring the diversity in the top returned snippets. According to the Zipf's law, assuming the top returned documents or snippets are on a single topic, it is expected that the slope of the word distribution will be steeper. On the other hand, if the documents are diversified, it is expected that the slope of the word distribution will be flatter.

We therefore model the search engine's underlying message on exploration as

$$\mu = 1 - \frac{\text{total \# of the top } m \text{ frequent non-stop-words in } D_t}{\text{total \# of non-stop-words in } D_t} \quad (8)$$

Here, $P(exploration) = \mu$ represents the overall probability of how "exploratory" the results are by measuring the percentage of top m occurring words in all words returned in the search results. Here we only examine the first 10 snippets since they are the content that the search engine would like to use to influence the user at the first sight. They are what the search engine presents in front of the user, and would like to use them to influence the user the most to do a cooperative exploration together. We empirically set m to 5.

We use an example to show the details. In the second iteration of Session 1011, 8 out of the top 10 documents retrieved by the search engine are highly relevant, as provided in the ground truth. Therefore, $\epsilon = 1 - \frac{4}{5} = 0.2$.

Moreover, there are 428 non-stop-words totally in the top 10 snippets, and the most frequent 5 words are:

"$smoke''$(59), "$quit''$(34), "$hypnosis''$(30), "$stop''$(19), "$button''$(7)

The total counts of these 5 most frequent non-stop-words {"smoke", "quit", "hypnosis", "stop", "button"} is 123. Therefore, $\mu = \frac{123}{428} = 0.29$.

Based on the estimation for the two dimensions, the maximum judgment distribution value is 0.568 which means "relevant and exploitation" for this iteration, which is correct based on our manual evaluation.

Algorithm 1 Feedback Learning by EM.

1: **procedure** FEEDBACKLEARNING(π, h)
2: $\pi \leftarrow randomPolicy()$
3: $\varepsilon \leftarrow random(0,1)$, $\mu \leftarrow random(0,1)$
4: **while** the search session has not finished **do**
5: $o_t \leftarrow observe()$
6: $a_t \leftarrow \pi(o_t)$
7: $h_t \leftarrow [h_{t-1}, (o_t, a_t)]$
8: **repeat**
9: **E-Step:**
10: compute $P(\mathcal{J}|o_t, a_t, \pi)$ by Eq. (2) to (5)
11: $j = \text{argmax}_{j' \in [`RE', `NRE', `RNE', `NRNE']} P(J = j'|o_t, a_t, \pi)$
12: $r(t,h) = P(\mathcal{J} = j|o_t, a_t, \pi)$
13: $\pi' \leftarrow \text{argmax}_\pi \int_0^1 \int_0^1 p(\varepsilon, \mu|h, \pi) \ln p(h, \varepsilon, \mu|\pi) d\varepsilon d\mu$
14: $a'_t = \pi'(o_t)$
15: performAction(agent, a'_t)
16: **M-Step:**
17: compute ε' and μ'
18: **until** $\varepsilon = \varepsilon'$ and $\mu = \mu'$
19: $t \leftarrow t + 1$
20: **end while**
21: $\pi^* = \pi$
22: **output** π^*

In future work, we will learn n and m through supervised learning for better prediction accuracy.

5. AN EM ALGORITHM FOR FEEDBACK LEARNING

We propose to use Expectation Maximization [7] to compute the maximum likelihood estimation (MLE) of the decision-making distribution.

Algorithm 1 outlines the EM algorithm. The EM algorithm starts with a random/initial estimation of the two hidden parameters, ϵ and μ, and an initial random policy, which generates random actions for both agents given an observation. For the history up to the t^{th} search iteration, we also be able to go into the history and go through iteration by iteration to update the policy.

In the Expectation step, we compute the decision-making distribution based on Equations (1) to (4). The most likely joint decision by both dimensions are indexed by j. The reward function $r(t,h)$ is calculated by the decision-making distribution. A new policy is estimated by finding the best policy at step t given the current estimates of ϵ and μ, at line 13. At line 14, an action is selected based on the new estimate of the policy, and at line 15, the corresponding agents, either the user or the search engine, will perform the action to the documents. If it is the search engine's turn, it will retrieve documents. If it is a user's turn, the user will click or browse the documents.

In the Maximization step, we re-compute ε and μ at time t based on the new estimate of π' and the new actions yielded from π', by using the formulas (5) to (8).

The loop repeats until the history is running out or reaching convergence. The algorithm then outputs the final learned policy π^*.

6. EXPERIMENTS

We apply our model to the recent TREC Session Tracks. The TREC Session Track [19, 20] has has been running since 2010 to 2014. It is an annual evaluation for session search systems conducted at the National Institute of Standards and Technology (NIST). For each session, queries, top 10 retrieved documents and interaction information in the current session is given in a query log. The last query that the user issues does not have interaction data.

A query log is provided to the participants. The log was created by the following process. Users are provided search topics containing the information need for a session. The information need usually contains multiple sub-topics. Table 2 lists an example of sessions from TREC Session Tracks.In order to find relevant documents for the search topic, users formulate a series of queries, examine the returned snippets, and click some urls to read the full documents if the snippets seem worthy to read more. It also records every user's query formulations, click operations and dwell time on documents into the log file. The search task is to retrieve a ranked list of 2,000 documents for the last query that users' trigger in each search session. The interactions are recored in the log.

The task of TREC Session Tracks is to retrieve a list of 2,000 documents for the last query in a session to satisfy the information need for the entire session. The document collection used to build the index is ClueWeb. In TREC 2012, the corpus is ClueWeb09 CatB with 50 million English web pages crawled in 2009. In 2013 and 2014, the corpus is ClueWeb12 CatB with also 50 million English web pages crawled in 2012. The participating systems are supposed to retrieve documents for the last query. The participating systems are not allowed to use the topics when performing retrieval. In our implementation, the spam documents are removed if their Waterloo spam scores are less than 70. Duplicated documents are removed too.

We evaluate our algorithm on the TREC Session 2012, 2013 and 2014 data for their recency and the completeness of the ground truth data. In earlier years of TREC Session Tracks, such as TREC 2010 and 2011, the tasks and how the sessions were created were different from the recent years; we therefore did not experiment on them. The tasks in this set of experiments are equivalent to TREC 2012 RL4, TREC 2013 RL2 and TREC 2014 RL2.

6.1 Evaluation Metrics

One of the most attractive features of session search algorithms is that they have the potential to perform early retrieval for documents relevant to later queries or to the entire session, even if the document may not seem relevant to the current query. Using the relevance judgment for the whole session, we can evaluate at earlier iterations how well a search engine can predict and produce relevant documents for future queries. We therefore measure the search accuracy at each search iteration and term this accuracy as "immediate search accuracy". We evaluate it on TREC 2012, TREC 2013, and TREC 2014 Session Tracks using nDCG@10 [16]. The six grades of relevance relevance judgments are provided by NIST.

6.2 Systems to Compare

We compare the following session search systems in the experiments.

- **lemur**, which is a state-of-the-art ad-hoc search en-

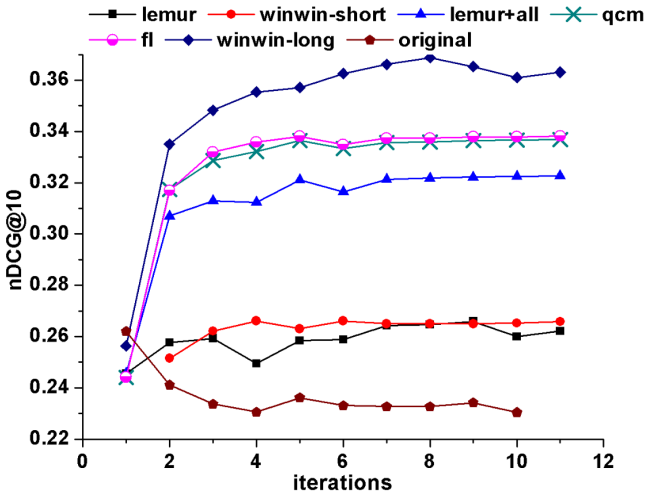

Figure 2: Immediate Search Accuracy (TREC 2012).

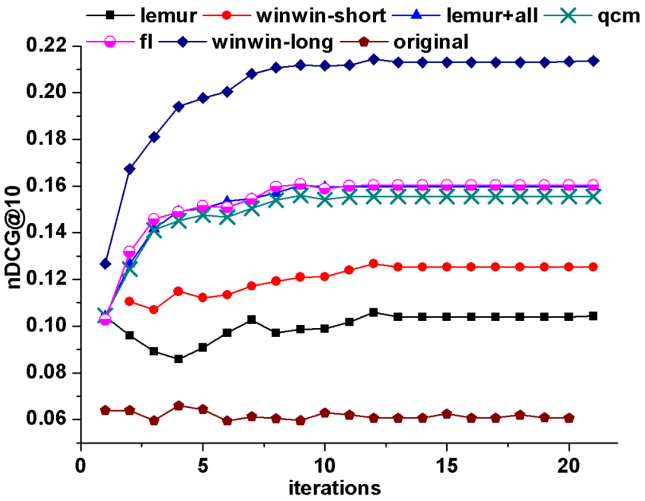

Figure 3: Immediate Search Accuracy (TREC 2013).

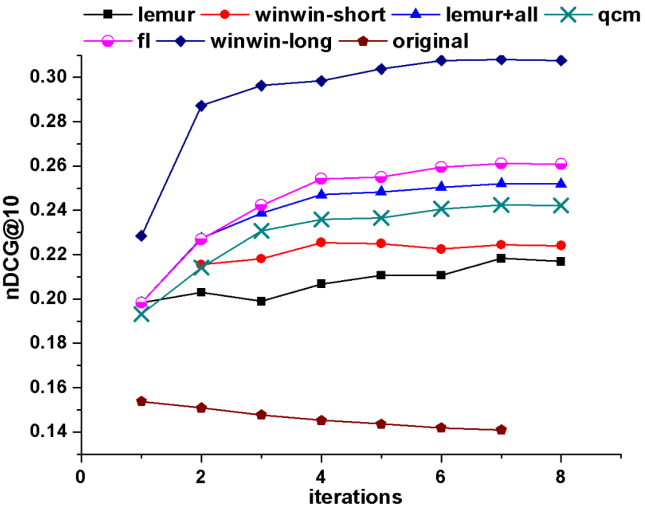

Figure 4: Immediate Search Accuracy (TREC 2014).

gine.[2] We use Lemur's multinomial language modeling and Dirichlet smoothing with the smoothing parameter $\mu = 5000$. The last query of each session is used as a query and sent to Lemur.

- **lemur+all**. Lemur with all queries in a session combined as a single query, feeding into it. For instance, TREC 2014 session 1011 yields a combined query *quit smoking quit smoking hypnosis side effects quit smoking*. Repeated terms are kept as they are.
- **qcm**, the query change retrieval mode, which is a state-of-the-art session search system. It is proposed by Guan et. al. [13].
- **winwin-short**, the win-win search session search framework proposed by Luo et. al. in [27]. This system models session search as POMDPs and works like a meta-search algorithm. Winwin-short only use the user clicks as a short-term feedback, to guide the session search.
- **winwin-long** is another version of winwin. Different from winwin-short, it uses an ideal reward function nDCG@10, which is generated by using the ground truth documents. Here we use winwin-long as an upper bound.
- **fl**, the **feedback learning** framework proposed here.

6.3 Immediate Search Accuracy

In order to compare the average immediate retrieval accuracy over all sessions, we first obtain the max lengths t_{max} of the sessions in a dataset. For TREC 2012, the maximum session length is 11. For TREC 2013, the maximum session length is 21. For the top 100 sessions in TREC 2014, the maximum session length is 8.

We compare the proposed algorithm *fl* with the *original* run, which is the retrieval results of the search engine used by the organizers to generate the sessions. We also compare our models to *lemur*, *lemur+all*, *qcm*, *winwin-short*, and *winwin-long* in this experiment. Among them, *qcm* and *winwin* are both RL-style session search systems, without using feedback learning.

Fig 2, 3 and 4 show the comparison result of above algorithms' immediate retrieval accuracy over TREC 2012, 2013 and 2014 datasets. Note that for the original run, it has one less iteration than the other runs, because the log file doesn't record retrieval results for the current/last query, which means a session with n steps only has $n - 1$ steps in the original run. For the winwin-short run, we don't plot the point of the first iteration, because winwin-short uses user clicks in the current session as reward signal, but at the first iteration there is no user clicks available yet.

We are happy to see that the proposed algorithm, fl, performs the best besides the upper bound algorithm (winwin). It is a strong indicator that our algorithm is very effective for session search. Our algorithm leads the immediate search accuracy since the second search iteration, and the advan-

[2]lemurproject.org.

tage increases as the number of iterations increase when the session develops.

Moreover, for all three TREC datasets, the immediate accuracy of lemur+all, qcm, winwin-long and fl all climb up, when the session continues and feedbacks are continually collected. We observe a boost of search accuracy around the second search iteration for lemur+all, qcm, winwin-long and fl in TREC 2012. And we observe the same phenomenon for them in both TREC 2013 and TREC 2014. The gains from the later queries are marginal, which suggests that the first few queries may be more representative for the whole session than the later ones.

In addition, we observe a consistent retrieval accuracy order of above algorithms over all three TREC datasets. Since iteration 2, the retrieval accuracy order is winwin-long $\geq$ fl, qcm and lemur+all $\geq$ winwin-short $\geq$ lemur $\geq$ original. Here winwin, fl, qcm and lemur+all utilize every queries in the current session. Winwin-short shows a better retrieval accuracy than lemur, however it is the lowest nDCG@10 score among all RL-style algorithms. It suggests that pure click models which use clicks as the reward function is not adequate. More complex modeling, such as what we proposed here, including the query reformulations, previously retrieved documents and clicks, as a whole to produce the reward function is more promising.

Finally, we observe that most runs reach their convergence after 5 or 6 search iterations. It suggests that in a long session, feedbacks received after step 5 or 6 provide less benefits to the search algorithm than feedbacks from the earlier iterations. It suggests from another angle that the proposed session search algorithm is able to reach a pretty good whole-session search accuracy at an early stage, which will greatly save the user's search efforts in real-time interactive search.

We set qcm as the baseline system and performed a one-tailed t-test for proposed algorithms against the baseline. Note that qcm is a strong baseline. Nonetheless, the experiments show that fl could achieve a 3% and a 5% absolute nDCG@10 improvement over qcm for the 2^{nd} and 3^{rd} iterations on the TREC'13 dataset. The improvements are statistically significant, with p-value=0.05. fl also achieves a 2%$\sim$7% absolute nDCG@10 improvement over qcm for all iterations on the TREC'14 dataset, which is statistically significant with p-value=0.05.

7. DISCUSSION

Our solution to the feedback-learning framework adopts multiple-agent reinforcement learning. EM algorithms can often be guaranteed to converge to local minimum points. When there are two (or more) agents in the process, in theory, they are able to jointly reach the local optima for a common solution, which is known as the Nash equilibrium [15] in game theory.

To reach Nash equilibrium, it is assumed that all agents are rational and try their best to reach own local optima. This assumes that the user is rational and always produces good queries to show the search intent. However, the assumption is often not true. We see many non-optimal user actions, which impact negatively on the retrieval.

For instance, TREC 2014 Sessions 232 and 89 are two session sharing the same information need – "*You are writing a summary article about the Pocono Mountains region. Find as many relevant articles as you can describing the region, things to see and do there ...*" In Session 232, the user submitted three queries: "pocono mountains," "mountains," and "mountains." We can see that the second and the last query are not very informative. It turns out that the number of relevant documents retrieved in the top 10 results drops from 6 to 4 when transitioning from the first query to the rest. It shows that when a user did not work hard enough to produce reasonable and cooperative queries, it is not surprising to see a negative impact on retrieval results. This suggests that it is important for both parties to commit to cooperative activities in session search. In the future, our research could potentially be used to detect user behaviors that could be explained by rational models and develop new user models from there.

On the contrary, in Session 89, the user submitted "Pocono Mountains", "Pocono Mountain parks", "Pocono Mountain things to do", "Pocono Mountain community", "Pocono Mountain community". Among the 6 relevant search results from the first query, the user clicked and examined "clueweb12-0304wb-75-03048" for 23.77 seconds. The document talks about "*... State National Parks Forests ... The Camelbeach Mountain Waterpark has hours of family-friendly activities for you. ... Hickory Run State Park and Boulder Field has acres of land sure to break in the toughest of hiking footwear ....*" The word "park" appears multiple times and probably inspired the user to create the second query "Pocono Mountain parks." The matching between the query change and the previously retrieved documents is interpreted as "exploitation" in our model, which generates positive reward for $r(t, h)$ as if the user performs optimally. As a result, in this run of search, the number of relevant documents in top 10 boosts from 6 to 8. It suggests that when the user behaviors align well with our models, i.e., the user acts optimally, the cooperation between the user and the search engine functions well and yields good retrieval results. It leaves opportunities for future work in query suggestion.

8. CONCLUSION

In this paper, we study the interactions in session search from a new perspective. We believe we are the first to consider the user and the search engine as two equal partners who collaborate and explore together to achieve a common goal – satisfying the information need. Based on this assumption, we model two-way feedback from both the search engine and the user, such as user clicks and query formulations, as rewarding signals to guide session search in a reinforcement learning framework.

Our work is the first to apply multiple-agent reinforcement learning on sessions search. By modeling the two-way feedback from both the relevance and the novelty perspectives, we successfully integrate the rich form of feedbacks in sessions, both explicit and implicit, and use them as rewards to enhance search accuracy in session search. In the experiment, we compare our approaches' search accuracy with state-of-the-art session search algorithms over three years of TREC Session Track data. The results show that our approach greatly improves the immediate search accuracy by effectively utilizing abundant feedback in session search.

9. ACKNOWLEDGMENT

The research is supported by DARPA FA8750-14-2-0226, NSF IIS-1453721, and NSF CNS-1223825. Any opinions, findings, conclusions, or recommendations expressed in this

paper are of the authors, and do not necessarily reflect those of the sponsor.

References

[1] Y. Artzi and L. Zettlemoyer. Bootstrapping semantic parsers from conversations. In *EMNLP '11*.

[2] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, 1987.

[3] S. R. K. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. Reinforcement learning for mapping instructions to actions. In *ACL '09*.

[4] S. R. K. Branavan, D. Silver, and R. Barzilay. Non-linear monte-carlo search in civilization ii. In *IJCAI'11*.

[5] S. R. K. Branavan, L. S. Zettlemoyer, and R. Barzilay. Reading between the lines: learning to map high-level instructions to commands. In *ACL '10*.

[6] D. L. Chen and R. J. Mooney. Learning to sportscast: A test of grounded language acquisition. In *ICML '08*.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Roy. Statist. Soc. Ser. B*, 39(1):1–38, 1977.

[8] A. Diriye, R. White, G. Buscher, and S. Dumais. Leaving so soon?: Understanding and predicting web search abandonment rationales. In *CIKM '12*, pages 1025–1034, 2012.

[9] J. Eisenstein, J. Clarke, D. Goldwasser, and D. Roth. Reading to learn: constructing features from semantic abstracts. In *EMNLP '09*.

[10] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.*, 23(2), Apr. 2005.

[11] K. Georgila, C. Nelson, and D. Traum. Single-agent vs. multi-agent techniques for concurrent reinforcement learning of negotiation dialogue policies. In *ACL '14*.

[12] D. Goldwasser, R. Reichart, J. Clarke, and D. Roth. Confidence driven unsupervised semantic parsing. In *HLT '11*.

[13] D. Guan, S. Zhang, and H. Yang. Utilizing query change for session search. In *SIGIR '13*.

[14] R. Howard. *Dynamic Programming and Markov Process*. MIT Press, 1960.

[15] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *ICML '98*.

[16] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[17] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *J Artificial Intelligence Res.*, 4:237–285, 1996.

[18] P. Kanani, A. McCallum, and S. Hu. Resource-bounded information extraction: Acquiring missing feature values on demand. In *Advances in Knowledge Discovery and Data Mining*, 2010.

[19] E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2013 session track. In *TREC'13*.

[20] E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2014 session track. In *TREC'14*.

[21] R. J. Kate and R. J. Mooney. Learning language semantics from ambiguous supervision. In *AAAI '07*.

[22] M. Lauer and M. A. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *ICML '00*.

[23] V. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner, and S. Zhang. BIG: An Agent for Resource-Bounded Information Gathering and Decision Making. *Artificial Intelligence Journal, Special Issue on Internet Information Agents*, 118(1-2):197–244, 2000.

[24] P. Liang, M. I. Jordan, and D. Klein. Learning semantic correspondences with less supervision. In *ACL '09*.

[25] M. L. Littman. markov games as a framework for multi-agent reinforcement learning. In *ICML '94*.

[26] R. T. Loftin, J. MacGlashan, B. Peng, M. E. Taylor, M. L. Littman, J. Huang, and D. L. Roberts. A strategy-aware technique for learning behaviors from discrete human feedback. In *AAAI '14*.

[27] J. Luo, S. Zhang, and H. Yang. Win-win search: Dual-agent stochastic game in session search. In *SIGIR '14*.

[28] D. Meger, P.-E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little, and D. G. Lowe. Curious george: An attentive semantic robot. *Robot. Auton. Syst.*, 56(6):503–511, 2008.

[29] L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling. Learning to cooperate via policy search. In *UAI '00*.

[30] J. Schmidhuber. Sequential decision making based on direct search. In R. Sun and C. L. Giles, editors, *Sequence Learning: Paradigms, Algorithms, and Applications*. Springer, 2001.

[31] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[32] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML '93*.

[33] A. Vogel and D. Jurafsky. Learning to follow navigational directions. In *ACL '10*.

[34] H. Yang, M. Sloan, and J. Wang. Tutorial on dynamic information retrieval modeling. In *SIGIR '14*.

[35] H. Yang, M. Sloan, and J. Wang. Tutorial on dynamic information retrieval modeling. In *WSDM '15*.

[36] C. Zhang, S. Abdallah, and V. Lesser. Efficient multi-agent reinforcement learning through automated supervision. In *AAMAS '08*.