Federated Search using Query Log Evidence

João Damas[12], José Devezas[12], and Sérgio Nunes[12]

[1] INESC TEC
[2] Faculty of Engineering of the University of Porto, Portugal

Abstract In this work, we targeted the search engine of a sports-related website that presented an opportunity for search result quality improvement. We reframed the engine as a Federated Search instance, where each collection represented a searchable entity type within the system, using Apache Solr for querying each resource and a Python Flask server to merge results. We extend previous work on individual search term weighing, making use of past search terms as a relevance indicator for user selected documents. To incorporate term weights we define four strategies combining two binary variables: integration with default relevance (linear scaling or linear combination) and search term frequency (raw value or log-smoothed). To evaluate our solution, we extracted two query sets from search logs: one with frequently submitted queries, and another with ambiguous result access patterns. We used click-through information as a relevance proxy and tried to mitigate its limitations by evaluating under distinct IR metrics, including MRR, MAP and NDCG. Moreover, we also measured Spearman rank correlation coefficients to test similarities between produced rankings and reference orderings according to user access patterns. Results show consistency across all metrics in both sets. Previous search terms were key to obtaining a higher effectiveness, with runs that used pure search term frequency performing best. Compared to the baseline, our best strategies were able to maintain quality on frequent queries and improve retrieval effectiveness on ambiguous queries, with up to ∼six percentage points better performance on most metrics.

Information retrieval, Federated search, Domain-specific search

1 Introduction

This work was developed in the context of a Portuguese sports website focused on delivering all types of football-related information at national and international level. Our research focused on improving the current search engine by reframing the problem as an instance of Federated Search, as well as proposing changes to the indexing and retrieval processes by incorporating influence from previous searches in order to help predict future relevance. The underlying assumption is that, by incorporating external information, we can positively impact the retrieval performance.

This document is structured as follows. Section 2 presents a definition of Federated Search, as well as relevant proposals on new indexing strategies for similar contexts. Section 3 provides an overview of the available collections, with Section 4 detailing the indexing modifications by the introduction of previous search terms. Section 5 provides an overview of how the retrieval process and the overall system work, including our proposed modifications for new relevance formulas. The solution is evaluated in Section 6, with Section 7 reflecting on the results and potential future work.

2 Related Work

Federated search is a classic information retrieval task that involves querying a set of independent search engines and then centrally merging the results to provide a single result list to the user [14]. This task involves four phases: *Resource Description*, *Resource Selection*, *Results Merging* and *Results Presentation* [4], with the first and third being the most relevant in our context. The first phase concerns the indexing process. After resources are selected and their relevant documents retrieved, in the *Results Merging* phase there is the need to order them in a single ranking. Even if they have similarities in the retrieval process, they are not equal and, therefore, query scores for documents from different resources cannot be directly compared [1].

We center our survey on previous research focused on new indexing strategies, particularly those that use past searches to enrich document representations. The motivation is that people might remember the document not by its content, but by some description given by another person other than the author.

Fagin et al. [6] present a system that uses three separate indices to index an intranet's pages: one for their content, one for the title and related metadata, and one that combines all anchor text leading to that page. Ding et al. [5] use a similar approach, but with an index based on previous searches. They reason that a single source of evidence "*is not enough to construct a good website search engine, especially when the page is new or seldom accessed*". This is an important point that resembles our work closely: we expect to have a heavy tail of pages that are less visited and, therefore, would suffer from a lack of data problem if trying to use solely log data for indexing. In their tests, they compared their proposed approach with a single index with all the combined content and concluded that the former performed better. Moreover, the log index proved most effective in retrieving some top results instead of a full relevant set.

Zhou et al. [17] also use a three-fold index using anchor texts and search log entries, however, their work differs on index construction, with a sliding window to capture surrounding text, as well as term propagation between consecutive pages visited through referrer information. Not all propagated terms have the same importance, as terms from pages closer to the one being processed assume higher weights. Their best results also came when linearly combining the separate indices instead of merging all terms into a single structure.

Oakes et al. [12] take an extreme approach and index documents only by previous search terms. They argue that, in a system with multilingual documents and searches, this strategy allows to merge all previous searches, regardless of language, providing users with easier access to documents in a broader set of languages, as many can be present in the index. Terms are weighted in a TF-IDF like approach. In our case, this strategy alone wouldn't likely work, due to the heavy-tail limitation presented above.

3 Resource Description

The system incorporated four different collections, corresponding to the same amount of distinct searchable entity types within the search engine, amongst the most frequently searched. The first collection stored a sample of 2,000 competitions. Each one had a name stored in a description-like field, as well as, occasionally, an abbreviation. A separate collection stores teams. Each team in our collected sample of 5,000 entries had a unique name stored in a dedicated field. The third collection contains entities of type Manager. Each one of the 3,000 sampled managers had their full name stored. Moreover, there was a keywords field that stored alternative designations, similar to nicknames, for them. However, this field was rarely filled, with only 2.4% of sample entries containing a non-empty value associated. Finally, the last collection holds entities of type Player, which are grouped together to form teams. Our sample contained 10,000 players, storing their name, past teams and in-field preferred positions.

4 Search Term Payloads

We added one extra field to all collections, *searchTerms*, that stored these terms, alongside a search relevance weight for each one. We used the formula provided by Oakes et al. [12] to calculate search term relevance. However, this relevance was complementary to the overall document relevance score instead of a replacement. The difference also shows how the goals between both works differed: in the original proposal, authors aimed at exploring how this technique could aid in multilingual search, while we intended to facilitate previous search pattern recognition for future searchers. More specifically, given an entity e, for each query term t, they defined its search relevance with a TF-IDF like weight as: $Weight(t, e) = TF_{t,e} \times \log \frac{N}{E_t}$ where $TF_{t,e}$ is the number of searches for entity e that use term t, E_t is the number of entities whose term t led to a click in it and N is the total number of documents (here, limited to our samples' size).

Their weighting scheme, despite being based on a traditional relevance formula, allowed to highlight terms that were often used to reach an entity, were seldom used to search others, or ideally both. In order to achieve an optimal weighting scheme, and because term frequency differences between distinct terms for a given entity were sometimes considerable, we implemented two variants. The first is as described, while the second utilizes logarithmic term frequency.

To incorporate this into collection documents, we made use of a Lucene feature that was recently ported to Solr: payloads.[1] The idea is to associate some score to individual terms that, here, represents a relevance confidence weight.

[1] `https://lucidworks.com/post/solr-payloads/`

5 Execution Flow

Despite the actual search process responsibility being delegated to Solr, we were not able to implement a fully federated solution within it. The lack of out-of-the-box support for a federated architecture led us to introduce a middleware server, using Python's Flask[2], that interacted with Solr to query collections and then merge results locally. When a user submitted a query, the server would build the request object that was then sent through the Solr API to each collection. Upon receiving all results, score normalization was performed, before sending a final ordered list back to the client.

5.1 Querying Independent Collections

Given the difference in schemas, requests had to be adjusted according to the entity that was being queried. We started by constructing the common core of the request. This included defining the query parser and general options. We opted to use Solr's eDismax query parser[3] due to its flexibility, namely supporting, e.g., field-dependent boosts. In addition, the *stopwords* parameter was set to false, so that these terms weren't removed during the analyzer pipeline. Moreover, despite the existence of a dedicated Payload query parser, that wasn't what met our needs: we intended on calculating document relevance by Solr standards *and* by term payloads, not one or the other. We used eDismax to calculate traditional document relevance and made use of the in-built *payload* function and the possibility of defining custom, calculated fields for each returned document. This function takes a payload field and a term and returns the respective weight or 0 otherwise. Upon receiving a query, the server splits it using a whitespace delimiter and creates one custom field per term. Each term is assigned a sequential field (payload_0, payload_1, ...) that searches the *searchTerms* field common to all entities.

Given this common core, we added the search fields, along with the respective boosts, for each entity, so that we could take advantage of the data structure to value matches on more relevant fields. Furthermore, we replicated these boosts into a phrase search parameter. This way, not only could we boost documents that matched terms in a given field, but also on consecutive term occurrences, much like phrase querying. Boost wise, we valued an entity's main name most over all other fields, followed by other stored designations (e.g., nicknames, abbreviations), and, finally, other entity-specific fields (e.g., former teams for managers and players).

5.2 Incorporating Payload Scores

Having queried a collection, the resulting documents all have a *score* field, with the estimated Solr relevance formula result, as well as *payload_** fields, one for

[2] `https://flask.palletsprojects.com/en/1.1.x/`

[3] `https://lucene.apache.org/solr/guide/8_4/the-extended-dismax-query-parser.html`

each query term issued. To merge these scores, we use a set of strategies based both on linear scaling and linear combination of factors and terms, respectively. To this end, given a document relevance score *RelS* and payload scores $\mathrm{PL}_{0..n}$ for that document, we defined and experimented with 4 strategies. In the first strategy, *Prod*, the document's final score linearly scaled with the matching payload scores, that is: $Score_{Prod} = RelS \times \prod_{i=0}^{n} \mathrm{PL}_i, [\mathrm{PL}_i \neq 0]$.

On the other hand, the *Sum* strategy performed a linear combination between the relevance score and the sum of the matching payload scores: $Score_{Sum} = \alpha \times RelS + (1 - \alpha) \times \sum_{i=0}^{n} \mathrm{PL}_i, \alpha \in [0, 1]$ where the α parameter could attribute more or less importance to term payloads and defaulted to 0.5. The last two strategies follow from the first ones, but with log-smoothing. The *ProdLog* strategy is similar to *Prod*, but the sequence product of the payloads was log-smoothed. Finally, the *SumLog* incorporation strategy was derived from *Sum*, similarly to *Prod* and *ProdLog*, i.e., the summation component is log-smoothed.

5.3 Results Merging

The last step in the retrieval process was the merging of results from all four collections in order to produce one ordered list. To merge documents, we had to normalize their scores, so that they could be comparable between different collections. One of the most well-known algorithms for this task is CORI [3]. For a given document D retrieved from collection C, CORI defines its normalized score as $FinalScore = S_D * \frac{1+0.4*S_C}{1.4}$, where S_D is the original document score (after term payload incorporation) and S_C is the collection's score, while the normalization constants are a product of experimentation.

An important component in score normalization is the document's origin collection's score. This value should reflect the retrieved documents' overall importance for the final ordered list, thus its value could be the difference between a document placing in the top 10 or much further down. We followed the work of Hawking et al. [10], who, inspired by Rasolofo et al. [13], used a LMS strategy to calculate a collection's score. LMS, short for "*using result Length to calculate Merging Score*" [13], requires no collection metadata or samples, ranking collections based on the result set size for a given query. More specifically, a collection's LMS score is defined as: $LMS_C = \log\left(1 + \frac{|R_C| \times K}{\sum_{i=1}^{n} |R_i|}\right)$ where $|R_i|$ is the size of the result set returned by collection i for the query, n is the number of collections to merge, and K is a scaling constant. In the original proposal, the authors set a value of $K = 600$, which we adopted.

6 Evaluation

To evaluate our solution, we used a collection of real queries and evaluated the performance under several standard metrics. In order to annotate test queries with the correct answers, we used click information as a relevance proxy. The work of Joachims et al. [8] exposes the potential dangers of using click-through information to this end. To mitigate these problems, a diverse set of metrics was adopted. In order to validate the results, behavior across metrics should remain reasonably consistent. Finally, when possible, work that supports the use of click-through data as an acceptable replacement for manual relevance judgement was cited.

6.1 Datasets

We used two distinct sets of queries to analyze the engine's behavior under different circumstances and scenarios. Queries were mostly short, with a large majority spanning no longer than 2 terms. The first set of queries contained popular queries in terms of frequency. To build it, we collected the 200 most frequently submitted queries and all the entities clicked as a result of that search, keeping only the most clicked one, alongside the number of clicks it received. Liu et al. [11] concluded that, for navigational queries, it was possible to automatically annotate queries with the most clicked result as the correct answer, obtaining over 96% accuracy on the annotation process when compared to manual judgements. In our context, queries were expected to be predominantly navigational, as users mostly sought a specific entity when searching. On average, the top clicked result in each of the 200 queries had an average click share of 85%. Moreover, in nearly 120 queries the top clicked result had over 90% of all clicks for that query. Such skewness towards one result was also a good indicator that helped validate the automation of the annotation process.

The second set of queries consisted of interrogations that produced high variability in clicked results, i.e., different entities were considered the correct answer depending on the search session, and none of them had a noticeable click share majority. In this case, we filtered queries based on individual entity click entropy, using the formula presented by Kulkarni et al. [9]. The building process followed a similar flow to the frequent queries set. We started by collecting the 200 queries with the highest entropy value. However, this time, we considered all results clicked as potential answers. In this case, the top clicked results had an average click share of 36.9%. Thus, due to the ambiguous nature of these queries, we considered multiple possible answers per query. Accordingly, these query sets will be referred to as *Frequent Query Set (FQS)* and *Entropy Query Set (EQS).*

6.2 Evaluation Metrics

To evaluate the robustness of the system, both query sets were tested against different metrics. As we had no access to a test collection with full relevance judgements for all documents, we avoided measures that directly dealt with recall. More specifically, we used *Mean Reciprocal Rank (MRR)* [16], including a click-weighted variant proposed by Walter Underwood [15], *Mean Average Precision (MAP)* [2], *Success (at N)*, a percentage of queries with at least one correct answer in the top N results, as proposed by Zhu et al. [18] (with values $N = \{1, 5\}$). Moreover, for the Entropy Query Set, we also used *Normalized Discounted Cumulative Gain (nDCG)* [7], as well as Spearman's rank correlation coefficient, which assesses if the relationship between two variables from dependent samples (here, ranking positions) can be characterized as a monotonic function, i.e., both variables grow in the same direction. Statistical hypothesis testing was performed to assess if coefficient values varied significantly between different strategies in our solution and if there were improvements over the baseline.

6.3 Results and Discussion

We now present the obtained results, partitioned by query set. All metrics were applied with a cutoff at ranking 10. In the following subsections, we consider each independent combination of search term payload score calculation and incorporation as an independent search engine. The engine resulting from the combination of Pure TF (in payload calculation) and the product operator (in payload incorporation) is referred to as *PProd*. *CurrSol* represents the baseline that is the current solution in the product engine. Finally, *NoPayload* refers to a local baseline for our solution, where there was no search term payload in use, i.e., *plain federated Solr*. Regarding Sum and Sumlog strategy variants, we experimented, *a priori*, with α values in the range [0, 1], with a step of 0.1, in order to assess the best linear combination weights. Optimality was achieved when $\alpha = 0.3$, that is, the document's score was made mostly from search term payloads (70%). As we approached both extremes of the tested range, performance tended to get worse, as expected.

Frequent Query Set Results for the Frequent Query Set are shown in Table 1. For each metric we highlight the strategies that had the highest value. An immediate observation is that the performance of the plain Solr baseline (*NoPayload*) ranks worst in every metric. More specifically, in around 47% of the queries, the engine was able to return the correct result in the first position, and only in a little over two-thirds placed it in the top 5. This was further corroborated by the lower values of weighted and unweighted MRR, reflecting the typically lower ranking positions where correct results were found. Note that, by considering only one correct answer per query, unweighted MRR and MAP values were equal, therefore we show only one designation, in this case the former.

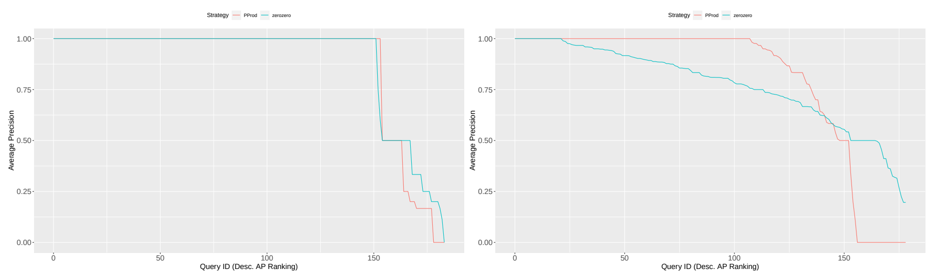
Regarding the different combinations of our solution, it appeared that using Sumlog for search term payload incorporation, regardless of the TF type used for its calculation, lead to worse results. In fact, when using TF Log, results were almost comparable to not using payloads at all. This was likely a sign that both the Sumlog incorporation strategy and logarithmic TF calculation flattened payload values beyond significance. The latter's effect could be seen on the other variants as well: when compared to their Pure TF counterpart, no variant performed better on any metric. In the remaining three strategies (*PProd*, *PSum* and *PProdlog*), performances were more similar and close to the CurrSol baseline. As for the other two, they were almost indistinguishable performance-wise. When compared to the CurrSol baseline, results were also very similar, with a small advantage for the baseline in the Success@5 metric. Indeed, frequently searched entities already produced good results, as was shown by these results. Therefore, the main conclusion was that some of the combinations of our solution were capable of replicating the good results already provided by the baseline for these situations.

Table 1. Evaluation results for the Frequent Query Set.

Strategy	MRR@10	wMRR@10	Success@1	Success@5
CurrSol	0.84535	0.99580	0.83696	0.87370
NoPayload	0.56864	0.66987	0.47283	0.68478
PProd	**0.84375**	**0.99508**	**0.83696**	**0.85326**
PSum	0.84013	0.99476	0.83152	**0.85326**
PProdlog	0.82201	0.97689	0.79348	**0.85326**
PSumlog	0.66498	0.78311	0.56522	0.78804
LProd	0.80471	0.95340	0.76630	**0.85326**
LSum	0.68554	0.83061	0.59239	0.80435
LProdlog	0.76771	0.91409	0.70109	0.84783
LSumlog	0.60714	0.70529	0.51630	0.73370

Finally, we focused on one metric and looked at its individual values for all queries across one of the top combinations, *PProd*, and the CurrSol baseline. We chose Average Precision (AP) as the metric to observe, not only due to its robustness, but also since it considered multiple answers per query, an important factor so that we could more confidently perform the same analysis later on the Entropy Query Set. Figure 1a shows the obtained values. As expected, values were very close, reflecting what was obtained in their averages. Despite being able to keep a better performance for a short while, the current production engine then had a slower rate of descent for the lowest scoring queries.

Entropy Query Set Evaluation results for this query set are presented in Table 2. Furthermore, note that, while the NDCG column label reads *NDCG@10*, the values presented are the arithmetic average for all queries, and not individual



(a) Frequent Query Set (b) Entropy Query Set

Figure 1. AP values per query in descendent order for each query set.

values. Once again, the local baseline of not having search payload incorporation achieves the worst performance of any combination, a repeated behavior. MAP, which was equivalent to MRR in the Frequent Query Set, also follows a similar pattern, though with even lower values. Finally, the newly introduced metric, NDCG@10, also suggests that there wasn't much gain as we moved in the ranking when compared to other strategies. Being a pattern present in both query sets, this confirms that payloads are key for improving retrieval performance.

Sumlog strategy variants are, once again, the ones with the lowest performance, demonstrating the low expressiveness of payloads in that scenario. However, contrary to the Frequent Query Set, there were more strategies that suggest an improvement over the CurrSol baseline. More specifically, both *PProd* and *LProd* surpass other strategies in all metrics (e.g., they are the only ones with a MAP value over 0.8). Moreover, they also present the highest discounted cumulative gain, with NDCG@10 values of around 0.82 each. In terms of correct result presence in the top 1 and 5 ranking positions, other strategies, such as *PSum* and *PProdlog*, perform nearly as well as the former two. In fact, their difference in other metrics is usually small (around 3 percentage points maximum in MAP).

Table 2. Evaluation results for the Entropy Query Set.

Strategy	**MRR@10**	**wMRR@10**	**MAP@10**	**Success@1**	**Success@5**	**NDCG@10**
CurrSol	0.7682	0.9540	0.7492	0.7324	0.8388	0.7451
NoPayload	0.5082	0.7187	0.4822	0.3966	0.6760	0.5198
PProd	0.8304	**0.9955**	0.8011	**0.7989**	0.8659	**0.8201**
PSum	0.8216	**0.9953**	0.7797	0.7933	0.8547	0.8035
PProdlog	0.8233	0.9927	0.7880	0.7933	0.8603	0.8038
PSumlog	0.6974	0.9133	0.6617	0.5978	0.8436	0.6979
LProd	**0.8317**	0.9932	**0.8102**	**0.7989**	**0.8715**	0.8161
LSum	0.7832	0.9683	0.7472	0.7263	0.8547	0.7648
LProdlog	0.8042	0.9865	0.7760	0.7654	0.8547	0.7825
LSumlog	0.6057	0.7458	0.5803	0.4916	0.7709	0.6043

The last metric we used, the Spearman ranking correlation coefficient, was used to assess which, if any, of our combinations produced rankings that were, on average, closer to the results users tend to click on. Therefore, we performed hypothesis tests by comparing each combination from our solution with the CurrSol baseline and verifying if we could confidently state that correlation values tend to be higher. We started by assessing the coefficient data's normality using the Shapiro-Wilk normality test. The null hypothesis $H0$ states that the data follows a normal distribution, and can be rejected if the *p-value* falls under the chosen alpha, which we adopted to a standard 0.05. After running on all coefficient datasets (from our solution and the CurrSol baseline), the *p-value* was always much lower than the threshold set, therefore the normality assumption failed every time. Thus, we discarded parametric tests that relied on the normality assumption. Our choice was then the non-parametric Wilcoxon signed-rank test. This test is a reliable alternative when there is no support for data normality and can be used to assess if there are significant changes in the distribution of two variables, and if these changes are one or two-sided. Given two variables X and Y, the null hypothesis is $H0 : X \leq Y$. Our interest was then in checking which combinations were able to reject the null hypothesis. Once again, we considered a threshold of $p = 0.05$. Table 3 shows the obtained results. Results below the defined threshold are marked with an asterisk (*). Values lower than 0.001 are highlighted with a double asterisk (**).

Table 3. Wilcoxon signed-rank test *p-values* for Spearman correlation coefficients.

Strategy	***p-value***
NoPayload	0.922
PProd	1.352e-12**
PSum	1.521e-6**
PProdlog	4.444e-9**
PSumlog	0.036*
LProd	1.311e-7**
LSum	0.007*
LProdlog	3.685e-6**
LSumlog	0.517

Results show that, for both *NoPayload* and *LSumlog* strategies, we can't state that the correlation coefficients are higher than the baseline. Moreover, the *PSumlog* strategy, despite being able to reject the null hypothesis, results in a much higher value (0.036), attributing less confidence to this strategy as well. Otherwise, all strategies' results allow us to say that they produce rankings with higher correlation to typical user result access patterns. Finally, Figure 1b shows the individual AP values for this query set. Differences are more noticeable in this scenario, also reflecting the enhancement visible when evaluating this query set. There is a sudden break for our strategy towards the worst performing queries. This is likely due to the correct answer inclusion strategy: we consider all queries that had at least one answer present in the samples, no matter how often it was clicked. For queries that closely resemble this edge case, it was not possible for our solution to place them in the top 10 answers. Overall, our solution appears to produce much better results according to user access patterns.

7 Conclusions

In this work, we took an existing sports search engine and proposed reframing it as a Federated Search instance, where each collection corresponds to a searchable entity type. We also indexed previous search terms for a given document, a strategy that was shown to have high discriminative power. Each term's weight is derived from a TF-IDF adaption, reflecting how often it was used to reach that and other documents. To incorporate payload values, we defined four strategies as a product of two binary variables: score update (linear scaling or linear combination) and previous search term frequency (raw value or log-smoothed). Finally, for merging and normalization, we made use of previous work on CORI variations that used returned result set size as a main signal for collection quality.

Our evaluation process consisted of comparing results from different combinations of our proposed solution and the current production engine. For this, we extracted two query sets from the search logs, one with the most frequents queries and one with the most ambiguous ones (entropy set). To annotate queries with the correct answer, we had to resort to using clicks as a relevance proxy. For the frequent query set, there was one correct answer, the most clicked entity, since they always had a high click share, averaging 85%. As for the entropy query set, a majority was seldom found, hence all entities clicked were considered in a way to reflect graded relevance. In order to mitigate limitations of this approach, we used several distinct IR metrics, including MAP, MRR and NDCG. Results for frequent queries show that we were able to match the current system quality, and, additionally, increase retrieval effectiveness up to six percentage points on most collected metrics for ambiguous queries. Raw search term frequency achieves better results than its counterpart, with both linear scaling and combination providing the best results alongside it, with a slight advantage for the former. In fact, stability was observed across metrics for different strategies, reinforcing confidence on the reliability of the evaluation methodology. This shows how a single, uniform and integrated system was able to provide quality answers for a diverse set of information needs and queries. Moreover, it demonstrates how past searches can be a positive influence in relevance determination for document ordering for future searchers.

As future work, it would be interesting to experiment with other types of incorporation strategies beyond what was tested here. This could lead to a more systematic analysis in order to assess possible patterns concerning optimal strategy types. Another possible continuation includes replicating the behavior of the current system by favoring terms used more recently, so as to avoid wrongful bias of possible search term spikes. Regarding evaluation, quality measurement under real system usage would further aid in testing the solution's quality.

Acknowledgements

This paper would have not been possible without the collaboration of the zerozero.pt team, who kindly provided us continuously refined search logs that were the foundation of the work developed. This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020.

References

1. Arguello, J.: Federated Search for Heterogeneous Environments. Ph.D. thesis, Carnegie Mellon University (2011)
2. Buckley, C., Voorhees, E.M.: Evaluating Evaluation Measure Stability. SIGIR Forum 51(2), 235–242 (Aug 2017)
3. Callan, J.P., Lu, Z., Croft, W.B.: Searching Distributed Collections with Inference Networks. In: Proceedings of the 18th Annual International ACM SIGIR Conference

on Research and Development in Information Retrieval. p. 21–28. SIGIR '95, ACM, New York, NY, USA (1995)

4. Callan, J.: Distributed Information Retrieval. In: Advances in Information Retrieval, pp. 127–150. Kluwer Academic Publishers, Boston (2005)
5. Ding, C., Zhou, J.: Log-based Indexing to Improve Website Search. In: Proceedings of the 2007 ACM symposium on Applied computing - SAC '07. p. 829. ACM Press, New York, New York, USA (2007)
6. Fagin, R., Kumar, R., McCurley, K.S., Novak, J., Sivakumar, D., Tomlin, J.A., Williamson, D.P.: Searching the Workplace Web. In: Proceedings of the twelfth international conference on World Wide Web - WWW '03. p. 366. ACM Press, New York, New York, USA (2003)
7. Järvelin, K., Kekäläinen, J.: Cumulated Gain-Based Evaluation of IR Techniques. ACM Trans. Inf. Syst. 20(4), 422–446 (Oct 2002)
8. Joachims, T.: Optimizing Search Engines Using Clickthrough Data. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02. p. 133. ACM Press, New York, New York, USA (2002)
9. Kulkarni, A., Teevan, J., Svore, K.M., Dumais, S.T.: Understanding Temporal Query Dynamics. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining. p. 167–176. WSDM '11, ACM, New York, NY, USA (2011), `https://doi.org/10.1145/1935826.1935862`
10. Li, P.V., Thomas, P., Hawking, D.: Merging Algorithms for Enterprise Search. In: Proceedings of the 18th Australasian Document Computing Symposium. p. 42–49. ADCS '13, ACM, New York, NY, USA (2013)
11. Liu, Y., Fu, Y., Zhang, M., Ma, S., Ru, L.: Automatic Search Engine Performance Evaluation with Click-through Data Analysis. In: Proceedings of the 16th International Conference on World Wide Web. p. 1133–1134. WWW '07, ACM, New York, NY, USA (2007)
12. Oakes, M., Xu, Y.: A Search Engine Based on Query Logs, and Search Log Analysis by Automatic Language Identification. In: Proceedings of the 10th Cross-Language Evaluation Forum Conference on Multilingual Information Access Evaluation: Text Retrieval Experiments. p. 526–533. CLEF'09, Springer-Verlag, Berlin, Heidelberg (2009)
13. Rasolofo, Y., Abbaci, F., Savoy, J.: Approaches to Collection Selection and Results Merging for Distributed Information Retrieval. In: Proceedings of the Tenth International Conference on Information and Knowledge Management. p. 191–198. CIKM '01, ACM, New York, NY, USA (2001)
14. Shokouhi, M., Si, L.: Federated Search. Foundations and Trends® in Information Retrieval 5(1), 1–102 (2011), `http://dx.doi.org/10.1561/1500000010`
15. Underwood, W.: Measuring Search Relevance with MRR. `https://observer.wunderwood.org/2016/09/12/measuring-search-relevance-with-mrr/` (2016), [Online; accessed June 2022]
16. Voorhees, E.M., Tice, D.M.: Building a Question Answering Test Collection. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 200–207. SIGIR '00, ACM, New York, NY, USA (2000)
17. Zhou, J., Ding, C., Androutsos, D.: Improving Website Search Using Web Server Logs. In: Proceedings of the 2006 Conference of the Center for Advanced Studies on Collaborative Research - CASCON '06. p. 22. ACM Press, New York, USA (2006)
18. Zhu, H., Raghavan, S., Vaithyanathan, S., Löser, A.: Navigating the Intranet with High Precision. In: Proceedings of the 16th international conference on World Wide Web - WWW '07. p. 491. ACM Press, New York, New York, USA (2007)