Effective Structured Query Formulation for Session Search

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

Abstract

In this work, we emphasize on formulating effective structured queries for session search. For a given query, phrase-like text nuggets are identified and formulated into Lemur queries to feed into the Lemur search engine. *Nuggets* are substrings in q_n, similar to phrases but not necessarily as semantically coherent as phrases. We assume that a valid nugget appears frequently in top returned snippets for q_n. In this work, the longest sequences of words consisting of frequent bigrams within the top returned snippets are identified as nuggets and are used to formulate a new query. By formulating structured query using the nuggets, we greatly boost the search accuracy than just using q_n. We experiment both strict and relaxed forms of structured query formulation. The strict form of query formulation achieves an improvement of 13.5% and the relaxed form achieves an improvement of 17.8% on nDCG@10 on TREC 2011 query sets. We further combine the nuggets generated from all queries q_1, … , q_{n-1}, q_n, to formulate one *structured session query* for the entire session. Nuggets from each query are weighed by various weighting schemes to indicate their relations to the current query and their potential contributions to the retrieval performance. We experiment three weighting schemes, *uniform* (all queries share the same weight), *previous vs. current* (previous queries q_1, … , q_{n-1} share the same weight while q_n uses a different and higher weight), and *distance-based* (the weights are distributed based on how far a query's position in the session is from the current query). We find that *previous vs. current* achieves the best search accuracy. For retrieval, we first retrieve a large pool of documents for q_n. We then employ a re-ranking model that considers document similarity between clicked documents and documents in the pool as well as dwell time.

1 Introduction

TREC 2012 Session track features sequences of queries q_1, q_2, … , q_{n-1}, q_n, with only current query q_n being the subject for retrieval. Four subtasks progressively include four types of session information. They are: RL1: a subtask only use the current query q_n; RL2: a subtask uses the previous queries q_1, q_2, … , q_{n-1} and the current query q_n; RL3: a subtask provides top retrieved documents for previous queries; and RL4: additional information about which top results are clicked by users.

In this research, we emphasize on formulating effective structured queries for search tasks within a session. In RL1, we attempt to find text nuggets in the current query to generate structured queries. In RL2, we merge the nuggets extracted from all queries to build a *structured session query*. In RL3 and RL4, we employ anchor texts in the top 10 search results to expand the structured session query. We remove duplicated queries from the query sequence. Dwell time for clicked documents is employed for document re-ranking in RL4. In the following sections, we present our methods for query formulation, query expansion, duplicate removal and document re-ranking.

2 Structured Query Formulation

In a query, several words often bundle together as a phrase to express a coherent meaning. We identify phrase-like text nuggets and formulate them into Lemur queries for retrieval . *Nuggets* are substrings in q_n, similar to phrases but not necessarily as semantically coherent as phrases. We observe that a valid nugget appears frequently in the top returned snippets for q_n. We then use nuggets to formulate new structured queries in the Lemur query language. Particularly, we look or nuggets in the top k snippets returned by Lemur with the original current query q_n. The nuggets are identified by two methods, strict and relaxed, as described below.

2.1 Strict Method

First, we send the original query q_n into Lemur and retrieve the top k snippets over an index built for ClueWeb CatB. Then all snippets are concatenated as a reference document R. The original query is represented as a word list $q = w_1w_2...w_n$.

cubs sport injuries with girls back of knee injuries school sports injuries bladder assistance for **spinal cord** injuries reigning ...
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Figure 1. The word indexing in a snippet built from TREC 2012 session 53 query "servering spinal cord consequenses", where bold words "spinal cord" positioned at 16 and 17 appear in the query.

For every bigram in q_n, we count its occurrences in the reference document R. Frequent bigrams are marked "candidate" if its normalized occurrence exceeds a threshold.

$$\frac{\text{count}(w_i w_{i+1}; R)}{\min(\text{count}(w_i; R), \text{count}(w_{i+1}; R))} \geq \theta \quad (1)$$

where count(x; R) denotes the occurrence of x in the reference document R, w_i and w_{i+1} are adjacent words in the query, θ is the threshold, which is trained to be 0.97 according to the TREC 2011 session data.

For example, in TREC 2012 session 53 query "servering spinal cord consequenses", we find bigram "spinal cord" as candidates.

The bigrams could connect to form longer n-grams. For instance, in TREC 2011 session 11 query "hawaii real estate average resale value house or condo news", we discover that "hawaii real" and "real estate" are both marked "candidate", and they can be merged into a longer sequence "hawaii real estate". On the contrary, "estate average", is not a candidate, hence we cannot form "hawaii real estate average" and "hawaii real estate" is the longest sequence and is recognized as a *nugget*. Formally, we define:

$$nugget = \#1(w_i \, \text{␣} \, w_{i+1} \, \text{␣} \, \dots \, \text{␣} \, w_{i+k}) \quad (2)$$

such that

$$\begin{aligned} &w_j w_{j+1} \text{ is connected for each } j \in [i, i+k-1] \\ &w_{i-1} w_i \text{ and } w_{i+k} w_{i+k+1} \text{are not connected} \end{aligned} \quad (3)$$

where #1 is the ordered window operator with size 1 in Lemur query language which means the words in the bracket are all adjacent, "␣" denotes the space. Consequently the query is broken down into nuggets and single words. All serve as the elements to build up a structured query using the Lemur query language

$$\#\text{combine}(nugget_1 \text{␣} nugget_2 \text{␣} \dots nugget_m \text{␣} w_1 \text{␣} w_2 \text{␣} \dots w_r) \quad (4)$$

where we suppose there are m nuggets and r single words.

2.2 Relaxed Method

#1 is a strict structure operator and may produce many false negatives in retrieval. We therefore introduce another relaxed method for query formulation. We relax the constraints based on the intuition that distance between two words reflects the associativeness of them. Particularly, we first retrieve the top k snippets as in Section 2.1. For RL2, every word's position in the snippet is marked as shown in Figure 1.

We then estimate the position of word w_i as

$$\bar{x}(w_i) = \frac{1}{k} \cdot \sum_{t=1}^{k} \frac{\sum_j x_j(w_i; S_t)}{\text{count}(w_i; S_t)} \quad (5)$$

where k is the number of snippets, S_t is the t^{th} snippet, $x_j(w_i; S_t)$ is the index of the j^{th} instance of w_i in S_t, count(w_i; S_t) is the occurrence of w_i in S_t. The estimated postion is averaged over all the snippets.

For every bigram in the query, the distance between their estimated postions is calculated. We predict the window size (X in #X) of a nugget by learned through a decision tree:

$$nugget = \begin{cases} \#1(w_i \; w_{i+1}) & |\bar{x}(w_i) - \bar{x}(w_{i+1})| \leq 5 \\ \#2(w_i \; w_{i+1}) & 5 < |\bar{x}(w_i) - \bar{x}(w_{i+1})| \leq 10 \\ \emptyset & |\bar{x}(w_i) - \bar{x}(w_{i+1})| > 10 \end{cases} \quad (6)$$

The structured query is then formulated as in eq (4).

For TREC 2012 session 53 query "servering spinal cord consequenses", we obtain the structured query "#2(spinal cord) servering consequenses".

3 Query Expansion

To include more information to formulate effective queries, we expand the queries by using top search results for previous queries and click information.

http://animals.howstuffworks.com/mammals/marsupials.htm http://animals.howstuffworks.com/mammals Marsupials

Figure 2. A sample anchor text.

3.1 Query expansion with previous queries (RL2)

For RL2, we first obtain a set of nuggets and single words from every query q_k = {$nugget_{ik}$, w_{jk}} by the approach presented in Section 2. We then merge these nuggets to form an expanded query:

$$\begin{aligned}&\#\text{weight}(\\&\lambda_1\ \#\text{combine}(nugget_{11}\ nugget_{12}\ldots\ nugget_{1m}\ w_{11}\ w_{12}\ldots w_{1r})\\&\lambda_2\ \#\text{combine}(nugget_{21}\ nugget_{22}\ldots\ nugget_{2m}\ w_{21}\ w_{22}\ldots w_{2r})\\&\ldots\\&\lambda_n\ \#\text{combine}(nugget_{n1}\ nugget_{n2}\ldots\ nugget_{nm}\ w_{n1}\ w_{n2}\ldots w_{nr})\\&)\end{aligned} \tag{7}$$

where λ_k denotes the weight of query q_k. Note that the last #combine is for the current query q_n.

Three weighting schemes are designed to determine the weight λ_k, namely *uniform*, *previous vs. current*, and *distance-based*.

- *uniform*. All queries are assigned the same weight, i.e., $\lambda_k = 1$.
- *previous vs. current*. All previous queries share the same weight while the current query uses a complementary and higher weight. Particularly, we define:

$$\lambda_k = \begin{cases} \lambda_p & k = 1, 2, \ldots, n-1 \\ 1-\lambda_p & k = n \end{cases} \tag{8}$$

  where λ_p is trained to be 0.4 on TREC 2011 session track data.
- *distance-based.* The weights are distributed based on how far a query's position in the session is from the current query. We use a reciprocal function to model it.

$$\lambda_k = \begin{cases} \dfrac{\lambda_p}{n-k} & k = 1, 2, \ldots, n-1 \\ 1-\lambda_p & k = n \end{cases} \tag{9}$$

  where λ_p is trained to be 0.4 based on TREC 2011 session track data, k is the position of a query..

3.2 Query expansion with previous search results (RL3 and RL4)

Anchor texts pointing to a page often provide valuable human-created description to this page [1], which enable us to expand the query by the words in the anchor texts. The anchor log is extracted by the *harvestlinks* in Lemur toolkit. The format is shown in Figure 2: The first column is the page linked by an anchor. The second column is the page this anchor lies in. The third column is the anchor text that describes the page in the first column.

We collect the anchor texts for all previous search results and sort them by frequency in decreasing order. The top 5 frequent anchor texts are appended to the expanded query generated for RL2 with a weight proportional to their frequencies.

$$\begin{aligned}&\#\text{weight}(\\&\lambda_1\ \#\text{combine}(nugget_{11}\ nugget_{12}\ldots\ nugget_{1m}\ w_{11}\ w_{12}\ldots w_{1r})\\&\lambda_2\ \#\text{combine}(nugget_{21}\ nugget_{22}\ldots\ nugget_{2m}\ w_{21}\ w_{22}\ldots w_{2r})\\&\ldots\\&\lambda_n\ \#\text{combine}(nugget_{n1}\ nugget_{n2}\ldots\ nugget_{nm}\ w_{n1}\ w_{n2}\ldots w_{nr})\\&\beta\omega_1\ \#\text{combine}(e_1)\ \beta\omega_2\ \#\text{combine}(e_2)\ \ldots\ \beta\omega_5\ \#\text{combine}(e_5)\\&)\end{aligned} \tag{10}$$

where e_i (i = 1 ... 5) is the top 5 anchor texts, ω_i (i = 1 ... 5) denotes the corresponding frequency of the anchor texts normalized by the maximum frequency, β is a factor to adjust the intervention of the anchor texts, which is trained to be 0.1 on the TREC 2011 session data.

For example, in TREC 2012 session 53, the anchor texts with top frequency are "type of paralysi", "quadriplegia paraplegia", "paraplegia", "spinal cord injury", and "quadriplegic tetraplegic", so the final structured query is "#weight(1.0 #1(spinal cord) 0.6 consequenses 0.4 paralysis 1.0 servering 0.380723 #combine(type of paralysi) 0.004819 #combine(quadriplegia paraplegia) 0.004819 paraplegia 0.004819 #combine(spinal cord injury) 0.00241 #combine(quadriplegic tetraplegic))", where the underlined part is from anchor texts.

This expansion is applied to both RL3 and RL4. In RL4, only the anchor texts in clicked documents are extracted to expand the final structured session query.

Table 1. The nDCG@10 for RL1 using 2011 session track data. Dirichlet smoothing method is used. μ = 4000, k = 10 for strict method and μ = 4000, k = 20 for relaxed method. Methods are compared to the baseline - original query. A significant improvement over the baseline is indicated with a † at p<0.05 level and a ‡ at p<0.005 level. The best run and median run in TREC 2011 are listed for comparison

Method	original query	strict	relaxed	Best run in 2011	Median run in 2011
nDCG@10	0.3378	0.3834	0.3979	0.3789	0.3232
%chg		+13.50%†	+17.79%‡		

4 Duplicated Queries

The trace of how a user modifies queries in a session may suggest the intention of the user so that it can be exploited to study the real information need of the user. We notice that sometimes the user repeats a previous query and makes a duplicated query. We thus refine the final structured session query as follows.

- If there exists a previous query that is the same as the current query q_n, we only use the current query to generate the structured session query.
- If several previous queries are duplicated but they are all different from q_n, we remove these queries when formulating the structured session query.

We also consider a special situation as follows. If one substring is the abbreviation of another, we also consider these two queries duplicates. For example, the only difference between queries "History of DSEC" and "History of dupont science essay contest" is "DSEC" and "dupont science essay contest", in which the former is the abbreviation of the latter, so they are considered duplicates. To detect abbreviations, we scan the query string and split a word into letters if this word is entirely uppercase. In the example above, the first query is transformed to "History of D S E C". When comparing two queries, two words in corresponding positions are considered the same if one of them contains only one capital letter and they start with the same letter. Also in the above example, "dupont" and "D" are considered the same. We process duplicated queries in RL3 and RL4.

5 Document Re-ranking

Users intend to stay in a page that he is interested for longer time [3,4,5]. We use dwell time, which is defined as the elapsed time that user stays in the page, to re-rank the search results in RL3.

The click information provided in RL4 is associated with a start time t_s and an end time t_e. The dwell time Δt can be derived by t_e - t_s. In a session, we retrieve all clicked pages c_i with their dwell time Δt_i. For each returned document d_j for the structured query, the cosine similarity to c_i is computed. We calculate the score of d_j by

$$s(d_j) = \sum_i \mathrm{Sim}(d_j, c_i) \cdot \Delta t_i \tag{11}$$

where Sim(d_j, c_i) is the cosine similarity between d_j and c_i. We rank d_j by $s(d_j)$ in decreasing order as the final search results.

The re-ranking method is applied in run gurelaxphr for RL4. In our experiments, the raw dwell time used in our method strongly bias the document weights towards those with long dwell time, which corresponds to satisfing visits receive much higher weights.

6 Experiments

6.1 Dataset and Evaluation Metrics

We employ the Lemur search engine[1] as the basis and use ClueWeb09 Category B (CatB) as the document collection. The index is built on CatB, and the anchor log is acquired by applying *harvestlinks* on ClueWeb09 Category A (CatA) since the official previous search results are from CatA.

Previous research demonstrates that ClueWeb09 collection involves many spam documents. We filter out spam documents based on Waterloo "GroupX" spam ranking score[2] less than 70[2].

[1] http://www.lemurproject.org/, version 5.0

[2] http://durum0.uwaterloo.ca/clueweb09spam/

Table 2. The nDCG@10 for RL2 using 2011 session track data. Dirichlet smoothing method and strict method are used. μ = 4000, k = 5 for *uniform*, μ = 4500, k = 5 for *previous vs. current* and *distance-based*. Methods are compared to the baseline - original query. A significant improvement over the baseline is indicated with a † at p<0.05 level and a ‡ at p<0.005 level. The best run and median run in TREC 2011 are listed for comparison.

Sheme	original query	uniform	previous vs. current	distance-based	Best run in 2011	Media run in 2011
nDCG@10	0.3378	**0.4475**	**0.4626**	**0.4431**	0.4281	0.3215
%chg		32.47%‡	36.94%‡	31.17%‡		

Table 3. The nDCG@10 for RL3 and RL4 using 2011 session track data. All runs use strict method and the configuration of μ = 4500, k = 5. Methods are compared to the baseline - original query. A significant improvement over the baseline is indicated with a † at p<0.05 level and a ‡ at p<0.005 level. The best run and median run in TREC 2011 are listed for comparison

Baseline = 0.3378 / Method	anchor text				nDCG@10 in 2011	
	all documents		clicked documents (RL4 only)		Best run	Median run
	nDCG@10	%chg	nDCG@10	%chg	RL3	RL3
all queries	**0.4695**	38.99%‡	**0.4680**	38.54%‡	0.4307	0.3259
remove duplicated queries	**0.4836**	43.16%‡	**0.4542**	34.46%‡	RL4	RL4
re-rank by dwell time (RL4 only)	0.4435	31.29%	N/A		0.4540	0.3354

Language model with Bayesian smoothing using Dirichlet priors is applied when performing search by Lemur. The language model is a multinomial distribution, for which the conjugate prior for Bayesian analysis is the Dirichlet distribution:

$$p_\mu(w|d) = \frac{c(w;d) + \mu p(w|C)}{\sum_w c(w;d) + \mu} \tag{12}$$

where $c(w;d)$ denotes the occurrences of term w in document d, $p(w|C)$ is the collection language model, μ is the parameter. The parameter μ is tuned based on the 2011 session data. We do not use the topic descriptions provided by NIST. nDCG@10 is the main metric to evaluate the retrieval performance.

6.2 Results on TREC 2011

For RL1, where only the current query q_n is available, we generate the structured query from q_n by the approach described in 2 and send it into Lemur. The Dirichlet parameter μ and the number of pseudo relevance feedback k are tested on TREC 2011 session data. The documents retrieved by directly searching q_n serve as the baseline. Table 1 shows the nDCG@10 results for RL1 on TREC 2011. By formulating structured query using nuggets, we greatly boost the search accuracy than baseline by 13.50%. The relaxed form achieves even better search accuracy of 0.3979 (+17.79%).

For RL2, we apply query expansion with the previous queries explained in Section 0. We observe that the strict method performs much better, because the window size in relaxed method is hard to optimize for multiple queries. Table 2 presents the nDCG@10 for RL2 on TREC 2011 session data. We find that previous vs. current gives the best search accuracy. It is worth noting that distance-based scheme performs even worse than uniform scheme, which implies that the modification of user intention is complex and we cannot assume that the early query has less importance in the entire session.

For RL3 and RL4, we combine several methods, including anchor texts, removing duplicated queries and re-ranking by dwell time. Table 3 displays the nDCG@10 for RL3 and RL4 on 2011 session track data. It illustrates that removing duplicated queries significantly improves the performance. However, neither re-ranking nor only involving clicked document contributes to the results. The reason may lie in that we treat the time too roughly. Bold fonts in Table 1 to Table 3 indicate a performance better than the 2011 best run.

6.3 Results on TREC 2012

We submit three runs to TREC 2012 session track. The run names, methods and parameters are listed in Table 4, where μ is the Dirichlet smoothing parameter and k is the number of pseudo relevance feedback.

The evaluation results of nDCG@10 and Average Precision (AP) by TREC are presented in Table 5 and Table 6. They show similar trends as what we observe on the TREC 2011 data, but in a much lower range even beneath the results using the original query; which may imply that our query formulation methods may overfit on TREC 2011 session data. We also realize that we fail to well handle the typo in the queries like “consequenses” in TREC 2012 session 53. Nonetheless, using previous queries and eliminating duplicates keep demonstrating significant improvement in search accuracy.

Table 4. Methods and parameter settings in TREC 2012 runs. μ is the Dirichlet smoothing parameter, k is the number of pseudo relevance feedback.

run	RL1	RL2	RL3	RL4
guphrase1	strict method $\mu = 4000$, $k = 10$	strict method query expansion $\mu = 4500$, $k = 5$	strict method query expansion anchor text remove duplicates $\mu = 4500$, $k = 5$	strict method query expansion anchor text all queries $\mu = 4500$, $k = 5$
guphrase2	strict method $\mu = 3500$, $k = 10$	strict method query expansion $\mu = 5000$, $k = 5$	strict method query expansion anchor text remove duplicates $\mu = 5000$, $k = 5$	strict method query expansion anchor text all queries $\mu = 5000$, $k = 5$
gurelaxphr	relaxed method $\mu = 4000$, $k = 20$	relaxed method query expansion $\mu = 4500$, $k = 20$	relaxed method query expansion anchor text remove duplicates $\mu = 4500$, $k = 20$	strict method query expansion anchor text re-ranking by time $\mu = 4500$, $k = 5$

Table 5. nDCG@10 for TREC 2012 runs. Mean of the median of the evaluation results in TREC 2012 are listed.

run	original query	guphrase1	guphrase2	gurelaxphr	Mean of the median
RL1	0.2474	0.2298	0.2265	0.2334	0.1746
RL2		0.2932	0.2839	0.2832	0.1901
RL3		0.3021	0.2995	0.3033	0.216
RL4		0.3021	0.2995	0.29	0.2261

Table 6. AP for TREC 2012 runs. Mean of the median of the evaluation results in TREC 2012 are listed.

run	original query	guphrase1	guphrase2	gurelaxphr	Mean of the median
RL1	0.1274	0.1185	0.1186	0.1223	0.0967
RL2		0.1466	0.1457	0.1455	0.1024
RL3		0.149	0.1483	0.1482	0.1112
RL4		0.149	0.1483	0.1467	0.1176

7 Conclusions

We attempt to generate structured Lemur query for the entire session by involving the previous queries with the rules to eliminate duplicated queries. The evaluation results show that search accuracy can be significantly increased. Although, due to overfitting to TREC 2011 and typos in TREC 2012 queries, the nugget finding method does not work well as what we expect. We still believe it is a promising approach since it greatly boosts the performance on TREC 2011 data. We will investigate the overfitting issue and refine our approach as the future work.

8 References

[1] Albakour, m.-d., Kruschwitz, u., and Nanas, N., University of Essex at the TREC 2011 Session Track. In *TREC* 2011.

[2] Cormack, G.V., Smucker, M.D., and Clarke, C.L.A., Efficient and effective spam filtering and re-ranking for large web datasets. *CoRR*, (2010).

[3] Fox, S., Karnawat, K., Mydland, M., Dumais, S., and White, T., Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst. 23*, 2, 147-168 (2005).

[4] Guo, Q. and Agichtein, E., Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *SIGIR* 2010, 130-137.

[5] Xu, S., Jiang, H., and Lau, F.C.M., User-oriented document summarization through vision-based eye-tracking. In *IUI* 2009, 7-16.