Query Aggregation in Session Search

Dongyi Guan
Department of Computer Science
Georgetown University
37th and O Street, NW
Washington, DC, 20057
dg372@georgetown.edu

Hui Yang
Department of Computer Science
Georgetown University
37th and O Street, NW
Washington, DC, 20057
huiyang@cs.georgetown.edu

ABSTRACT
Session search retrieves documents for a sequence of queries in a session. Prior research demonstrated that query aggregation is an effective technique for session search. This paper proposes a novel query aggregation scheme based on the discount factor in reinforcement learning. Moreover, we compare various query aggregation schemes and investigate the best scheme for aggregating queries in session search. Evaluation conducted over TREC 2011 and 2012 shows that the proposed scheme works the best and outperforms the TREC best system as well as learned weights by learning to rank.

1. INTRODUCTION
Complex search tasks often require more than one query to accomplish the tasks. These queries form a search session and work as a whole to fulfill one information need. Session search is the Information Retrieval (IR) task that retrieves documents for a session. It differs from ordinary search tasks in the need of considering not only a single query but the entire session [2, 4, 5, 7, 12, 14].

The TREC (Text REtrieval Conference) 2010-2012 Session tracks [9, 10, 11] study session search with a focus on the “current query”, which retrieves relevant documents for the current/last query \( q_n \) in a session with \( n \) queries, given the previous queries and previous interactions.\(^1\) The users (NIST assessors) search for a given information need such as *When is scientific glass blowing used? What are the purposes? What organizations do scientific glass blowing?* (s85, Table 1) The users interact with a search engine (Yahoo! BOSS in this case) and create queries in the sessions. Each query triggers and represents an interaction between the search engine and the user. The top 10 returned documents are shown to the users and the user clicks the interesting ones to read. TREC collects the queries, retrieved documents, and interaction data such as clicked documents. Table 1 shows queries from the TREC 2012 Session track.

\(^1\) 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.

<table>
<thead>
<tr>
<th>Session</th>
<th>Query 1</th>
<th>Query 2</th>
<th>Query 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>q1.glass blowing</td>
<td>q2.glass blowing science</td>
<td>q3.scientific glass blowing</td>
</tr>
<tr>
<td>47</td>
<td>q1.pseudocyesis</td>
<td>q2.pseudocyesis epidemiology</td>
<td>q3.pseudocyesis history</td>
</tr>
</tbody>
</table>

Even though the TREC Session task is to retrieve documents only for the current/last query, the document relevance is related to the entire session. It makes sense to count in all queries in a session for better search accuracy. For example, s47 contains three queries “pseudocyesis”, “pseudocyesis epidemiology”, and “pseudocyesis history”. They are all about the topic “pseudocyesis”. Prior research demonstrated that aggregating the queries in a session when ranking the documents for the last query is an effective technique for session search. Generally, query aggregation [4, 5, 1] can bring a gain around 15% ~ 20% in nDCG@10 [3]. In this paper, we investigate this effective technique, query aggregation, and its impact on session search.

We test several hypotheses on how to effectively aggregate queries for session search. The first hypothesis is that every query in a session is equally important since they are all about the same topic. This hypothesis suggests equal weight for all queries in a session when combining them.

The second hypothesis is that the current/last query is the most concerned by the user; hence it should be treated differently from all other queries. For instance, in s47, we assume that the user needs more information about “pseudocyesis history” rather than “pseudocyesis epidemiology” since the search ends at “pseudocyesis history”. This hypothesis suggests that we should assign a higher weight to the last query than to the previous ones.

The third hypothesis is that the importance of a query is related to the relative position to the current/last query. In fact, the closer to the current query, the more important a query is. The intuition is that earlier queries probably diverge further from the goal of the current query. This is also supported by the ‘novelty’ assumption: once a user obtains search results from previous queries, the user reads and judges the relevance of the search results, then changes the query and moves to the next goal to gain novel information. For example, in s47, the user issues \( q_1 \) “pseudocyesis history” after examining the search results of \( q_2 \) “pseudocyesis epidemiology”; it suggests that the users probably now prefers documents about ‘history’ rather than ‘epidemiology’. This hypothesis suggests that the importance of previous queries should be discounted according to their relative distances to the current query.
In this work, we examine the above hypotheses and introduce a new query aggregation scheme based on discounting in reinforcement learning [8]. Through experimenting query aggregation schemes on both language modeling document retrieval and reinforcement learning based document retrieval, we draw similar conclusions on which schemes are the best for aggregating queries in session search. The proposed reinforcement learning discounting scheme achieves the best search accuracy with a 4.56% absolute gain over the TREC 2012 best session search system.

2. QUERY AGGREGATION

Various query aggregation schemes in session search can be represented in a general formula. If Score$_{session}$(q$_n$, d) denotes the overall relevance score for a document d to a session that ends at q$_n$, it can be written as:

$$\text{Score}_{session}(q_n, d) = \sum_{i=1}^{n} \lambda_i \cdot \text{Score}(q_i, d)$$ (1)

Score(q, d) is the relevance score between document d and the i-th query q$_i$ in the session. It can be calculated by a wide range of document retrieval methods such as query generation language model [5] and query expansion [4, 1]. In this work, we calculate the probability that t generates d (term weight) $P(t|d)$ based on multimodal query generation language model with Dirichlet smoothing [15]:

$$P(t|d) = \frac{#(t|d) + \beta}{|d| + \beta}$$

where #(t, d) denotes the number of occurrences of term t in document d, $P(t|C)$ is the Maximum Likelihood Estimation (MLE) of the probability that t appears in corpus C, |d| is the document length, and $\mu$ is the Dirichlet smoothing parameter (set to 5000).

Score(q, d) = $P(q|d)$ is calculated as $1 - \prod_{i \in q} (1 - P(t|d))$. $\lambda_i$ in Eq. 1 represents the query weight for each query $q_i$. By formulating $\lambda_i$ differently, we obtain a variety of query aggregation schemes:

- **Uniform**: All queries are weighted equally with the same weight, i.e.,

$$\lambda_i = 1 \text{ for } i = 1, 2, \ldots, n$$ (2)

Several TREC 2011 and TREC 2012 systems adopt this scheme. They expand the current query with either query terms [4] or document terms [1] generated by the previous queries. [1] expands each previous query’s anchor text in search results, then reformulates the current query by equally concatenating the expanded previous queries. [4] weights individual queries equally when applying language modeling for document retrieval. This scheme corresponds to the first hypothesis.

- **Previous vs. current (PvC)**: All queries before the current/last query share the same weight while the current/last query employs a complementary and higher weight:

$$\lambda_i = \begin{cases} \lambda_p & i = 1, 2, \ldots, n - 1 \\ 1 - \lambda_p & i = n \end{cases}$$ (3)

For instance, the 3rd column in Table 2 shows query weights for s47 under this scheme: 0.4, 0.4, and 0.6. Two top TREC 2012 systems [4] and [5] both use this aggregation scheme. [5] sets a discounted weight for the previous queries when computing the MLE of features such as single terms, ordered and unordered phrases. [4] directly applies the discounted weights when combining queries. This scheme corresponds to the second hypothesis.

- **Distance-based discounting**: For session search, we observe that the influence of previous queries and previous search results becomes weaker and weaker. The user shows a desire for novel documents. The last query, which is the most novel, is the most concerned by the user, hence the query weights reversely correlate to the distance between a query to the current/last query. We use a reciprocal function to model this discounting:

$$\lambda_i = \begin{cases} \frac{\lambda_p}{n-i} & i = 1, 2, \ldots, n - 1 \\ 1 - \lambda_p & i = n \end{cases}$$ (4)

For example, if the last query is weighted by 0.6 ($\lambda_p = 0.4$), the term weights in s47 can be defined as the 0.2, 0.4, and 0.6 (the 4th column in Table 2). This scheme corresponds to the third hypothesis. [4] reports their experiments for this scheme in TREC 2012. Table 2 demonstrates the actual query weights for s47 under various query aggregation schemes.

3. A REINFORCEMENT LEARNING BASED QUERY AGGREGATION SCHEME

Besides the distance-based discounting, we propose a new discounting scheme for query aggregation. It is inspired by the discount factor used in reinforcement learning [8]. We first model session search as a Markov Decision Process (MDP) [13] and design a novel reinforcement learning based document retrieval framework. In this session search MDP, queries are states, agents are search engine and user, and actions include the user keeps, adds, and removes query terms and the search engine increases, decreases, and maintains the term weights. The entire session search process can be viewed as the following. 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. We associate each state with a reward function $R$ that indicates the relevance between query $q_i$ and a document $d$. We thus model the relevance of a document $d$ to the current query $q_i$ in a reinforcement learning fashion as:

$$\text{Score}(q_i, d) = P(q_i|d) + \gamma \sum_a P(q_i|q_{i-1}, D_{i-1}, a) \max_{D_{i-1}} P(q_{i-1}|D_{i-1})$$ (5)

where $D_{i-1}$ is the documents retrieved by the previous query $q_{i-1}$, $\max_{D_{i-1}} P(q_{i-1}|D_{i-1})$ is the maximum value of the past rewards while $P(q_i|d)$ is the current reward. $P(q_i|q_{i-1}, D_{i-1}, a)$ is the query transition model, $a$ is the search engine’s action. $\gamma \in (0, 1)$ is the discount factor.

This reinforcement learning document retrieval framework in fact demonstrates a new method of aggregating queries in a session. Instead of discounting the future rewards, we discount the past rewards, i.e., the relevant documents that appear in previous search results. Eq. 5 recursively calculates the reward starting from state $q_1$ and continues until $q_i$. To reflect that a new query plays more important role

<table>
<thead>
<tr>
<th>Query</th>
<th>Uni.</th>
<th>PvC</th>
<th>Dist</th>
<th>RL</th>
<th>Learn.</th>
<th>3-step</th>
</tr>
</thead>
<tbody>
<tr>
<td>pseudocyesis</td>
<td>1</td>
<td>0.4</td>
<td>0.2</td>
<td>0.85</td>
<td>0.95</td>
<td>0.7</td>
</tr>
<tr>
<td>pseudocyesis</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.92</td>
<td>0.94</td>
<td>1</td>
</tr>
<tr>
<td>pseudocyesis history</td>
<td>1</td>
<td>0.6</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2: Queries and query aggregation weights for s47.
than the old ones, the contribution of an earlier query is reduced exponentially by $\gamma \in (0, 1)$. The further apart from the last query, the more discounted the query weight. Considering the whole session, we score the overall relevance for document $d$ to a session that starts at $q_1$ and ends at $q_n$, as:

$$Score_{session}(q_n, d) = Score(q_n, d) + \gamma Score_{session}(q_{n-1}, d)$$

where $q_1, \ldots, q_n$ are the queries in the session. $\gamma \in (0, 1)$ is the discount factor.

### 4. LEARNING THE QUERY WEIGHTS

Besides query aggregation schemes derived based on the hypotheses, we investigate the optimal query weights by a supervised learning to rank method. We employ SVMRank\(^2\) \cite{Joachims:2002} to learn the query weights $\lambda$ from training data. We represent the order of the relevant documents in ground truth as a list of ordered pairs of documents. For example, $(d_j, d_k)$ denotes that document $d_j$ is ranked higher than $d_k$ in a result list. With a linear ranking model $f_\Phi(q)$, we have

$$(d_j, d_k) \in f_\Phi(q) \iff \tilde{w} \Phi(q, d_j) > \tilde{w} \Phi(q, d_k)$$

where $\tilde{w}$ is the weight vector and $\Phi(q, d_j)$ is the kernel function that describes the relevance between query $q$ and document $d_j$. $Score(q, d)$ in Eq. (1) corresponds to the kernel function $\Phi$ and weights $\lambda$ correspond to $\tilde{w}$.

To produce ideal rankings $r^*$ as the training data, we employ the annotation scores assigned by the NIST assessors in the ground truth. Each document in the ground truth is annotated with one the following scores: spam documents whose Waterloo’s “GroupX” spam score is below 70\(^4\) are filtered out from the index. TREC’s official ground truth and evaluation scripts are used. nDCG@10 \cite{JCG2012} and MAP serve as the evaluation metrics for search accuracy.

We experiment the query aggregation schemes, including uniform, previous vs. current (PvC), distance-based discounting (distance-based), reinforcement learning discounting (RL), learned weights and a 3-step scheme. We investigate the best query aggregation scheme(s) for session search.

### 5. EXPERIMENTS

We use the TREC 2011 and TREC 2012 Session Track \cite{Brin2004, TREC2012} data in our evaluation. TREC 2011 contains 76 sessions and 62 topics; TREC 2012 contains 98 sessions and 48 topics. The document collection is ClueWeb09 CatB\(^3\). Spam documents whose Waterloo’s “GroupX” spam score is below 70\(^4\) are filtered out from the index. TREC’s official ground truth and evaluation scripts are used. nDCG@10 \cite{JCG2012} and MAP serve as the evaluation metrics for search accuracy.

For the reinforcement learning discounting scheme (RL), we test $\gamma$ over (0, 1) with an interval of 0.02. Figure 1 illustrates the relationship between nDCG@10 and $\gamma$. nDCG@10 climbs to its peak 0.3368 when $\gamma = 0.92$. The result suggests that a good discount factor $\gamma$ is very close to 1. It implies that the previous query contributes nearly the same as the current query and the discount between two adjacent queries should be mild, not too dramatic.

Figure 3 illustrates the query weights learned by the techniques presented in Section 4. Query weights learned for sessions with various lengths are shown in separate curves. The x-axis displays the query index and the y-axis shows

\(^2\)http://svmlight.joachims.org/

\(^3\)http://lemurproject.org/clueweb09/

\(^4\)http://durum0.uwaterloo.ca/clueweb09spam/
the corresponding query weight. All learned query weights \( \lambda_i \) are normalized by \( \lambda_n \), the weight of the current/last query \( q_n \). For example, \( s47 \) has three queries, i.e., \( n = 3 \), the query weights for \( q_1, q_2, \) and \( q_3 \) are 0.93, 0.94, and 1 (the curve with solid circle in Figure 3).

We observe that the learned query weights \( \lambda_i \) in general decreases while a query’s distance to the last query increases. This indicates that earlier queries have less impact, which confirms with our third hypothesis. However, we notice that the first queries in all sessions demonstrate a higher weight than all other queries except the last/current query. It is probably because that the user often starts with the search theme at the beginning of a session.

5.2 Search Accuracy

The search accuracy for different query aggregation schemes are compared Table 4. Table 4 is based on the retrieval framework of reinforcement learning (Eq. 5). The best result of TREC 2012 Session track [5] is used as the baseline.

Since we have learned optimal query weights from the training data, the learned weights can be directly used to aggregate queries. The result of using the learned method is shown in Table 4 as well. The nDCG@10 for TREC 2012 with reinforcement learning retrieval model reaches 0.3360, which is almost as good as the RL discounting scheme.

In addition, we attempt to simulate the learned weights by adjusting weights for the PvC scheme. Particularly, we use a 3-step function with weights of \((0.7, 0.6, 1)\); the last two queries receive the highest weights 1, the intermediate queries receive the lowest, and the first query receives a middle weight. This 3-step scheme is also plotted in Figure 4 with circles.

We adopt the parameters suggested in Section 5.1 for the aggregation schemes. We observe that PvC, RL, Learning, and 3-step schemes outperform the TREC best. Particularly, RL performs the best for both TREC 2011 and 2012, followed by Learning and 3-step. The distance-based scheme performs the worst. In our opinion, the possible reason is that it discounts the previous queries too much. The 3-step scheme improves the search accuracy a little as compared with the original PvC scheme that is a 2-step function. It implies that the first (two) query is more important than the intermediate queries.

5.3 Plotting the Query Weight Curves

Figure 4 draws the curves for all query aggregation schemes as well as the optimal query weights learned from the training data. The learned weights are averaged over sessions with the same session length as indicated in Table 3. The curve of the averaged learned query weights is the solid line.

We find that RL (indicated by the star symbol) aligns the best with the learned weights; whereas the distance-based discounting scheme the worst. We believe that an aggregation scheme that fits well to the learned curve will perform well in search accuracy. This may explain why RL works the best and the distance-based the worst as well as the performance of other schemes.

Since most sessions have 3 or less queries, we focus more on the last 3 query weights, i.e., the weights for \( q_n, q_{n-1}, \) and \( q_{n-2} \). We find that the proposed RL discounting scheme aligns almost perfectly with the learned weights at the last two queries and only deviates a bit at \( q_{n-3} \). Since most sessions have only 3 queries, this good match to the optimal query weights for the first 3 queries is one of the reasons that makes RL outperform other aggregation schemes.

6. CONCLUSION

This paper studies the effects of using query aggregation for session search. We discuss and experiment several aggregation models based on a few hypotheses. Through experiments over the TREC 2011 and 2012 Session Tracks, we find that the queries that are closer to the current/last query tend to be more important in a session. The learned weights also show that the first query is also important in a session.

7. ACKNOWLEDGEMENT

This research was supported by NSF grant CNS-1223825 and DARPA Memex program.

8. REFERENCES