Utilizing Query Change for Session Search

Dongyi Guan, Sicong Zhang, Hui Yang
Department of Computer Science
Georgetown University
37th and O Street, NW, Washington, DC, 20057
{dg372, sz303}@georgetown.edu, huiyang@cs.georgetown.edu

ABSTRACT

Session search is the Information Retrieval (IR) task that performs document retrieval for a search session. During a session, a user constantly modifies queries in order to find relevant documents that fulfill the information need. This paper proposes a novel query change retrieval model (QCM), which utilizes syntactic editing changes between adjacent queries as well as the relationship between query change and previously retrieved documents to enhance session search. We propose to model session search as a Markov Decision Process (MDP). We consider two agents in this MDP: the user agent and the search engine agent. The user agent's actions are query changes that we observe and the search agent's actions are proposed in this paper. Experiments show that our approach is highly effective and outperforms top session search systems in TREC 2011 and 2012.

Categories and Subject Descriptors

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

Keywords

Session search; query change model; retrieval model

1. INTRODUCTION

Session search is the Information Retrieval (IR) task that retrieves documents for a search session [4, 8, 13, 14, 15, 25, 32]. During a search session, a user keeps modifying queries in order to find relevant documents that fulfill his/her information needs. In session search, many factors, such as relevance feedback, clicked data, changes in queries, and user intentions, are intertwined together and make it a quite challenging IR task. TREC (Text REtrieval Conference) 2010-2012 Session tracks [18, 19, 20] studied session search with a focus on the "current query" task, which retrieves relevant documents for the current/last query in a session based on previous queries and interactions. Table 1 shows examples from the TREC 2012 Session track.[1]

[1]All examples mentioned in this paper are from TREC 2012. For simplicity, we use 'sx' to refer to a TREC 2012 session where x is the session identification number.

Table 1: Examples of TREC 2012 Session queries.

session 6	session 28
1.pocono mountains pennsylvania	1.france world
2.pocono mountains pennsylvania hotels	cup 98 reaction
3.pocono mountains pennsylvania things to do	stock market
4.pocono mountains pennsylvania hotels	2.france world
5.pocono mountains camelbeach	cup 98 reaction
6.pocono mountains camelbeach hotel	3.france world
7.pocono mountains chateau resort	cup 98
8.pocono mountains chateau resort attractions	session 32
9.pocono mountains chateau resort getting to	1.bollywood
10.chateau resort getting to	legislation
11.pocono mountains chateau resort directions	2.bollywood law
session 85	session 37
1.glass blowing	1.Merck lobbists
2.glass blowing science	2.Merck lobbying
3.scientific glass blowing	US policy

From Table 1, we notice that queries change constantly in a session. The patterns of query changes include general to specific (*pocono mountains* → *pocono mountains park*), specific to general (*france world cup 98 reaction* → *france world cup 98*), drifting from one to another (*pocono mountains park* → *pocono mountains shopping*), or slightly different expressions for the same information need (*glass blowing science* → *scientific glass blowing*). These changes vary and sometimes even look random (*gun homicides australia* → *martin bryant port arthur massacre*), which increases the difficulty of understanding user intention. However, since query changes are made after the user examines search results, we believe that *query change is an important form of feedback.* We hence propose to study and utilize query changes to facilitate better session search.

One approach to handle query change is to classify them based on various types of explorations [20], such as specification, generalization, drifting, or slight change, then perform retrieval. Another approach is mapping queries into semantic graphical representations, such as ontologies [7] or query flow graphs developed from query logs [2], then studying how queries move in the graphs. However, ontology mapping is challenging [17], which may introduce inaccurate intermediate results and hurt the search accuracy. Moreover, relying on large scale query logs may not be applicable due to lack of such data. Therefore, although these approaches have been applied to IR tasks such as query reformulation [3] and query suggestion [2, 30], they have yet to be directly applied to session search. It is therefore necessary to explore new solutions to utilize query change for session search.

We propose to model session search as a Markov Decision Process (MDP) [16, 28], which is applicable to many human decision processes. MDP models a state space and an action space for all agents participating in the process. Actions from the agents influence the environment/states and the environment/states influence the agents' subsequent actions

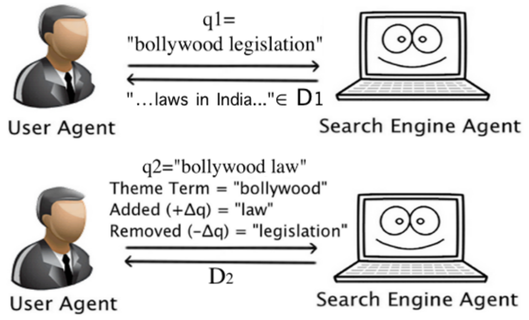

Figure 1: Session search MDP. (example is from s32)

based on certain policies. A transition model between states indicates the dynamics of the entire system. In our MDP, *queries* are modeled as states. Previous queries that the user wrote influence the search results; the search results again influence the user's decision of the next query. This interaction continues until the search stops.

As illustrated in Figure 1, we consider two agents in this entire process: the user agent and the search engine agent. The user agent's actions are mainly human actions that are able to change search results, such as adding and deleting query terms, i.e. *query change*. Clicking is a human action; however, it does not explicitly impact the retrieval. Therefore, it is not considered as a user action here. Query change is the only form of user action in this paper. Based on the user actions, we design corresponding policies for the search engine agent; which is the main focus of this paper.

It is difficult to interpret the user intent [5, 31] behind query change. For instance, for a query change from *Kurosawa* to *Kurosawa wife* (s38), there is no indication about 'wife' in the search results returned for the first query. However, *Kurosawa's wife* is actually among the information needs provided to the user by NIST's topic descriptions. Our experience with TREC Session tracks suggests that information needs and previous search results are two main factors that influence query change. However, knowing information needs before search could not easily be achieved. This paper focuses on utilizing evidence found in previous search results and the relationship between previous search results and query change to improve session search.

In this paper, we summarize various types of query changes based on potential user intents into user agent policies. We further propose corresponding policies for the search engine agent and model them in the query change retrieval model (QCM), a novel reinforcement learning [16] inspired framework. The relevance of a document to the current query is recursively calculated as the reward beginning from the starting query and continuing until the current query. This research is perhaps the first to employ reinforcement learning to tackle session search. Our experiments demonstrate that the proposed approach is highly effective and outperforms the best performing TREC 2011 and 2012 session search systems.

The remainder of this paper is organized as follows. Section 2 analyzes query change and summarizes policies for the user agent. Section 3 proposes policies for the search engine agent. Section 4 elaborates the query change retrieval model. Section 5 discusses how to handle duplicated queries. Section 6 evaluates our approach, followed by a discussion in Section 7. Section 8 presents the related work and Section 9 concludes the paper.

Table 2: Evidence that query change Δq appears in previous search results D_{i-1}.

	# in TREC'11		# in TREC'12	
	$\in D_{i-1}$	$\notin D_{i-1}$	$\in D_{i-1}$	$\notin D_{i-1}$
q_{theme}	184	20	178	21
$+\Delta q$	80	124	97	102
$-\Delta q$	141	63	112	87
Total	204 adjacent query pairs, 76 sessions		199 adjacent query pairs, 98 sessions	

2. USER AGENT: QUERY CHANGE AS A FORM OF FEEDBACK

We define a *search session* $\mathcal{S} = \{\mathcal{Q}, \mathcal{D}, \mathcal{C}\}$ as a combination of a series of queries $\mathcal{Q} = \{q_1, ..., q_i, ..., q_n\}$, retrieved document sets $\mathcal{D} = \{D_1, ..., D_i, ..., D_n\}$, and clicked information $\mathcal{C} = \{C_1, .., C_i, ..., C_n\}$, where n is the number of queries in the session (i.e., the session length) and i indexes the queries. In TREC 2010-2012 Session tracks, each retrieved document set D_i contains the top 10 retrieval results $d_{i1}, ..., d_{i10}$ ranked in decreasing relevance for q_i. Each clicked data C_i contains the user-clicked documents, clicking order, and dwell time. For instance, for s6 q_6, *pocono mountains camelbeach hotel* (Table 1), C_6 tells us that the user clicked the 4^{th} ranked search result, followed by the 2^{nd}, with dwell time 15 seconds and 17 seconds, respectively.

TREC 2010-2012 Session Tracks aim to retrieve a list of documents for the current query, i.e. the last query q_n in a session, ordered in decreasing relevance. Without loss of specificity, we assume that any query between q_1 to q_n could be the last query. We therefore study the problem of retrieving relevant documents for q_i, given all previous queries q_1 to q_{i-1}, previous retrieval results D_1 to D_{i-1}, and previous clicked data C_1 to C_{i-1}.

We define *query change* Δq_i as the syntactic editing changes between two adjacent queries q_{i-1} and q_i:

$$\Delta q_i = q_i - q_{i-1}$$

q_i can be written as a combination of the shared portion between q_i and q_{i-1} and query change: $q_i = (q_i \cap q_{i-1}) + \Delta q_i$.

The query change Δq_i comes from two sources. First, the added terms, which we call *positive* Δq, are new terms that the user adds to the previous query. Second, the removed terms, which we call *negative* Δq, are terms that the user deletes from the previous query. For example, in Table 1 s37, 'US' and 'policy' are the added terms; while in s28, 'stock' and 'market' are the removed terms.

We call the common terms shared by two adjacent queries *theme terms* since they often represent the main topic of a session. For example, in Table 1 s37 the theme terms are "Merck lobby".[2]

We thus decompose a query into three parts as *theme terms*, *added terms*, and *removed terms* and write it as:

$$q_i = (q_i \cap q_{i-1}) + (+\Delta q_i) - (-\Delta q_i) = q_{theme} + (+\Delta q_i) - (-\Delta q_i)$$

where q_{theme} are the theme terms, $+\Delta q_i$ and $-\Delta q_i$ represent added terms and removed terms, respectively.

Our observations suggest that documents that have been examined by the user factor in deciding the next query change. We therefore propose the following important assumption between Δq_i, the query change between adjacent

[2]We perform K-stemming to all query terms. For instance, 'lobbists' and 'lobbying' are both stemmed to 'lobby'.

Table 3: User agent's policies and actions about a query term $t \in q_{i-1}$. (Refer to sessions shown in Table 1)

	user likes D_{i-1}				**user dislikes** D_{i-1}			
	user intention	user action	example	type	user intention	user action	example	type
$t \in D_{i-1}$	1. find more information about t	add new terms t' about t	s85 $q_1 \rightarrow q_2$	specification	5. remove the wrong terms	remove t	s28 $q_1 \rightarrow q_2$	generalization
$t \in D_{i-1}$	2. satisfied & move to the next information need	remove t & add new terms t' as new focus	s6 $q_8 \rightarrow q_9$	drift	6. not satisfied & move to the next information need	remove t & add new terms t'	s6 $q_6 \rightarrow q_7$	drift
$t \in D_{i-1}$	3. satisfied	keep t	theme term	no change	7. try different expression for t	slight change of t to t'	s85 $q_2 \rightarrow q_3$	slight change
$t \notin D_{i-1}$	4. inspired by terms t' in D_{i-1}	add terms t'' about t'	s37 $q_1 \rightarrow q_2$	specification	8. try different expression for t to get more documents for t	slight change of t to t'	s32 $q_1 \rightarrow q_2$	slight change

queries q_i and q_{i-1}, and D_{i-1}, the search results for q_{i-1}:

$$\Delta q_i \leftarrow D_{i-1}.$$

The assumption basically says that *previous search results decide query change*. In fact, previous search results D_{i-1} could influence query change Δq_i in quite complex ways. For instance, the added terms in s37 (Table 1) q_1 to q_2, are 'US' and 'policy'. D_1 contains several mentions of 'policy', such as "*A lobbyist who until 2004 worked as senior policy advisor to Canadian Prime Minister Stephen Harper was hired last month by Merck*". However, these 'policy'-related mentions are about "Canada policy" whereas the user adds "US policy" in q_2. This suggests that the user might have been inspired by 'policy' in D_1, however he preferred the policy in *US*, not in *Canada*. Therefore, instead of simply cutting and pasting identical terms from D_{i-1}, the user creates related terms to add for the next search.

In another example, s28 (Table 1) q_1, 'stock' and 'market' are frequent terms that are similar to stopwords. Documents in D_1 are hence all about them and totally ignore the theme terms "france world cup 98." In q_2, the user removes "stock market" to boost rankings for documents about the theme terms. In this case, removing terms is not only about generalization, but also about document re-ranking.

To provide a convincing foundation for our approach, we look for evidence to support our assumption. We investigate whether Δq_i (at least) appears in D_{i-1}. Table 2 shows how often theme terms, added terms, and removed terms are present in D_{i-1} for both TREC 2011 and 2012 datasets. Around 90% of the time theme terms occur in D_{i-1} and most removed terms (>60%) appear in D_{i-1}.[3] Added terms are new terms for the previous query q_{i-1}; we thus expect to see few occurrences of added terms in D_{i-1}. Surprisingly, however, more than a third of them appear in D_{i-1}. It suggests that it is quite probable that previous search results motivate the subsequent query change.

Table 3 summarizes various types of query changes into possible policies for the user agent. This table mainly serves as a guide for us to design the policies for the search engine agent. We do not perform a thorough user study to validate this table. However, we believe that it is a good representative of various search scenarios and can help design a good session search agent.

Along two dimensions, Table 3 summarizes the user agent's actions and possible policies. The dimensions are whether a previous query term $t \in q_{i-1}$ appears in previous search results D_{i-1} (the left most column) and whether the user likes D_{i-1} and the occurrence of t in D_{i-1} (the top most row). Combinations of the two dimensions yield 4 main cases (as in a contingency table) and 8 sub-cases. For each case, we identify four items: a rough guess of user intention, the user's actual action, an example, and the semantic exploration type for this action. For example, query change in s6 $q_8 \rightarrow q_9$, *pocono mountains chateau resort attractions* $\rightarrow$ *pocono mountains chateau resort getting to* can be interpreted as the following. Previous query term 'attractions' appears in D_{i-1} and the user likes the returned documents D_{i-1}. One possibility is that he is satisfied with what he reads and moves to the next information need. Therefore, the user removes 'attraction' and adds new terms "getting to" as the new query focus. This is a *drift* in search focus. (case 2 in Table 3)

We further group the cases in Table 3 by types of user actions, i.e., query change, and summarize them into:

- Theme terms (q_{theme}), terms that appear in both q_{i-1} and q_i. In fact, they often appear in many queries in a session. It implies a strong preference for those terms from the user. If they appear in D_{i-1}, it shows that the user favors them since the user issues them again in q_i. If they do not appear in D_{i-1}, the user still favors towards them and insists to include them in the new query. This corresponds to t in cases 1 and 3 in Table 3.
- Added terms ($+\Delta q$), terms that appear only in q_i, not in q_{i-1}. They indicate specification, destination of drifting, or destination of slight change. If they appear in D_{i-1}, for the sake of novelty [14], they will not be favored in D_i. If they do not appear in D_{i-1}, which means that they are novel and the user favors them now. This corresponds to t' in cases 1, 2, 6, 7, and 8, and t'' in case 4 in Table 3.
- Removed terms ($-\Delta q$), terms that appear only in q_{i-1}, not in q_i. They indicate generalization, source of drifting, and source of slight change. If they appear in D_{i-1}, removing them means that the user observes them and dislikes them. If they do not appear in D_{i-1}, the user still dislikes the terms since they are not in q_i anyway. This corresponds to t in cases 2, 5, 6, 7, and 8 in Table 3.

3. SEARCH ENGINE AGENT: STRATEGIES TO IMPROVE SEARCH

The search engine agent observes query change from the user agent and takes corresponding actions. For each type of query change, *theme terms*, *added terms*, and *removed terms*, we propose to adjust the term weights accordingly for better retrieval accuracy. The search engine agent's action include

[3]A third of query terms that do not appear in D_{i-1} are removed by the user.

Table 4: Search engine agent's policy. Actions are adjustments on the term weights. $\uparrow$ means increasing, $\downarrow$ means decreasing, and $\rightarrow$ means keeping the original term weight.

	$\in D_{i-1}$	action	Example
q_{theme}	Y	$\uparrow$	"pocono mountain" in s6
	N	$\uparrow$	"france world cup 98 reaction" in s28, $q_1 \rightarrow q_2$
$+\Delta q$	Y	$\downarrow$	'policy' in s37, $q_1 \rightarrow q_2$
	N	$\uparrow$	'US' in s37, $q_1 \rightarrow q_2$
$-\Delta q$	Y	$\downarrow$	'reaction' in s28, $q_2 \rightarrow q_3$
	N	$\rightarrow$	'legislation' in s32, $q_2 \rightarrow q_3$

increasing, *decreasing*, and *maintaining* the term weights. Based on the observed query change as well as whether the query terms appeared in the previous search results D_{i-1}, we can sense whether the user will favor the query terms in the current run of search. Table 4 illustrates the policies that we propose for the search engine agent.

As shown in Section 2, theme terms q_{theme} often appear in many queries in a session and there is a strong preference for them. Thus, we propose to increase the weights of theme terms no matter whether they appeared in D_{i-1} or not (rows 1 and 2 in Table 4). In the latter case, if a theme term was not found in D_{i-1} (top retrieval results), it is likely that the documents containing them were ranked low. Therefore, the weights of theme terms need to be raised to boost the rankings of those documents (row 2 in Table 4). However, since theme terms are topic words in a session, they could appear like stopwords within the session. To avoid biasing too much towards them, we lower their term weights proportionally to their numbers of occurrences in D_{i-1}.

For added terms $+\Delta q$, if they occurred in previous search results D_{i-1}, we propose to decrease their term weights for the sake of novelty [14]. For example, in s5 $q_1 \rightarrow q_2$, "pocono mountains"$\rightarrow$"pocono mountains park", the added term 'park' appeared in a document in D_5. If we use the original weight of 'park', this document might still be ranked high in D_2 and the user may dislike it since he read it before. We hence decrease added terms' weights if they are in D_{i-1} (row 3 in Table 4). On the other hand, if the added terms did not occur in D_{i-1}, they are the new search focus and we increase their term weights (row 4 in Table 4). In an interesting case (s37 $q_1 \rightarrow q_2$), part of $+\Delta q$, 'policy', occurred in D_1 whereas the other part, 'US', did not. To respect the user's preference, we increase the weight of 'US' while decreasing that of 'policy' to penalize documents about other 'polices' including "Canada policy".

For removed terms $-\Delta q$, if they appeared in D_{i-1}, their term weights are decreased since the user dislikes them by deleting them (row 5 in Table 4). For example, in s28 $q_2 \rightarrow q_3$, 'reaction' existed in D_2 and is removed in q_3. However, if the removed terms are not in D_{i-1}, we do not change their weights since they are already removed from q_i by the user (row 6 in Table 4).

In the sections below, we follow policies proposed for the search engine agent as shown in Table 4 and incorporate them into a novel query change retrieval model (QCM).

4. MODELING SESSION SEARCH

Markov Decision Process (MDP) [16, 28] models a state space S and an action space A. Its states $\mathcal{S} = \{s_1, s_2, ...\}$ change from one to another according to a transition model $T = P(s_{i+1}|s_i, a_i)$, which models the dynamics of the entire system. A policy $\pi(s) = a$ indicates that at a state s, what are the actions a can be taken by the agent. In session search, we employ *queries* as states. Particularly, we denote q as state, T as the transition model $P(q_i|q_{i-1}, a_{i-1})$, D as documents, and A as actions. Actions include keeping, adding, and removing query terms for the user agent and increasing, decreasing, and maintaining the term weights for the search engine agent.

In a MDP, each state is associated with a *reward function* R that indicates possible positive reward or negative loss that a state and an action may result. In session search, we consider the reward function to be the relevance function.

Reinforcement learning [16] offers general solutions to MDP and seeks for the best policy for an agent. Each policy has a value associated with the policy and denoted as $V_\pi(s)$, which is the expected long-term reward starting from state s and continuing with policy π from then on. In a MDP, it is believed that a future reward is not worth quite as much as a current reward and thus a discount factor $\gamma \in (0, 1)$ is applied to future rewards. By considering the discount factor, the value function starting from s_0 for a policy π can be written as: $V_\pi(s_0) = E_\pi[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + ...] = E_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_i)]$. The Bellman equation [16] describes the optimal value V^* for a state s in the long run and is often used to obtain the best value for a MDP:

$$V^*(s) = \max_a R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s')$$

where s' is the next state after s, $V^*(s)$ and $V^*(s')$ are the optimal values for s and s'.

For session search, we observe that the influence of previous queries and previous search results to the current queries, becomes weaker and weaker. The user's desire for novel documents also supports this argument. We hence propose to employ the reinforcement learning model backwards. That is, instead of discounting the future rewards, we discount the past rewards, i.e. the relevant documents that appeared in the previous search results.

We propose the query change retrieval model (QCM) as the following. We consider the task of retrieving relevant documents for q_i as ranking documents based on the reward, i.e., how relevant it is to q_i. Inspired by the Bellman equation, we model the relevance of a document d to the current query q_i as:

$$Score(q_i, d) = P(q_i|d) + \gamma \sum_a P(q_i|q_{i-1}, D_{i-1}, a) \max_{D_{i-1}} P(q_{i-1}|D_{i-1}) \quad (1)$$

which recursively calculates the reward starting from q_1 and continues with the search engine agent's policy until q_i. $\gamma \in (0, 1)$ is the discount factor, $\max_{D_{i-1}} P(q_{i-1}|D_{i-1})$ is the maximum of the past rewards, $P(q_i|d)$ is the current reward, and $P(q_i|q_{i-1}, D_{i-1}, a)$ is the query transition model.

The first component in Eq.1, $P(q_i|d)$, measures the relevance between q_i and a document d that is under evaluation. This component can be estimated by the Bayesian belief network model [27]: $P(q_i|d) = 1 - \prod_{t \in q_i} (1 - P(t|d))$, where $P(t|d)$ is calculated by the multinomial query generation language model with Dirichlet smoothing [33]: $P(t|d) = \frac{\#(t,d) + \mu P(t|C)}{|d| + \mu}$, where $\#(t, d)$ denotes the number of occurrences of term t in document d, $P(t|C)$ calculates the probability that t appears in corpus C based on Maximum Likelihood Estimation (MLE), $|d|$ is the document length, and μ is the Dirichlet smoothing parameter (set to 5000).

The remaining challenges of calculating Eq.1 include maximizing the reward function $\max_{D_{i-1}} P(q_{i-1}|D_{i-1})$ and estimating the transition model $P(q_i|q_{i-1}, D_{i-1}, a)$. They are described in Section 4.1 and Section 4.2, respectively.

4.1 Maximizing the Reward Function

When considering the past/future rewards, MDP uses only the optimal (the maximum possible) values from those past /future rewards. This is reflected in $\max_{D_{i-1}} P(q_{i-1}|D_{i-1})$ as part of Eq. 1.

Prior research [10, 22] suggests that *Satisfying (SAT)* clicks, i.e., clicked documents with dwell time longer than 30 seconds [10, 22], are probably the only ones that are effective at predicting user behaviors and relevance judgments. Since the user also skims snippets in search interactions, in this work, we consider both the top 10 returned snippets and SAT clicks as *effective previous search results* and denote them as D^e_{i-1}.

To obtain an maximum reward from all possible reward functions $P(q_{i-1}|d_{i-1})$, i.e., the text relevance of previous query q_{i-1} and all previous search results $d_{i-1} \in D_{i-1}$, we propose to generate a *maximum rewarding document*, denoted as d^*_{i-1}. We further propose that the candidates for the d^*_{i-1} should only be selected from the *effective previous search results* D^e_{i-1}. We define d^*_{i-1} as the document(s) that is the most relevant to q_{i-1}. To discover d^*_{i-1}, we first rank all the documents (either a snippet or a document) $d_{i-1} \in D^e_{i-1}$ by measuring the relevance between q_{i-1} and d_{i-1} as: $P(q_{i-1}|d_{i-1}) = 1 - \prod_{t \in q_{i-1}} \{1 - P(t|d_{i-1})\}$, where $P(t|d_{i-1})$ is calculated by MLE: $P(t|d_{i-1}) = \frac{\#(t,d_{i-1})}{|d_{i-1}|}$, $\#(t, d_{i-1})$ is the number of occurrences of term t in document d_{i-1}, and $|d_{i-1}|$ is the document length. We do not apply smoothing here since $P(t|d_{i-1})$ can be zero, i.e., $t \notin D^e_{i-1}$. In fact, we rely on this property in later calculation.

After ranking documents d_{i-1} in D^e_{i-1}, we generate d^*_{i-1} by the following options: (1) using the document with the largest $P(q_{i-1}|d_{i-1})$, (2) concatenating the top k documents in D^e_{i-1} with the largest $P(q_{i-1}|d_{i-1})$, or (3) concatenating all documents in D^e_{i-1}. Experiments show that option (1) works the best and we use this setting throughout the paper. For notation simplicity, we use D_{i-1} from now on to denote effective previous search results.

4.2 Estimating the Transition Model

The transition model indicated in Eq. 1 is $\sum_a P(q_i|q_{i-1}, D_{i-1}, a)$. It includes the probabilities of query transitions under various actions. We incorporate polices designed in Table 4 to calculate it.

Search engine agent performs actions based on user agent's actions. We need to identify user's actions, i.e. query change Δq before search engine takes actions. Particularly, we recognize Δq by the following procedure. First, we generate q_{theme} based on the Longest Common Subsequence (LCS) [11] in both q_{i-1} and q_i. A subsequence is a sequence that appears in two strings in the same relative order but is not necessarily continuous. The LCS can be the common prefix or the common suffix of the two queries; it can also consist of several discontinuous common parts from the two queries. Take s6 $q_6 \rightarrow q_7$ as an example: q_6="pocono mountains camelbeach hotel", q_7="pocono mountains chateau resort", q_{theme} = LCS(q_6, q_7) = "pocono mountains". Next, we recognize added terms $+\Delta q$ and removed terms $-\Delta q$. Generally, the terms that occur in the current query but not in the previous query constitute $+\Delta q$; while the terms occur in the previous query but not in the current query constitute $-\Delta q$. In the above example, $-\Delta q_7$ = "camelbeach hotel", and $+\Delta q_7$ = "chateau resort".

The search engine actions are *decreasing*, *increasing*, and *maintaining* term weights. According to Table 4 rows 3 and 5, we decrease a term's weight if the query change, either $+\Delta q$ or $-\Delta q$, occurred in the effective previous search results D_{i-1}. We propose to deduct term t's weight by $P(t|d)$, i.e. t's default contribution to the relevance score between q_i and the document under evaluation (denoted as d). Furthermore, since t already occurred in D_{i-1}, for the sake of novelty, we deduct more weight that is proportional to t's frequency in D_{i-1} such that the more frequently t occurred in D_{i-1}, the more heavily t's weight is deducted from the current query q_i and d. We formulate this weight deduction for a term $t \in +\Delta q$ or $t \in -\Delta q$ as:

$$\log P_{new}(t|d) = (1 - P(t|d^*_{i-1})) \log P(t|d) \tag{2}$$

where d^*_{i-1} denotes the maximum rewarded document, d is the document under evaluation, and $P(t|d)$ is calculated by MLE. We apply the log function to avoid numeric underflow.

We notice that Eq. 2 has an interesting connection with the Kullback-Leibler divergence (KL divergence) [33]:

$$\begin{aligned} - P(t|d^*_{i-1}) \log P(t|d) &= P\left(t|d^*_{i-1}\right) \log \frac{1}{P\left(t|d\right)} \\ &\overset{rank}{=} P\left(t|d^*_{i-1}\right) \log \frac{P\left(t|d^*_{i-1}\right)}{P\left(t|d\right)} \\ &\overset{rank}{=} KLD_t\left(\theta_{d^*_{i-1}}||\theta_d\right) \end{aligned} \tag{3}$$

where $KLD_t\left(\theta_{d^*_{i-1}}||\theta_d\right)$ denotes the contribution of term t to the KL divergence between two documents' language models $\theta_{d^*_{i-1}}$ and θ_d. In Eq. 3, the larger the divergence between $\theta_{d^*_{i-1}}$ and θ_d, the more novel document d is compared to D_{i-1}, and the less deduction to the relevance score. In this sense, Eq. 2 models novelty for the added terms and the removed terms during a query transition.

According to Table 4 row 4, we increase a term's weight if it is an added term and did not occur in D_{i-1}. We propose to raise the term weight proportional to its inverse document frequency (*idf*). This is to make sure that while increasing a preferred term's weight, we avoid increasing its weight too much if it is a common term in many documents. We formulate this weight increase for a novel added term t ($t \in +\Delta q$ and $t \notin D_{i-1}$) as:

$$\log P_{new}(t|d) = (1 + idf(t)) \log P(t|d) \tag{4}$$

where $idf(t)$ is the inverse document frequency of t in Corpus C and $P(t|d)$ is calculated by MLE. Note that this term weight adjustment is in a form of *tf-idf*.

The increasing in term weights also applies to theme terms, which corresponds to rows 1 and 2 in Table 4. Theme terms repeatedly appear in a session, which implies the importance of them. Similar to the novel added terms, we should avoid increasing their weights too much. We could discount the increment proportional to *idf*. However, theme terms are topical/common terms within a session, not necessarily common terms in the entire corpus. Therefore, *idf* may not be applicable here. We hence employ the negation of the number of occurrences of t in previous maximum rewarding document, $1 - P\left(t|d^*_{i-1}\right)$, to substitute *idf*. We formulate this weight increase for a theme term $t \in q_{theme}$ as:

$$\log P_{new}(t|d) = \left(1 + \left(1 - P\left(t|d^*_{i-1}\right)\right)\right) \log P(t|d) \qquad (5)$$

where d^*_{i-1} denotes the maximum rewarded document and $P(t|d)$ is calculated by MLE.

For removed terms that did not appear in D_{i-1} (Table 4 row 6), the search agent does not change their term weights.

By considering all possible cases for the transition model as defined in Eq. 1, the relevance score between the current query q_i and a document d is represented as below:

$$\begin{aligned} Score(q_i, d) &= \log P(q_i|d) + \alpha \sum_{t \in q_{theme}} \left[1 - P\left(t|d^*_{i-1}\right)\right] \log P\left(t|d\right) \\ &- \beta \sum_{\substack{t \in +\Delta q \\ t \in d^*_{i-1}}} P\left(t|d^*_{i-1}\right) \log P\left(t|d\right) + \epsilon \sum_{\substack{t \in +\Delta q \\ t \notin d^*_{i-1}}} idf(t) \log P(t|d) \\ &- \delta \sum_{t \in -\Delta q} P\left(t|d^*_{i-1}\right) \log P\left(t|d\right) \end{aligned} \qquad (6)$$

where α, β, ϵ, and δ are parameters for each types of actions. Note that we apply different parameters β and δ on $+\Delta q$ and $-\Delta q$, since added terms and removed terms may affect the retrieval differently. We report the parameter selection in Section 6.

4.3 Scoring the Entire Session

It is worth noting that Eq. 6 is valid only when $i > 1$. When $i = 1$, there is no previous result for q_1. We thus use

$$Score(q_1, d) = \log P(q_1|d) \qquad (7)$$

as a base case. $P(q_1|d)$ is calculated by Eq. 4.

Using Eq. 7 as the base case for the recursive function described in Eq. 1, we obtain the overall document relevance score $Score_{session}(q_n, d)$ for a session that starts at q_1 and ends at q_n by considering all queries in the session:

$$\begin{aligned} &Score_{session}(q_n, d) = Score(q_n, d) + \gamma Score_{session}(q_{n-1}, d) \\ &= Score(q_n, d) + \gamma \left[Score(q_{n-1}, d) + \gamma Score_{session}(q_{n-2}, d)\right] \\ &= \sum_{i=1}^{n} \gamma^{n-i} Score(q_i, d) \end{aligned} \qquad (8)$$

where $q_1, q_2, \cdots, q_n$ are in the same session, and $\gamma \in (0, 1)$ is the discount factor. Eq. 8 provides a form of aggregation over the relevance functions of all the queries in a session.

5. DUPLICATED QUERIES

Duplicated queries sometimes occur in a search session. Prior work shows that removing duplicated queries could effectively boost the search accuracy [8, 19]. Duplicated queries often occur when a user is frustrated by irrelevant documents in search results and comes back to one of the previous queries for a fresh start. For example, in s6 (Table 1), q_2 and q_4 are duplicates and both search for *pocono mountains pennsylvania hotels*. The query between them is q_3: *pocono mountains pennsylvania things to do*. It suggests that the user might dislike the search results for q_3 and he returns to q_2 to search again ($q_2 = q_4$).

To detect query duplicates, we first remove punctuations and white spaces in queries, then apply stemming on them.

Table 5: Dataset statistics for TREC 2011 and 2012 Session.

	2011	2012		2011	2012
#topics	62	48	#queries/session	3.68	3.03
#sessions	76	98	#sessions/topic	1.23	2.04
#queries	280	297	#pages judged	19,413	17,861
#dups	16	5	#sessions w/o rel. docs	2	4

Next we determine exact string matches between every query pair. The exactly matched query pairs are identified as duplicated queries.

Since the user may dislike the queries and their corresponding search results between two duplicated queries, we propose to eliminate from the MDP the undesired queries and their interactions. We achieve this by setting the discount factor to zero for any interaction between two duplicated queries as well as that for the earlier query in the two. The new discount factor γ' can be calculated as:

$$\gamma'_i = \begin{cases} 0 & \{i | i \in [j, k), \exists q_j = q_k, j < k)\} \\ \gamma_i & \text{otherwise} \end{cases} \qquad (9)$$

where γ_i is the original discount factor for the i^{th} query, γ'_i is the updated discount factor for the i^{th} query after deduplication.

For the above example s6, the effects from q_2 and q_3 on the session are eliminated. The entire session is now equivalent to $q_1, q_4, q_5, ..., q_{11}$.

6. EVALUATION

The evaluation datasets are from TREC 2011 and 2012 Session tracks [18, 19]. Table 5 lists the statistics about these two datasets. Each search session includes several queries and the corresponding search results. The users (NIST assessors) were given a topic description about information needs before they searched. For example, s85 (Table 1) are related to topic 43 "When is scientific glass blowing used? What are the purposes? What organizations do scientific glass blowing?" Multiple sessions can relate to the same topic. The search engine used to create the sessions was Yahoo! BOSS. The top 10 returned documents were shown to the users and they clicked documents that were interesting to them and interacted with the system. We use TREC's official ground truth and official evaluation metrics nDCG@10 and MAP.

The corpus used in this evaluation is ClueWeb09 Category B collection (CatB).[4] CatB contains the first 50 million English pages crawled from the Web during January to February 2009. We filter out the spam documents by removing documents whose WateQCMoo's "GroupX" spam ranking scores [6] are less than 70.

We compare the following systems in this evaluation:

- *Lemur*: Directly submitting the current query q_n (with punctuations removed) to the Lemur search engine [21] (language modeling + Dirichlet smoothing) and obtain the returned documents.
- *TREC best* : The top TREC system as reported by NIST [13, 14]. It adopts a query generation model with relevance feedback and handles document novelty. CatB was used in their TREC submissions. *This system is used as the baseline system in this evaluation.*
- *Nugget*: Another top TREC 2012 session search system groups semantically coherent query terms as *nuggets* and

[4]http://lemurproject.org/clueweb09/.

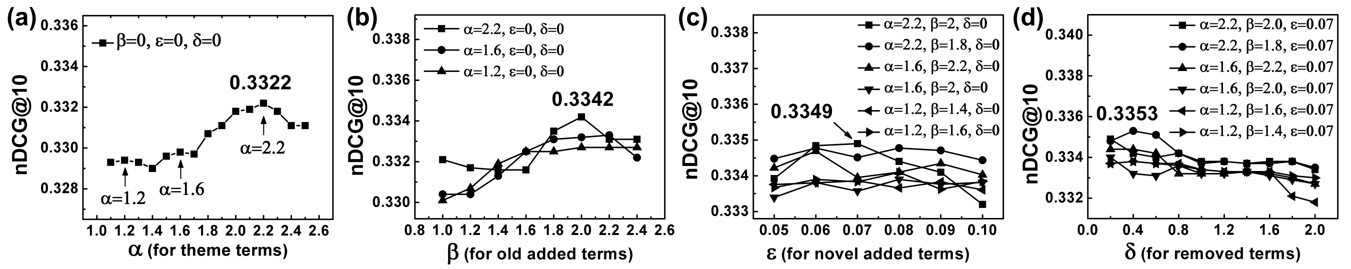

Figure 2: nDCG@10 for TREC 2012 against the parameters. (a), (b), (c), and (d) are about α, β, ϵ, and δ respectively.

Table 6: nDCG@10, MAP, and their improvements over the baseline (%chg) for TREC 2012 sessions. The runs are sorted by nDCG@10. A statistical significant improvement over the baseline is indicated with a † at $p < 0.05$ level.

Approach	nDCG@10	%chg	MAP	%chg
Lemur	0.2474	-21.54%	0.1274	-18.28%
TREC median	0.2608	-17.29%	0.1440	-7.63%
Nugget	0.3021	−4.19%	0.1490	-4.43%
TREC best	0.3221	0.00%	0.1559	0.00%
QCM	**0.3353**	**4.10%**†	0.1529	-1.92%
QCM+Dup	**0.3368**	**4.56%**†	0.1537	-1.41%

Table 7: nDCG@10, MAP, and their improvements over the baseline (%chg) for TREC 2011 sessions. The runs are sorted by nDCG@10. A statistical significant improvement over the baseline is indicated with a † at $p < 0.05$ level.

Approach	nDCG@10	%chg	MAP	%chg
Lemur	0.3378	-23.38%	0.1118	-25.86%
TREC median	0.3544	-19.62%	0.1143	-24.20%
TREC best	0.4409	0.00%	0.1508	0.00%
QCM	**0.4728**	**7.24%**†	**0.1713**	**13.59%**†
QCM+Dup	**0.4821**	**9.34%**†	**0.1714**	**13.66%**†
Nugget	0.4836	9.68%†	0.1724	14.32%†

creates structured Lemur queries [8]. We re-implement and apply it on both TREC 2011 and 2012.

- *TREC median*: The median TREC system as reported by NIST [18, 19].
- *QCM*: The proposed query change retrieval model.
- *QCM + De-Duplicate (Dup)*: The proposed query change retrieval model with duplicated queries removed.

6.1 Search Accuracy

Table 6 and Table 7 demonstrate search accuracy for all systems under comparison for TREC 2012 and TREC 2011, respectively. The evaluation metrics are nDCG@10 and MAP, the same as in the official TREC evaluations. *TREC best* serves as the baseline.

Table 6 shows that the proposed QCM approach outperforms the best TREC 2012 system on nDCG@10 by 4.1%, which is statistically significant (one sided t-test, $p = 0.05$). The search accuracy is further improved by 0.46% through removing the duplicated queries. The experimental results strongly suggest that our approach is highly effective.

Table 7 shows that for TREC 2011, our approach again outperforms the baseline by a statistically significant 7.24% (one sided t-test, $p = 0.05$) and achieves a further improvement of 9.34% by the QCM+Dup approach. For TREC 2011, the performance gain by performing de-dup is 2.1%, which is bigger than that for TREC 2012 (0.46%). The reason is probably because that TREC 2012 only has 5 duplicated queries while TREC 2011 has 16 (shown in Table 5). However, the best approach for TREC 2011 is the *nugget* approach, which is slightly better than QCM+Dup.

Table 5 illustrates the dataset differences between TREC 2011 and 2012. These differences may affect search accuracy. The average number of sessions per topic is 2.04 in 2012, that is more than that in 2011 (1.23). Moreover, on average, TREC 2012 sessions contain less queries per session (3.03) than 2011 (3.68). As a result, the shorter sessions in 2012 may make the search task more difficult than 2011 since less information are provided by previous interactions. Another difference is that 2012 sessions have fewer (sometimes even none) relevant documents than 2011 sessions in CatB ground truth. It unavoidably hurts the performance for any retrieval system. Generally, we observe lower search accuracy in 2012 (Table 6) than in 2011 (Table 7).

6.2 Parameter Tuning

We investigate good values for parameters in Eq. 6. A supervised learning-to-rank method should be able to find the optimal values for those parameters. However, in this paper, we take a step-by-step parameter tuning procedure and leave the supervised learning method as future work.

We add each component, i.e., theme terms, added terms, and removed terms, one by one into Eq. 6. The tuning is performed for QCM only and the parameters are shared between QCM and QCM+Dup.

First, we plot nDCG@10 against α while setting other parameters to 0 (Figure 2(a)). α represents the parameter for theme terms. α ranges over [1.1, 2.5] by an interval of 0.1. We notice that nDCG@10 reaches its maximum at $\alpha = 2.2$. We find 2 other local maximums at 1.6 and 1.2 for α.

Next, we fix α to the above values and plot nDCG@10 against β (Figure 2(b)). β is the parameter for added terms that appeared in effective previous search results; we call them old added terms. β ranges over [1.0, 2.4] by an interval of 0.2. We choose the top 2 local values from each curve and pick 6 combinations for (α, β) as indicated in Figure 2(c).

Then, we fix (α, β) and plot nDCG@10 against ϵ (Figure 2(c)). ϵ is the parameter for added terms that did not appear in effective previous search results; we call them novel added terms. ϵ ranges over [0.05, 0.1] by an interval of 0.01. All the curves show similar trends and reach the highest nDCG@10 at around 0.07. We hence fix ϵ to 0.07.

Finally, we plot nDCG@10 against δ (Figure 2(d)) with the parameter combinations that we discover eerlier. Eventually, nDCG@10 reaches its peak 0.3353 at $\alpha = 2.2, \beta = 1.8, \epsilon = 0.07$, and $\delta = 0.4$. We apply this set of parameters to both QCM and QCM+Dup.

As we can see, α, β, and δ are much larger than ϵ. This is because that in Eq. 4, $idf(t) = \log(N/n_d)$ falls in the range of $[1, 10]$, while in Eq. 2 and Eq. 5, $P(t|d^*_{i-1})$ falls

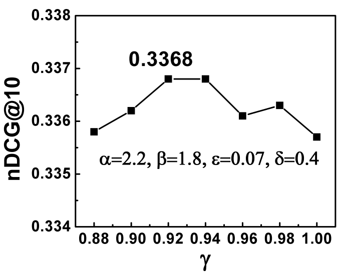

Figure 3: Discount factor γ.

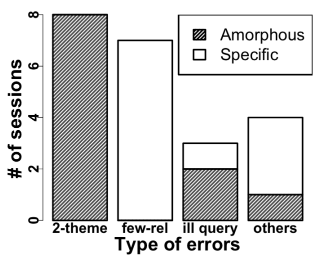

Figure 4: Error types.

Table 8: Aggregation schemes.

Approach		q_n	$q_i (i \in [1, n-1])$
Query change model		1	γ^{n-i}
Aggregation Scheme	Uniform	$\lambda_n = 1$	$\lambda_i = 1$
	PvC	$\lambda_n = 1 - \lambda_p$	$\lambda_i = \lambda_p$
	Distance-based	$\lambda_n = 1 - \lambda_p$	$\lambda_i = \frac{\lambda_p}{n-i}$

Table 9: nDCG@10 for various aggregation schemes. λ_p is 0.4 in PvC. γ is 0.92 in QCM and QCM+Dup. TREC 2012 best serves as the baseline. A significant improvement over the baseline is indicated with a † at $p < 0.05$ level.

Aggregation Scheme	TREC 2011		TREC 2012	
	nDCG@10	%chg	nDCG@10	%chg
Distance-based	0.4431	-2.40%	0.3111	−3.42%
TREC best	0.4540	0.00%	0.3221	0.00%
Uniform	0.4626	1.89%†	0.3316	2.95%†
PvC	0.4713	3.81%†	0.3351	4.04%†
QCM	**0.4728**	**4.14%†**	**0.3353**	**4.10%†**
QCM+Dup	**0.4821**	**6.19%†**	**0.3368**	**4.56%†**

in the range of [0,0.1]. Therefore, the values of ϵ are two magnitudes less than that for the other parameters. Among α, β, and δ, we find that α and β are larger than δ, which implies that theme terms and added terms may play more important roles in session search than removed terms.

6.3 Aggregation for the Entire Session

QCM proposes an effective way to aggregate all queries in a session as in Eq.8. We compare how effective it is to prior query aggregation methods. A query aggregation scheme can be represented as: $Score(session, d) = \sum_{i=1}^{n} \lambda_i \cdot Score(q_i, d)$, where $Score(q_i, d)$ is the relevance scoring function of d and q_i and λ_i is the query weight for q_i.

[8] proposed several aggregation schemes for TREC 2012 Session track. The schemes are: *uniform* (all queries are equally weighted), *previous vs. current* (known as PvC; all previous queries are discounted by λ_p, while the current query uses a complementary and higher coefficient $(1 - \lambda_p)$, and *distance-based* (previous queries are discounted based on a reciprocal function of queries' positions in the session).

We express various query aggregation schemes in terms of the discount factor γ in order to compare them with our approach. From Table 8, we find that QCM degenerates to uniform when $\gamma = 1$. Previous queries in PvC and Distance-based schemes are also discounted as they are in QCM, but with different decay functions.

The search accuracy for different aggregation schemes are compared in Table 9. QCM performs the best for both TREC 2011 and 2012. The PvC scheme is the second best scheme, which confirms what is reported in [8]. The Distance-based scheme gives the worst performance.

We explore the best discount factor γ for QCM over $(0, 1)$ by an interval of 0.02. Figure 3 illustrates the relationship between nDCG@10 and γ. nDCG@10 climbs to its peak 0.3368 when $\gamma = 0.92$. The result suggests that a good discount factor γ is very close to 1, implying that previous queries contribute to the overall search accuracy nearly the same as the last query. It suggests that in QCM, a discount between two adjacent queries should be mild.

7. DISCUSSION

7.1 Advantages of Our Approach

A main contribution of our approach is that we treat a search session as a continuous process by studying changes among query transitions and modeling the dynamics in the entire session. Through the reinforcement learning style framework, our system provides the best aggregation scheme for all queries in a session (Table 9). This allows us to better handle sessions that demonstrate evolution and exploration in nature than most existing systems do. On the contrary, for sessions that are clear in search goals and lack of a exploratory nature, the advantage of our system over other systems looks less significant.

This can be seen in Table 10, which illustrates the search accuracy for the TREC best, Nugget, and our system for various classes of sessions. The TREC best is used as the baseline and we also show the percentile improvement over it in Table 10. TREC 2012 sessions were created by considering and hence can be classified into two facets: search target (factual or intellectual) and goal quality (specific/good or amorphous/ill) [19]. Table 10 shows that QCM works very well for all classes of sessions. Specifically, QCM works even better, i.e. outperforms the TREC best even more significantly, for sessions that search for intellectual targets as well as sessions that search with amorphous goals. In our opinion, this is due to that intellectual tasks produce new ideas or new findings (e.g. learn about a topic or make decision based on the information collected so far) while searching. Both intellectual and amorphous sessions rely more on previous search results. Thus, users reformulate queries based more on what they have retrieved, not the vague information need. This is a scenario where our approach is good at since we employ previous search results to guide the search engine's action. For specific and factual sessions, users are clearer in search goals, query changes may come less from the previous search results. In summary, our good performance on both intellectual task and amorphous task is consistent with our efforts of modeling query changes.

Moreover, we benefit from term-level manipulation in various aspects in our system. The first aspect is novelty. Both the TREC best system and our system handle novelty in a session. The TREC best system only deals with novelty at the document level. They consider documents that have been examined by the user in a previous interaction not novel and the rest are novel [14]. That is, they determine novelty purely based on document identification number, not the actual content. Through studying whether query terms appeared in previous search results, our approach evaluates and models novelty at the term level (or concept level), which we believe better represents the evolving information needs in a session. The second aspect is query handling. The Nugget approach [8] treats queries at the phrase level and formulates structured queries based on phrase-like *nuggets*. The approach achieves good performance, espe-

Table 10: nDCG@10 for different classes of sessions in TREC 2012.

	Intellectual	%chg	Amorphous	%chg	Specific	%chg	Factual	%chg
TREC best	0.3369	0.00%	0.3495	0.00%	0.3007	0.00%	0.3138	0.00%
Nugget	0.3305	-1.90%	0.3397	-2.80%	0.2736	-9.01%	0.2871	-8.51%
QCM	0.3870	14.87%	0.3689	5.55%	0.3091	2.79%	0.3066	-2.29%
QCM+DUP	0.3900	15.76%	0.3692	5.64%	0.3114	3.56%	0.3072	-2.10%

cially for TREC 2011. However, due to complexity in natural language, nugget detection is sensitive to dataset and the approach's performance is not quite as stable as ours on different datasets.

Lastly, our system benefits from trusting the user. Our approach does not use too much materials from other resources such as anchor texts, meta data, or click orders, as many other approaches do [8, 26]. We believe that the most direct and valuable feedback is the next query that the user enters. In this work, we manage to capture the query change and investigate the reasons behind it. We use ourselves as users to summarize possible human users' reasoning and actions. More detailed analysis about user intent might be useful for researchers to understand web users, however, it might be overwhelming (too fine-grained or too much semantics) for a search engine that essentially only counts words.

7.2 Error Analysis & Future Work

Our system retrieves nothing for 22 out of 98 sessions in TREC 2012. To analyze the reason for the poor performance for those sessions, we study their topic descriptions, queries, and ground truth documents. We summarize the types of errors as "two theme concepts", "ill query", "few relevant documents", and others. Figure 4 shows how many sessions that we fail to retrieve under each error type.

We call the first type of errors "two theme concepts". It comes from a type of session where the information need cover more than one concepts. For instance, s17 and s18 share the the same topic "... To what extent can decisions and policies of the Indian government be credited with these wins?". Queries in s17 and s18 ask about both concepts "indian politics' and "miss universe". Unfortunately, very few relevant documents about both theme concepts exist in the corpus. The retrieved documents are about either concept, but none is about both. Eight sessions belong to this type. As future work, we can improve our system by incorporating structures in queries, and enable more sophisticated operators such as Boolean and proximity search.

The second type of errors is "ill query", where in such sessions, queries themselves are ill formulated and do not well-represent the information needs indicated in the given topic. A common mistake is that the user misses some sub-information need. For example, the topic for s16 is: "... you want to reduce the use of air conditioning in your house ... you could protect the roof being overly hot due to sun exposure... Find information of ... how it could be done." A good query for this topic should include *roof* and *air conditioning*. However, the queries that the user issued for s60, "reduce airconditioning" and "attic insulation air conditioning costs", do not mention *roof* at all. Because of this ill query formulation, our system yields no relevant documents for s60. On the other hand, for s59, which shares the same information need with s60, our system achieves a nDCG@10 of 0.48 simply because s59 queries "cool roof". It suggests that ill queries mislead the search engine and yield poor retrieval performance. Four sessions belong to this type. As future work, we will explore effective query suggestion by studying sessions that share the same topic.

The third type of errors is "too few relevant documents". For sessions with too few relevant documents in the ground truth, our system do not perform well. In total 2,573 relevant documents exist in CatB for all 48 TREC 2012 topics; on average 53.6 relevant documents per topic. However, topics 10, 45, 47 and 48, each has no more than 2 relevant documents and topic 47 (s92 to s95) has no relevant document in CatB (Table 5). This problem could be reduced if we index the entire ClubWeb09 CatA collection.

Figure 4 also indicates in which classes of sessions these errors lie. We find that all "two theme concept" errors belong to sessions created with amorphous goals while all "too few relevant documents" errors belong to those with specific goals. Moreover, "ill queries" tend to occur more in sessions with amorphous goals. Note that "ill query" and "few relevant documents" are errors due to either the user or the data. There might not be much room for our system to improve over them. However, "two theme concepts" is where our system can certainly make further improvements.

8. RELATED WORK

Session search is a challenging IR task [4, 8, 13, 14, 25, 32]. Existing approaches investigate session search from various aspects such as semantic meanings of search tasks [23], document novelty [14], and phrase structure in queries [8]. The best TREC system [13, 14] employs an adaptive browsing model by considering both relevance and novelty; however it does not demonstrate improvement by handling novelty. In this paper, we successfully model query and document novelty by investigating the relationship between query change and previous search results. Moreover, our analysis on query change does not require knowledge of semantic types for the sessions as [23] proposed.

Our proposed work is perhaps the most similar to the problem of query formulation [1, 9, 12, 24] and query suggestion [29]. [12] showed that certain query changes such as adding/removing words, word substitution, acronym expansion, and spelling correction are more likely to cause clicks, especially on higher ranked results. The finding is generally consistent with our view of query change. However, their work only emphasized on understanding of query changes, without showing how to apply it to help session search. [24] examined the relationship between task types and how users change queries. They classified query changes by semantic types: Generalization, Specialization, Word Substitution, Repeat, and New. Similar to [12], however, [24] stopped at understanding query changes and didn't apply their findings to help session search. This probably makes us the first to utilize query changes in actual retrieval. [1] derived query-flow graph, a graph representation of user query behavior, from user query logs. The approach detected query chains in the graph and recommended queries based on maximum weights, random walk, or just the previous query. Other mining approaches [1, 29] identify the importance of query change in sessions; however, they require the luxury of large user query logs.

This research is perhaps the first to employ reinforcement learning to solve the Markov Decision Process demonstrated in session search. Reinforcement learning is complex and difficult to solve. Its solutions include model-based approaches and model-free approaches [16]. The former learn the transition model and the reward function for every possible states and actions and mainly employ MLE to estimate the model parameters. Others also use matrix inversion or *linear programming* to solve the Bellman equation. It works well when state spaces are small. However, in our case, the state space is large since we use natural language queries as the states; hence we could not easily apply model-based approaches in practice. In this work, we effectively reduce the search space by summarizing users' and search engine's actions into a few types and employ a model-free approach to learn value functions directly.

9. CONCLUSION

This paper presents a novel session search approach (QCM) by utilizing query change and modeling the dynamic of the entire session as a Markov Decision Process. We assume that query change is an important form of feedback. Based on this assumption, through studying editing changes between adjacent queries, and their relationship with previous retrieved documents, we propose corresponding search engine actions to handle individual term weights for both the query and the document. In a reinforcement learning inspired framework, we incorporate various ingredients present in session search, such as query changes, satisfactory clicks, desire for document novelty, and duplicated queries. The proposed framework provides a theoretically sound and general foundation that allows more novel features to be incorporated. Experiments on both TREC 2011 and 2012 Session tracks show that our approach is highly effective and outperforms the best session search systems in TREC. This research is perhaps the first to employ reinforcement learning in session search. Our MDP view of modeling session search can potentially benefit a wide range of IR tasks.

10. ACKNOWLEDGMENT

This research was supported by NSF grant 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.

11. REFERENCES

[1] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *CIKM '08*.

[2] I. Bordino, C. Castillo, D. Donato, and A. Gionis. Query similarity by projecting the query-flow graph. In *SIGIR '10*.

[3] P. Bruza, R. McArthur, and S. Dennis. Interactive internet search: keyword, directory and query reformulation mechanisms compared. In *SIGIR '00*.

[4] B. Carterette, E. Kanoulas, and E. Yilmaz. Simulating simple user behavior for system effectiveness evaluation. In *CIKM '11*.

[5] M.-A. Cartright, R. W. White, and E. Horvitz. Intentions and attention in exploratory health search. In *SIGIR'11*.

[6] 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), Oct. 2011.

[7] D. Guan and H. Yang. Increasing stability of result organization for session search. In *ECIR '13*.

[8] D. Guan, H. Yang, and N. Goharian. Effective structured query formulation for session search. In *TREC '12*.

[9] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *SIGIR '08*.

[10] Q. Guo and E. Agichtein. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *SIGIR '10*.

[11] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24(4), Oct. 1977.

[12] J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *CIKM '09*.

[13] J. Jiang, S. Han, J. Wu, and D. He. Pitt at trec 2011 session track. In *TREC '11*.

[14] J. Jiang, D. He, and S. Han. Pitt at trec 2012 session track. In *TREC '12*.

[15] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM '08*.

[16] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *J. Artif. Int. Res.*, 4(1), May 1996.

[17] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *Knowl. Eng. Rev.*, 18(1), Jan. 2003.

[18] E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2011 session track. In *TREC'11*.

[19] E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2012 session track. In *TREC'12*.

[20] E. Kanoulas, P. D. Clough, B. Carterette, and M. Sanderson. Session track at trec 2010. In *TREC'10*.

[21] Lemur Search Engine. `http://www.lemurproject.org/`.

[22] C. Liu, N. J. Belkin, and M. J. Cole. Personalization of search results using interaction behaviors in search sessions. In *SIGIR '12*.

[23] C. Liu, M. Cole, E. Baik, and J. N. Belkin. Rutgers at the trec 2012 session track. In *TREC'12*.

[24] C. Liu, J. Gwizdka, J. Liu, T. Xu, and N. J. Belkin. Analysis and evaluation of query reformulations in different task types. In *ASIST '10*.

[25] J. Liu and N. J. Belkin. Personalizing information retrieval for multi-session tasks: the roles of task stage and task type. In *SIGIR '10*.

[26] A. M-Dyaa, K. Udo, N. Nikolaos, N. Brendan, L. Deirdre, and F. Maria. University of essex at the trec 2011 session track. In *TREC '11*.

[27] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage.*, 40(5), Sept. 2004.

[28] S. P. Singh. Learning to solve markovian decision processes. Technical report, Amherst, MA, USA, 1993.

[29] Y. Song and L.-w. He. Optimal rare query suggestion with implicit user feedback. In *WWW '10*.

[30] Y. Song, D. Zhou, and L.-w. He. Query suggestion by constructing term-transition graphs. In *WSDM '12*.

[31] J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: modeling queries with variation in user intent. In *SIGIR '08*.

[32] R. W. White, I. Ruthven, J. M. Jose, and C. J. V. Rijsbergen. Evaluating implicit feedback models using searcher simulations. *ACM Trans. Inf. Syst.*, 23(3), July 2005.

[33] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, Apr. 2004.