

ISI at the TREC 2013 Federated task

Dipasree Pal, Mandar Mitra

Indian Statistical Institute,
203, B.T. Road, Kolkata-108

Abstract

The resource selection task contains a variety of Search Engines (SEs). There exists many domain specific SEs. These SEs can retrieve domain related query results more efficiently, whereas, they are not good at retrieving out-of-domain query results. Thus, it is difficult to predict the performance of a SE in a given query using the results of other queries. In our approach, each query has been searched in the web by all the SEs (using Google search API's search site option). We try to predict the performance of each SE with only top 8 retrieved results. Based on the term frequency of each query term in the retrieved results, our method ranks those SEs for that query.

At the time of run submission, we missed a few queries. After that, we rank SEs for all queries using the same method. We observed that the result is better in nDCG@20 than the ‘median’ result. Also, when measured in ERR@20, the result is better in more queries. Median ERR@20, is substantially higher than our result for one query, this affects the average.

In the results merging task, the same concept has been applied. Here also, we did not use the actual retrieved results, as it will take time after retrieval. The score of a SE found in the resource selection task, is used here. The documents retrieved by a SE are assigned a score that is a combination of its rank and the SE’s score. This can be computed without retrieving actual results.

1 Introduction

There are many search engines (SEs) available on the Web. They are good in different ways. For example, the result of a cartoon related query is expected

to be better in cartoon related search sites rather than other search sites. Likewise, an information need in another domain may be better served by some other SE. Thus, a metasearch engine that combines results coming from different SEs remains a potentially useful application. Metasearch can be accomplished in two stages: (i) resource selection, which will give priority to good SEs, and (ii) results merging, which combines the results coming from the selected SEs. In our approach, we focus on the first task. The second task makes use of a score computed during the first task.

2 Proposed method

The following sections describe the procedure that we adopted for resource selection and results merging.

2.1 Resource selection task

For this task, we take the help of Google search API¹. This API provides a “search site” option. Since most of the resources (search engines) generally search local content, we use this API for each test query along with the search site option. Naturally, this does not work well for SEs that do not confine their search to local content.

Let D denote the top 8 results² returned for a query $Q = \{q_1, \dots, q_n\}$ by a search engine E . The frequency of the query words q_i in those webpages is measured, and used to compute a score for E as shown in Equation 1.

$$Score(E) = \sum_{d \in D} \sum_{q_i \in Q} \log(tf(q_i, d) + 0.001). \quad (1)$$

Here, 0.001 is used for smoothing. We did not use raw linear tf since it has a tendency to emphasize on frequent occurrence of a single term and hence unable to cover multiple query aspects. Thus, in order to enforce number of distinct matches between query and documents, we used logarithm to restrict the growth of tf function.

2.2 Results merging task

We calculate the score of each ranked document by giving the fractional (as in Equation 2) score of that search engine according to the documents rank.

¹<http://code.google.com/apis/websearch>

²Since, we wanted a reasonable number of documents. Also, by default, this API returns 8 results.

Then according to the score, we merge all the ranked documents found from several search engines. It does not require any information from original retrieved documents.

$$Score_{ER} = Score(E) * \log\left(1 + \left(\frac{1}{R}\right)\right). \quad (2)$$

Here, $Score_{ER}$ denotes the score of the R -th document retrieved by the search engine E .

For example, a few top ranked documents (for one query) in the merged file are listed in Table 1. We presented the table in two parts. The first part shows that the first few documents in the merged file, are the top ranked retrieved documents by different SEs. The last part shows that several ranks of different SEs are mixed.

<i>Task – SE – Q – R</i>	<i>Rank</i>	<i>Score_{ER}</i>
FW13-e147-7001-01	0	35.792249
FW13-e037-7001-01	1	34.865858
FW13-e052-7001-01	2	32.803953
FW13-e036-7001-01	3	32.314452
FW13-e050-7001-01	4	29.216708
FW13-e103-7001-01	5	28.137963
FW13-e147-7001-02	19	20.937123
FW13-e018-7001-01	20	20.867404
FW13-e101-7001-01	21	20.846817
FW13-e138-7001-01	22	20.606018
FW13-e124-7001-01	23	20.537951
FW13-e037-7001-02	24	20.395219
FW13-e099-7001-01	25	20.387607
FW13-e082-7001-01	26	20.335621

Table 1: Example merged file. Here, ‘SE’ denote the search engine / resource identifier, ‘Q’ means query number, ‘R’ is the rank of that document when searched by the specific SE, ‘Rank’ denotes the rank of that document after merging the results of all SEs, and ‘Score’ denotes the merging score according to which the merging is done.

3 Results

3.1 Resource selection task

Our official submission missed a few queries. We corrected this omission after the official submission. For the (unofficial) run that includes all queries, we notice that, in case of nDCG@20, for 5 out of the 50 queries, our run produces results that are substantially better than the best official results (the difference ranges from 0.4 to 0.9).

Table 2 also shows that, the ‘max’ run have ERR@20 equal to 0.00 for most of the queries. For 4 queries, our method performs better than ‘median’ whereas ‘median’ run does better on 2 queries. But average performance of ‘median’ run is better than our method because of its highly significant effectiveness on a single query, which in essence influences the average value.

3.2 Results merging task

In case of P@10, 13 queries get similar or better performance in our method than ‘median’. In case of nDCG, 11 queries get better or similar results by our method. Note that we do not use the originally retrieved results while merging. The merging step simply uses the score of the search engine.

4 Analysis and Future work

In our approach, we used Google search API. It is meaningful for most of the search sites, as they generally search their own corpus rather than the Web. This is not correct for a few of the sites. They do not contain their own corpus, instead search the web.

Thus, we will next try to find an automatic search pattern of each search sites. Then with that automatic method, we will skip the part of Google search API and expect to have better results.

References

- [1] Dong Nguyen, Thomas Demeester, Dolf Trieschnigg, and Djoerd Hiemstra. Federated search in the wild: The combined power of over a hundred search engines. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM ’12, pages 1874–1878, New York, NY, USA, 2012. ACM.

topic	nDCG@20			ERR@20		
	isi_pal	median	max	isi_pal	median	max
7001	0.00000	0.00000	0.30103	0.00000	0.00000	0.00011
7003	0.00044	0.00039	0.34523	0.00000	0.00000	0.00000
7004	0.36539	0.15509	0.39784	0.00000	0.00000	0.00000
7007	0.00000	0.00000	0.42876	0.00000	0.00000	0.00000
7009	0.00772	0.34771	0.98758	0.00000	0.00000	0.00000
7018	0.00000	0.00000	0.39589	0.00000	0.00000	0.00000
7025	0.00005	0.23087	0.38744	0.00000	0.00000	0.00000
7030	0.26772	0.23835	0.97144	0.00000	0.00000	0.00000
7033	0.00000	0.00011	0.49791	0.00000	0.00000	0.00000
7034	0.01328	0.01785	0.43366	0.00000	0.00000	0.00000
7039	0.00000	0.00020	0.54499	0.00000	0.00000	0.00000
7040	0.00715	0.01175	0.49345	0.00000	0.00000	0.00000
7042	0.10747	0.03454	0.23077	0.00000	0.00000	0.00000
7046	0.23539	0.00461	0.31844	0.00000	0.00000	0.00000
7047	0.43068	0.00000	0.35621	0.00000	0.00000	0.00000
7056	0.00000	0.00109	0.99748	0.00000	0.00000	0.00000
7067	0.00028	0.00360	0.23889	0.00000	0.00000	0.00006
7068	0.12483	0.17240	0.61754	0.00000	0.00000	0.00000
7069	0.24040	0.26555	0.63061	0.00000	0.00000	0.00000
7075	0.00000	0.46520	0.99987	0.00000	0.29167	1.00000
7076	0.00000	0.00360	0.14775	0.00000	0.00000	0.00000
7080	0.00000	0.10048	0.43188	0.00000	0.00000	0.00000
7084	0.29076	0.00694	0.39022	0.00011	0.00000	0.00020
7087	0.61055	0.10969	0.67973	0.00000	0.00000	0.00000
7089	0.31554	0.00000	0.99890	0.12512	0.00000	1.00000
7090	0.29660	0.14671	0.68399	0.00000	0.00000	0.00000
7094	0.00093	0.00126	0.29844	0.00000	0.00000	0.00000
7096	0.00678	0.11895	0.29053	0.00000	0.00000	0.00000
7097	0.00000	0.11474	0.46924	0.00000	0.00000	0.00000
7099	0.00000	0.00000	0.38685	0.00000	0.00000	0.00000
7103	0.00067	0.00001	0.33471	0.00000	0.00000	0.00000
7109	0.00000	0.35085	0.87722	0.00000	0.00001	0.00004
7115	0.59973	0.01097	0.30016	0.00000	0.00000	0.00000
7124	0.00000	0.22753	0.43041	0.00000	0.00000	0.00000
7127	0.38685	0.19343	0.46845	0.00000	0.00000	0.00000
7129	0.58513	0.04126	0.45714	0.00000	0.00000	0.00000
7132	0.80023	0.00009	0.32643	0.00110	0.00000	0.00033
7145	0.16281	0.08209	0.43866	0.00000	0.00000	0.00000
7209	0.00003	0.00004	0.24145	0.00000	0.00000	0.00000
7258	0.00002	0.00817	0.30900	0.00000	0.00000	0.00000
7348	0.00000	0.00974	0.98491	0.00000	0.00000	0.00000
7404	0.23137	0.38684	0.99997	0.00000	0.00000	0.00000
7406	0.00001	0.00001	0.30102	0.00000	0.00000	0.00000
7407	0.38307	0.29009	0.85588	0.00186	0.00100	0.00781
7415	0.22325	0.11925	0.34930	0.00000	0.00000	0.00000
7465	0.00000	0.23153	0.38686	0.00000	0.00000	0.00000
7485	0.95263	0.01204	0.48456	0.00000	0.00000	0.00000
7504	0.00168	0.00168	0.62971	0.00000	0.00000	0.00000
7505	0.37360	0.37360	0.97655	0.00000	0.00000	0.00000
7506	0.00077	0.12010	0.38675	0.00000	0.00000	0.00000
amean	0.16047	0.14100	0.29913	0.00256	0.00835	0.02003

Table 2: Results of resource selection task. The labels isi_pal, median and max shows our run, median run and max run respectively.

topic	P@10			nDCG		
	isi_pal	median	max	isi-pal	median	max
7001	0.3000	0.50000	0.80000	0.4927	0.54420	0.60560
7003	0.7000	0.80000	1.00000	0.6825	0.70180	0.72940
7004	0.2000	0.20000	0.40000	0.3788	0.33820	0.67380
7007	0.0000	0.50000	0.70000	0.4148	0.54030	0.60570
7009	0.0000	0.10000	0.50000	0.3033	0.38690	0.56260
7018	0.0000	0.40000	0.60000	0.3543	0.50720	0.61300
7025	0.2000	0.50000	0.90000	0.5431	0.54310	0.68220
7030	0.0000	0.30000	0.60000	0.3274	0.47950	0.73280
7033	0.2000	0.10000	0.30000	0.4180	0.40510	0.48190
7034	0.0000	0.20000	0.30000	0.3664	0.44340	0.48420
7039	0.0000	0.10000	0.50000	0.2688	0.34310	0.42120
7040	0.1000	0.20000	0.30000	0.2725	0.35570	0.54210
7042	0.1000	0.30000	0.80000	0.5252	0.58670	0.65430
7046	0.1000	0.00000	0.10000	0.3601	0.31470	0.38290
7047	0.4000	0.10000	0.40000	0.5745	0.43380	0.57450
7056	0.1000	0.40000	0.50000	0.4384	0.49200	0.60870
7067	0.1000	0.30000	0.70000	0.4509	0.53770	0.62420
7068	0.1000	0.30000	0.60000	0.4303	0.45120	0.58650
7069	0.3000	0.60000	0.80000	0.5624	0.61350	0.65350
7075	0.6000	0.40000	0.90000	0.5774	0.62920	0.78560
7076	0.1000	0.30000	0.60000	0.5487	0.54340	0.62900
7080	0.0000	0.10000	0.40000	0.2370	0.37610	0.59970
7084	0.3000	0.40000	0.90000	0.5084	0.62840	0.77410
7087	0.0000	0.30000	0.60000	0.3391	0.43260	0.60070
7089	0.3000	0.20000	0.60000	0.4842	0.55620	0.72300
7090	0.0000	0.10000	0.40000	0.3515	0.42180	0.49850
7094	0.1000	0.30000	0.50000	0.4097	0.43000	0.58440
7096	0.1000	0.50000	0.70000	0.3807	0.57460	0.62490
7097	0.0000	0.00000	0.40000	0.3587	0.44570	0.50780
7099	0.0000	0.30000	0.70000	0.3036	0.37460	0.59860
7103	0.2000	0.40000	0.70000	0.5311	0.53110	0.59250
7109	0.1000	0.20000	0.60000	0.3915	0.47910	0.60790
7115	0.3000	0.50000	0.80000	0.5737	0.62140	0.70480
7124	0.1000	0.30000	0.70000	0.3676	0.44620	0.64000
7127	0.0000	0.30000	0.40000	0.2759	0.37210	0.54030
7129	0.0000	0.20000	0.50000	0.3341	0.39840	0.49650
7132	0.0000	0.10000	0.90000	0.4033	0.50380	0.85700
7145	0.1000	0.30000	0.60000	0.4420	0.56680	0.64630
7209	0.3000	0.50000	0.80000	0.5366	0.60210	0.67280
7258	0.2000	0.10000	0.30000	0.4620	0.38130	0.46200
7348	0.1000	0.10000	0.80000	0.3441	0.32640	0.49470
7404	0.5000	0.50000	0.80000	0.6388	0.63550	0.67740
7406	0.0000	0.20000	0.70000	0.4138	0.46310	0.59700
7407	0.6000	0.80000	1.00000	0.6975	0.72640	0.79590
7415	0.2000	0.60000	0.90000	0.4474	0.56530	0.68340
7465	0.0000	0.00000	0.30000	0.2958	0.35170	0.44480
7485	0.1000	0.10000	0.40000	0.4924	0.36350	0.49240
7504	0.0000	0.10000	0.50000	0.3545	0.42410	0.49870
7505	0.3000	0.30000	0.60000	0.5569	0.60570	0.69340
7506	0.0000	0.20000	0.30000	0.3947	0.47680	0.58720
all	0.1500	0.31800	0.41400	0.4323	0.46900	0.54380

Table 3: Results of results merging task. The labels isi_pal, median and max shows our run, median run and max run respectively.