A Search Architecture Enabling Efficient Diversification of Search Results

Gabriele Capannini, Franco Maria Nardini,
Raffaele Perego, and Fabrizio Silvestri

ISTI–CNR, Pisa, Italy
{name.surname}@isti.cnr.it

Abstract. In this paper, we deal with efficiency of the diversification of results returned by Web Search Engines (WSEs). We extend a search architecture based on additive Machine Learned Ranking (MLR) systems with a new module computing the diversity score of each retrieved document. Our proposed solution is designed to be used with other techniques, (e.g. early termination of rank computation, etc.). Furthermore, we use an efficient state-of-the-art diversification approach based on knowledge extracted from query logs, and prove that it can efficiently works in a additive machine learned ranking system, and we study its feasibility.

1 Introduction

Diversification of Web search results is a hot research topic. The majority of research efforts have been spent on studying effective diversification methods able to satisfy web users. In this paper we take a different turn and we consider the problem from the efficiency perspective. As Google's co-founder Larry Page declares[1]: *"Speed is a major search priority, which is why in general we do not turn on new features if they will slow our services down"*. In [5], we define a methodology for detecting when, and how, query results need to be diversified. We rely on the well-known concept of query refinement to estimate the probability of a query to be ambiguous. In the same paper, we show how to derive the most likely refinements, and how to use them to diversify the list of results. Then, we propose an original algorithm allowing the diversification task to be accomplished effectively and efficiently.

In this paper, our focus is on plugging efficient diversification in additive MLR systems. In modern WSE query response time constraints are satisfied employing a two-phase scoring. The first phase inaccurately selects a small subset of potentially relevant documents from the entire collection (e.g. a BM25 variant). In the second phase, resulting candidate documents are scored again by a complex and accurate MLR architecture. The final rank is usually determined by additive ensembles (e.g. boosted decision trees [6]), where many scorers are executed sequentially in a chain and the results of the scorers are added to compute the final document score.

[1] http://www.google.com/corporate/tech.html

The main contribution of this paper is to propose an architectural solution that can be integrated in an additive MLR pipeline so as to efficiently perform the diversification process at query-processing time. We provide a formal analysis of the problem and we study its feasibility in the existing MLR architectures.

The paper is organized as follows: Section 2 presents our formalization of the diversification problem. Section 3 describes the search architecture aiming at enabling the efficient diversification of search results. In Section 4, we present our conclusions and we outline possible future work.

2 Diversification using Query Logs

Users query WSEs by submitting sequences of requests that are recorded in query logs. Let Q be a query log. Let q and q' be two queries submitted by the same user during the same logical session recorded in Q. We adopt the terminology proposed in [1], and we say that a query q' is a "specialization" of q if the user information need is stated more precisely in q' than in q. Let us call S_q the set of specializations of an ambiguous/faceted query q mined from the query log. Given the popularity function that computes the frequency of a query topic in Q, and a query recommendation algorithm trained with query log Q, an algorithm that exploits the query log sessions to provide users with suggestions concerning related queries, can be adapted for devising specializations S_q at query-processing time.

Now, let us give some additional assumptions and notations. $\mathcal{D}$ is the collection of documents indexed by the WSE which returns, for any given query q, an ordered list of documents $R_q \subseteq \mathcal{D}$. The rank of document $d \in \mathcal{D}$ within R_q is indicated with $rank(d, R_q)$. A distance function $\delta : \mathcal{D} \times \mathcal{D} \rightarrow [0,1]$, having non-negative and symmetric properties is defined as $\delta(d_1, d_2) = 1 - cosine(d_1, d_2)$, where $cosine(\,)$ denotes the cosine similarity function.

The utility function defined in Equation (1) denotes how good $d \in R_q$ is for satisfying a user intent that is better represented by specialization q'.

$$U(d|R_{q'}) = \sum_{d' \in R_{q'}} \frac{1 - \delta(d, d')}{rank(d', R_{q'})} \tag{1}$$

The intuition for U is that a result $d \in R_q$ is more useful for specialization q' if it is very similar to a highly ranked item contained in the results list $R_{q'}$.

Using the above definitions of distance (δ) and utility (U), we are able to define a query-log-based approach to diversification.

MaxUtility(k): *Given: query q, the set R_q of results for q, two probability distributions $P(d|q)$ and $P(q'|q)\,\forall q' \in S_q$ measuring, respectively, the likelihood of document d being observed given q, and the likelihood of having q' as a specialization of q, the utilities $U(d|R_{q'})$ of documents, a mixing parameter $\lambda \in [0, 1]$, and an integer k. Find a set of documents $S \subseteq R_q$ with $|S| = k$ that maximizes*

$$U(S|q) = \sum_{d \in S} \sum_{q' \in S_q} (1 - \lambda) P(d|q) + \lambda P(q'|q)\, U(d|R_{q'})$$

with the constraints that every specialization is covered proportionally to its probability. Formally, let $R_q \bowtie q' = \{d \in R_q | U(d|R_{q'}) > 0\}$*. We require that for each* $q' \in S_q$*,* $|R_q \bowtie q'| \geq \lfloor k \cdot P(q'|q) \rfloor$*.*

Our technique aims at selecting from R_q, the k results that maximize the overall utility of the results list.

MaxUtility(k) aims to maximize directly the overall utility. The problem can thus be simplified and solved in a very simple and efficient way [5]. In the same paper we propose *OptSelect*, an algorithm that aims to maximize $U(S|q)$ by simply computing for each $d \in R_q$ the utility of d for specializations $q' \in S_q$ and, then, to *select* the top-k highest ranked documents. Obviously, we have to carefully select results to be included in the final list in order to avoid choosing results that are relevant only for a single specialization. To select results, we use a set of $|S_q|$ min-heaps each of those keeps the $\lfloor k \cdot P(q'|q) \rfloor + 1$ most useful documents for that specialization. *OptSelect* returns the set S maximizing the objective function of MaxUtility(k) in linear time. Moreover, the running time of *OptSelect* is linear in the size of document considered. Indeed, all the heap operations are carried out on data structures having a constant size $\leq k$.

3 Proposed Architecture

In the previous section we have sketched the *OptSelect* algorithm as an efficient solution for the diversification task. Here, we show how such a solution needs to be adapted in order to be plugged in a modern MLR system having a pipelined architecture. Let us assume that, given a query q, MLR algorithms are used to rank a set $D = \{d_1, \ldots, d_m\}$ of documents according to their relevance to q. Then the k documents with the highest score are returned. To this end, additive ensembles are used to compute the final score $s(d_i)$ of a document d_i as a sum over many, simple scorers, i.e. $s(d_i) = \sum_{j=1}^{n} f_j(d_i)$, where f_j is a scorer that belongs to a set of n scorers executed in a sequence. Moreover, the set of scorers is expected to be sorted by decreasing order of importance. This because, as argued in [4], if we can estimate the likelihood that d_i will end up within the top-k documents, we can early exit the $s(d_i)$ computation at any position $t < n$, computing a partial final score using only the first t scorers. For these reasons, it is important to define a solution that is fully integrable with the existing systems. Another important aspect to consider is the cost of each f_j that must be sustainable w.r.t. the others scorers. In particular, we assume that the cost c of computing $f_j(d_i)$ is constant and the total cost of scoring all documents in D is, thus $\mathcal{C}(D) = c \cdot m \cdot n$. For tasks with tight constraints on execution time, this cost is not sustainable if both m and n are high (e.g. $m > 10^5$ and $n > 10^3$ as shown in [4]).

To achieve the previously specified goal, WSE needs some additional modules in order to enable the diversification stage, see Figure 1. Briefly, our idea is the following. Given a query q, perform simultaneously both the selection of the documents potentially relevant for q from the entire collection (module `BM25`) and the retrieve of the specializations for q (module `SS`). Assuming that `SS`

performs faster than both DR and BM25, the module $f_{\texttt{DVR}}$ can be placed in any position of the MLR pipeline, i.e. $f_1 \rightarrow_{\ldots} f_n$. The target of $f_{\texttt{DVR}}$ is, then, to exploit Equation (1) for properly increasing the rank of the incoming documents as the other pipelined scorers do. Note that in this case, that is different from *OptSelect* running context, the final extraction of top-k documents is left to the MLR pipeline that already performs this operation automatically. In the following, we give more detail on our approach.

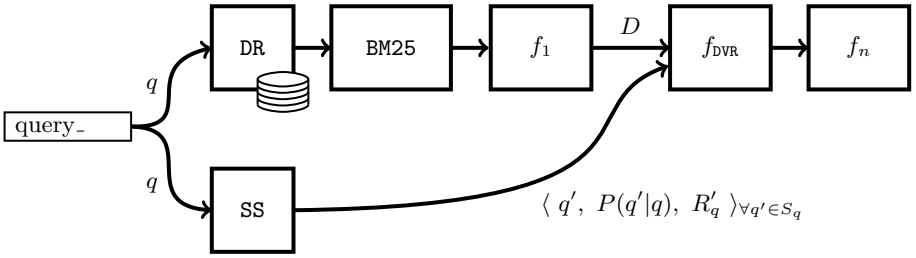

Fig. 1. A sketch of the WSE architecture enabling diversification.

For any given query q submitted to the engine, we dispatch q to the document retriever DR that processes the query on the inverted index, and to the module SS that generates the specializations S_q for q. SS processes q on a specific inverted index structure derived from query logs: the same proposed in [2]. SS returns a set of specializations S_q, a distribution of probability $P\left(q'|q\right) \forall q' \in S_q$, and a set $R_{q'} \forall q' \in S_q$ of sketches representing the most relevant documents for each specialization. Concerning the feasibility in space of the inverted index in SS, note that each set $R_{q'}$ related to a specialization $q' \in S_q$ is very small compared to the set of whole documents R_q to re-rank, i.e. $|R_{q'}| \ll |R_q|$. Furthermore, using *shingles* [3], only a sketch of a few hundred bytes, and not the whole documents, can be used to represent a document without significant loss in the precision of our method[2]. Resuming, let ℓ be the average size in bytes of a shingle representing a document and let h be the average space needed to store the set S_q of specializations for a query q by using the related inverted index, we need at most $(\ N \cdot |S_{\widehat{q}}| \cdot |R_{\widehat{q}'}| \cdot \ell\ +\ N \cdot h\ )$ bytes for storing N ambiguous query along with the data needed to assess the similarity among results lists. For example, considering a number of ambiguous queries of order of hundreds of thousands, tens of specializations per query, and hundreds of documents per specialization, we need an inverted index for SS of about 10 GB.

Now, let us focus on $f_{\texttt{DVR}}$. As the other modules belonging to the MLR pipeline, also $f_{\texttt{DVR}}$ receives a set of documents D as a stream from its preceding module, scores the elements, then release the updated set. However, contrarily to other diversifying methods analyzed in [5], $f_{\texttt{DVR}}$ is able to compute on the fly the

[2] note that shingles are already maintained by the WSE for near duplicate document detection.

diversity-score for each document d. In fact, exploiting the knowledge retrieved from the query log, our approach does not require to know in advance the composition of D to diversify the query result because SS provides the proper mix of different means related to q. In particular, we firstly compute for each $d \in D$ the related shingle. As stated in [3], the related sketch can be efficiently computed (in time linear in the size of the document d) and, given two sketches, the similarity $1-\delta(d,d')$ of the corresponding documents (i.e. $d \in D$ and each document d' returned by SS, i.e. $d' \in R_{q'}\ \forall q' \in S_q$) can be computed in time linear in the size of the sketches. The resulting similarity thus concurs to compute $U(d|R_{q'})$, i.e. the variation of final score of the document d.

4 Conclusions

We studied the problem of plugging the WSE results diversification step in a additive MLR system. In order to do that, we exploited a diversification technique suitable for working in this ranking system and thus able to compute at query-processing time the diversity score of each document. By exploiting this approach, the selection of the relevant results to return to the user can be done by simply selecting the top-k documents with the highest score. Our proposed solution is designed to be used with other techniques, (e.g. early termination of rank computation, etc.). We sketched the resulting MLR search architecture, and we outline a first preliminary study on the feasibility in space of the technique.

References

1. Boldi, P., Bonchi, F., Castillo, C., Vigna, S.: From 'dango' to 'cakes': Query reformulation models and patterns. In: Proc. WI'09. IEEE CS Press (2009)
2. Broccolo, D., Marcon, L., Nardini, F.M., Perego, R., Silvestri, F.: Generating suggestions for queries in the long tail with an inverted index. submitted for review
3. Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the web. Comput. Netw. ISDN Syst. 29, 1157–1166 (September 1997)
4. Cambazoglu, B., Zaragoza, H., Chapelle, O., Chen, J., Liao, C., Zheng, Z., Degenhardt, J.: Early exit optimizations for additive machine learned ranking systems. In: Proceedings of the third ACM international conference on Web search and data mining. pp. 411–420. ACM (2010)
5. Capannini, G., Nardini, F.M., Perego, R., Silvestri, F.: Efficient diversification of web search results. Proceedings of the VLDB, Vol. 4, No. 7 (April 2011)
6. Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., Sun, G.: A general boosting method and its application to learning ranking functions for web search. Advances in Neural Information Processing Systems 19 (2007)