BUPT_WILDCAT at TREC 2011 Session Track

Tang Liu, Chuang Zhang, Yasi Gao, Wenjun Xiao, Hao Huang
Pattern Recognition and Intelligent System Lab,
Beijing University of Posts and Telecommunications, P.R.China
zhangchuang@bupt.edu.cn, {liutangbupt, gaoyasi520, adaxiao.bupt,
huanghao511}@gmail.com

Abstract

This paper is an overview of the runs carried out at TREC 2011 Session track, which proposes several approaches to improve the retrieval performance over one session including the search model based on user behavior, VSM_meta similarity model, optimization based on history ranked lists, optimization based on user's attention time and anchor log. The evaluation results show that our implementations are effective.

1 Introduction

This is the second year of session track, and its goal is to test whether systems can improve their performance for a given query by using previous queries and user interactions with the retrieval system [1].

Participates should complete the following four tasks based on ClueWeb09 Category B corpus through taking advantages of the distinct history data:

a) RL1:only using the current query
b) RL2:using the current query and past query collections
c) RL3:using the current query, past query collections ,and the ranked lists of history query
d) RL4: using the current query, past query collections, the ranked lists of past query, and the attention time of each webpage which user clicks through

The result of RL1 is the basic standard, we can judge whether the user's history action data improve the retrieval results by comparing the retrieval results from RL2 to RL4

The organizers provide some ClueWeb09 corpus related experiment results [2] which are processed by some volunteers (mainly from CMU, Waterloo and some other research institution) on ClueWeb09 corpus. We used some of them and ClueWeb09 corpus based search engine Indri which was set up by CMU [3][4].

In this year's TREC Session Track, our group submitted three runs. The general research structure and all methods used in all runs are clearly listed in processing sequence as below.

Table 1 Methods in all Runs

Run ID	Wildcat1	Wildcat2	Wildcat3
RL1	• VSM_meta (2.2)	• Spam Ranking [5]	• Spam Ranking • PageRank [6]
RL2	• Phrase Recognition[1] • Search Model based on UserBehavior (RL2module)(2.1) • Spam Ranking • VSM_meta	• Phrase Recognition • Search Model based on User Behavior (RL2module) • Spam Ranking +VSM_meta • Webpage Comprehensive Evaluation Model(2.3)	• Word Recognition • Search Model based on User Behavior • Spam Ranking
RL3	• The Anchor Log(2.6) • VSM_meta • Webpage Comprehensive Evaluation Model(2.3)	• Optimization based on history ranked lists (2.4)	• The Anchor Log(2.6)
RL4	• Phrase Recognition • Search Model based on User Behavior (RL4module) • Spam Ranking • VSM_meta	• Optimization based on User's Attention Time (ranking by Cosine Similarity)(2.5)	• Optimization based on User's Attention Time (ranking by KL divergency)(2.5)

In the following sections, Section 2 introduces the different methods referred above in detail. And evaluation results are shown in Section 3. Lastly, Section 4 draws the conclusion.

2 Methodologies

2.1 Search Model based on User Behavior

2.1.1 User Behavior Analysis

In actual search process, users always begin an interaction with a search engine with a series of queries which they will need to reformulate multi times before they find what they are looking for. It is believed that each query has reflected the user's actual search intention to some extent. Each word

[1] Recognize the words in the past queries through matching maximum string

in the previous query, which maybe have been discarded or changed, also is believed to be related with the main search topic, only with little importance if compared with the current queries. In purpose of making up for the little information that the latest query can provide, we built the search model based on user behavior, to done query expansion using history queries all with different weights to each word.

2.1.2 User Behavior Search Model

Assume that in one session, $q_i = (w_1, w_2, \ldots w_n)$ represents the i^{th} input of the user query, W_j represents the j^{th} word in q_i, T_i represents the history query set produced after the i^{th} search behavior.

The model establishing steps are as follows:

a) Initialization$T_1 = \{q_1\}$.

b) $T_2 = T_1 \cup \{q_2\}$, and as follows, $T_{i-1} = T_{i-2} \cup \{q_{i-1}\} = (w_1', w_2', \ldots, w_m')$.

c) Taking $T_i = T_{i-1} \cup \{q_i\}$ as an example, we make the detailed analysis as below.

   Assume that each word in the new query which user presents every time is of the same importance value. That is, in the new query$q_i = (w_1, w_2, \ldots w_n)$, the weight of w_j is $\frac{1}{n}$, j = 1,2,…,n. Normalization $\sum_{i=1}^{n} \frac{1}{n} = 1$.

d) Before user has presented queryq_i, $T_{i-1} = (w_1', w_2', \ldots, w_m')$ is the query set, $(e_1, e_2, \ldots e_m)$ is the weight vector of T_{i-1} corresponding to each word. Normalization$\sum_{i=1}^{m} e_i = 1$.

e) Through the analysis of user search behavior, we believe that history queries can provide a positive effect to the search process, only with little importance compared with later ones. So we use history queries to expend later query, in the meanwhile, giving history queries' weight attenuation to a certain extent.

   Query set expansion:

$$T_i = T_{i-1} \bigcup \{q_i\}$$

   Normalization with expansion of history queries:

$$d\sum_{i=1}^{m} e_i + (1-d)\sum_{i=1}^{n} \frac{1}{n} = 1$$

   d represents the attenuation we have set to history query words, d<0.5. We set d=0.4 in our experiments.

f) Assume that the number of the words both presented in the history query set and query q_i is k.

$$T_{i-1} \bigcap q_i = (w_1^{'}, w_2^{'}, \ldots w_k^{'}) = (w_1, w_2, \ldots w_k), k \le m, k \le n .$$

Query set T_i can be divided as below,

$$T_i = T_{i-1} \bigcup \{q_i\} = (w_1^{'} \ldots w_k^{'}, w_{k+1}^{'} \ldots w_m^{'}, w_{k+1} \ldots w_n)$$

$(e_1^{'}, e_2^{'}, \ldots e_n^{'})$ is the weight vector of T_i corresponding to each word.

Weight normalization with expansion of history queries can be formatted as below corresponding to the Query set T_i above.

$$\sum_{i=1}^{k}[de_i + (1-d)\frac{1}{n}] + d\sum_{i=k+1}^{m}[e_i + (1-d)\sum_{i=k+1}^{n}\frac{1}{n}] = 1$$

g) As a summary, the weight of each element in $(e_1^{'}, e_2^{'}, \ldots e_n^{'})$ is listed below,

$$e_i^{'} = \begin{cases} de_i + (1-d)\frac{1}{n}, i \in [1,k] \\ de_i, i \in [k+1, m] \\ (1-d)\frac{1}{n}, i \in [m+1, n] \end{cases}$$

2.2 VSM_meta Similarity Model

For the purpose of better calculating the similarity between the query and webpage, we established VSM_meta model, which was based on the VSM cosine similarity model, with the expansion usage on the <meta> tag, <title> tag and anchor text information of the webpage, for the consideration that the expansion used information can play a better role of describing the main topic of webpage than others.

The model establishing steps are as follows:

a) For a specified webpage, extract the content of the <title> tag in the html source file as "title".

b) Make an analysis on the <meta> tag, extract the value of the content attribute corresponding to the name attribute whose value is keyword and description. Save the string sequence we have extracted above as "keyword" and "description" in order.

c) Using the anchor text dataset provided by Twente [7], extract the top10 frequency word of the

specified webpage as “anchor text”.

d) Using the html parser we developed to extract the main content of the webpage as “doc text”.

e) Calculate the similarity between the query and the extraction data above using the VSM cosine similarity model, denoted as sim_{title}, $sim_{keyword}$, $sim_{description}$, sim_{anchor}, sim_{doc}.

f) Make a combination of the similarity value above as sim,

$$sim = a * sim_{title} + b * sim_{keyword} + c * sim_{description} + d * sim_{anchor} + e * sim_{doc}$$

g) From a series of contrast experiment, we finally set the parameters as a = 4, b = 1, c = 2, d = 4, e = 1.

2.3 Webpage Comprehensive Evaluation Model

This model implements the synthesis of indri Rank mark and VSM_meta Similarity mark. Indri Rank mark represents the authority of the Webpage, while VSM_meta Similarity mark represents the relativity between Webpage and query. The comprehensive mark of these two marks is as follows:

$$mark = \frac{2}{abs(mark_{indri})} + mark_{VSM\_meta}$$

2.4 Optimization Based on History Ranked Lists

University of Lugano implemented generating optimized ranked list of current query by the rank of documents in ranked list of current query and only one past query [8]. One improved algorithm based on the referred above can be applied to a series of past ranked lists.

We suppose the ranked lists of history queries are L_1, L_2, L_3...L_{n-1}, and L_{n-1} is the ranked list of the last past query, so the retrieval result of current query is L_n. Also, we supposed that L_k'means the optimization of L_k when considering the prior past ranked lists. Finally, $L_n'$is obtained according to history query ranked lists (L_1- L_{n-1}) through the following formula:

$$\begin{cases} score(d, L_k') = \frac{1}{rank(d, L_k)} + a(\frac{1}{rank(d, L_k)} - \frac{1}{rank(d, L_{k-1}')}), k = 1, 2...n \\ L_1' = L_1 \end{cases}$$

In the above formula, *a* value stands for the significance degree of historical ranked lists. In our experiment, a is set to 0.2. We can get the rank of $L_k'$by sorting its score, therefore score(d, L_k') is the

grade of one document d in Lk′, while rank(d, Lk′) is ranking of document d in Lk′.

At last, re-ranking according to the score(d, L_k') by ascending order.

2.5 Optimization Based on User's Attention Time

Attention time reflects the usefulness of the information in the document as viewed by the user. Songhua Xu etc. [9] proposed the attention time prediction algorithm in 2008. It assumes if the contents of two documents are sufficiently similar. We used this method to predict the attention time result lists of current query based on attention time of past click through.

a) For all clicked document in one session build one training sample set,

$$S_i = \{C_{i1}, C_{i2}, ..., C_{in}\}$$

And S_i is the training sample set for the i^{th} session, while C_{ik} is the k^{th} clicked document in the i^{th} session.

b) For each C_{ik} in session i, the k^{th} corresponding attention time $T_{att}(k)$ can be computed by the following formula

$$\begin{cases} T_{inter}(C_{ik}) = (T_{end}(C_{ik}) - T_{start}(C_{ik})) * d_c \\ T_{offset}(C_{ik}) = \dfrac{2\exp(-d * rank(C_{ik}))}{1+\exp(-d * rank(C_{ik}))} \\ T_{att}(C_{ik}) = T_{inter}(C_{ik}) + T_{offset}(C_{ik}) \end{cases}$$

Time interval is denoted by $T_{inter,}$ time offset is $T_{offset,}$ T_{att} represents the attention time. T_{end} and T_{start} stand for the end time and start time which can be found in click tag respectively.

In the second formula rank(C_{ik}) denotes the ranking of k^{th} clicked document C_{ik} in the i^{th} session.

Aiming to reduce the time interval in proportion, we set control parameter d_c as 0.1, and parameter d controlling how sharp drop off is set of 0.2.

For the j^{th} document d_{ij} of RL3 and k^{th} clicked document C_{ik} in the same i^{th} session, calculate the similarity Sim(d_{ij} , C_{ik}) by cosine similarity (wildcat2) or KL distance (wildcat3)

c) In the i^{th} session, obtain the prediction attention time $T_{pre\text{-}att}(d_{ij})$ of the j^{th} document d_{ij} in RL3.

$$T_{pre-att}(d_{ij}) = Sim(d_{ij}, C_{ik}) * T_{att}(C_{ik})$$

d) At last, re-ranking according to the $T_{pre\text{-}att}(d_{ij})$ by ascending order.

2.6 The Anchor Log

University of Essex developed a method for extracting useful terms and phrases to expand the reformulated query in Session Track 2010[10]. On that basis, we modified it with four steps to adapt to the new requirements. In addition, the anchor log for the Category B is available by University of Twente [2].

a) Extract all the document numbers for 2011 Session Data For RL3 in a session.

b) From the anchor log, we extract the corresponding anchor texts out using document numbers

c) For all the anchor texts, we put all the key words in these texts together and count every word's frequency. Then take the top 10 phrases with highest word frequency as the query expansions.

d) Remove stop word from query expansions, combine them with current query to generate new query like this format:

#combime(

0.7#combine (r_c)

0.3#combine ($e_1 e_2 \ldots e_{10}$)

)

Where r_c is the current query, e_i is an expanision term or phrase extracted as explained in the previous step.

e) Filter the result with Spam Ranking.

f) Repeat step a) to e) for all the other sessions.

3 Evaluation Results

For 5 main measurements (err, nerr, ndcg, ap, gap) given by NIST, we use nerr@10,ndcg@10 and gap to compare the performance of our system for goal 1 (to test whether systems can improve their performance for a given query by using previous queries and user interactions with the retrieval system).

Result in bold means it is the highest in its group while result with a * on the right side means it is under median level of all the results.

Table2 shows the overall performance for all subtopics using the measures provided by NIST. It also contains the summary results of all participants in the track.

Table 2 Evaluation results in all_subtopics

wildcat	1	2	3	min	median	max
RL1.nerr@10	0.3849 *	**0.4672**	0.3596 *	0.1878	0.38565	0.4672
RL2.nerr@10	0.4026	0.449	**0.4961**	0.1712	0.38735	0.4961
RL3.nerr@10	0.4358	**0.496**	0.4901	0	0.3846	0.496
RL4.nerr@10	0.4196	**0.5374**	0.5106	0	0.3973	0.5374
RL1.ndcg@10	0.311	0.3496	0.2364 *	0.1384	0.30555	0.3663
RL2.ndcg@10	0.3248	0.3516	0.3847	0.1279	0.31055	0.4061
RL3.ndcg@10	0.3446	0.3851	0.3881	0	0.3084	0.4086
RL4.ndcg@10	0.3348	**0.432**	0.3946	0	0.32625	0.432
RL1.gap	0.098 *	0.1166 *	0.0772 *	0.0267	0.1171	0.1431
RL2.gap	0.1203	0.0751 *	0.0669 *	0.023	0.1091	0.1445
RL3.gap	0.0766 *	0.1181	0.0713 *	0	0.09895	0.1732
RL4.gap	0.1076 *	0.1244	0.1159	0	0.11455	0.1835

Table 3 shows the overall performance for last subtopic using the measures provided by NIST. It also contains the summary results of all participants in the track.

Table 3 Evaluation results in last_subtopics

	wildcat1	wildcat2	wildcat3	min	median	max
RL1.nerr@10	0.267 *	0.3322	0.2140 *	0.1153	0.29255	0.3545
RL2.nerr@10	0.2568	0.2706	0.3086	0.0826	0.2501	0.3648
RL3.nerr@10	0.3025	0.3088	0.3153	0	0.24255	0.3672
RL4.nerr@10	0.2747	0.3330	0.3285	0	0.25315	0.3749
RL1.ndcg@10	0.1954 *	0.2445	0.1351 *	0.0802	0.22075	0.2758
RL2.ndcg@10	0.1889 *	0.2028	0.2387	0.0648	0.19225	0.3034
RL3.ndcg@10	0.214	0.2390	0.2550	0	0.18585	0.3062
RL4.ndcg@10	0.2019	0.2594	0.2485	0	0.1972	0.3051
RL1.gap	0.0889 *	0.122	0.0622 *	0.0228	0.1161	0.1435
RL2.gap	0.0951	0.0614 *	0.0684 *	0.0178	0.09305	0.1438
RL3.gap	0.0677 *	0.1076	0.0776 *	0	0.08385	0.1417
RL4.gap	0.089 *	0.1149	0.1057	0	0.0973	0.1511

After analyzing the results, the findings are summarized as follows:

a) By using the metric ndcg@10 or nerr@10, it is show that whether in table or table, the general trend of results is: RL1<RL2<RL3<RL4. That indicates that systems can really improve their performance for a given query by using previous queries and user interactions with the retrieval system.

b) The RL4 group has the best performance in your experiments. That indicates the time factor can also influence the results.

c) By comparing overall results in all subtopics and last subtopic, we find some of our results have best performance in subtopics but perform not so well in last subtopic. That may result from that our methods take all the historical data in to consideration.

d) In RL2 group, the wildcat3 has the best performance. All the experiments in RL2 group we use the same method before filter spam. After filtering spam, in wildcat3 we only use indri to deal with the data. This may indicate that using indri to deal with data is better than using VSM_meta.

e) RL3.wildcat3 has the best score in RL3 group, which indicates using the anchor log for query expansion is useful.

f) 'gap' (graded average precision) is used to evaluate the average precision for the every system. Though in ndcg@10 and nerr@10 our system perform quite well, in 'gap' we have most scores under median level. We think this may be due to the our result in each experiment is of large quantity. Since the number of most relevant websites is limited, the more the results are, the less relevant websites there will be.

4 Conclusion

This year's Session Track provided a platform to evaluate the effectiveness of Information Retrieval systems in improving their performance for a given query by using previous queries and user interactions with the retrieval system (including clicks on ranked results, dwell times, etc.). The results of our runs are promising. We used a number of different techniques to attempt to improve performance over a user session, which will give suggestions for further research.

Reference

[1] TREC 2011 Session Track Guidelines.pdf

[2] http://lemurproject.org/clueweb09/derived-data.php

[3] http://boston.lti.cs.cmu.edu/Services/batchquery/

[4] http://boston.lti.cs.cmu.edu/Services/search/clueweb09_catb/lemur.cgi

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

[6] http://boston.lti.cs.cmu.edu/clueweb09/wiki/tiki-index.php?page=PageRank

[7] http://wwwhome.cs.utwente.nl/~hiemstra/2010/anchor-text-for-clueweb09-category-a.html

[8] Mostafa Keikha,Parvaz Mahdabi, Shima Gerani. University of Lugano at TREC 2010. University

of Lugano

[9] Songhua Xu, Yi Zhu, Hao Jiang and Francis C.M. Lau. A User-Oriented Webpage Ranking Algorithm Based on User Attention Time. Zhejiang University, Yale University, The University of Hong Kong.

[10] M-Dyaa Albakour, Udo Kruschwitz, Jinzhong Niu,Maria Fasli. Autoadapt at the Session track in TREC 2010. University of Essex