Scalable and Efficient Web Search Result Diversification

KAWEH DJAFARI NAINI, L3S Research Center
ISMAIL SENGOR ALTINGOVDE, Middle East Technical University
WOLF SIBERSKI, Bundesdruckerei GmbH

It has been shown that top-*k* retrieval quality can be considerably improved by taking not only relevance but also diversity into account. However, currently proposed diversification approaches have not put much attention on practical usability in large-scale settings, such as modern web search systems. In this work, we make two contributions toward this goal. First, we propose a combination of optimizations and heuristics for an implicit diversification algorithm based on the desirable facility placement principle, and present two algorithms that achieve linear complexity without compromising the retrieval effectiveness. Instead of an exhaustive comparison of documents, these algorithms first perform a clustering phase and then exploit its outcome to compose the diverse result set. Second, we describe and analyze two variants for distributed diversification in a computing cluster, for large-scale IR where the document collection is too large to keep in one node. Our contribution in this direction is pioneering, as there exists no earlier work in the literature that investigates the effectiveness and efficiency of diversification on a distributed setup. Extensive evaluations on a standard TREC framework demonstrate a competitive retrieval quality of the proposed optimizations to the baseline algorithm while reducing the processing time by more than 80% and up to 97%, and shed light on the efficiency and effectiveness tradeoffs of diversification when applied on top of a distributed architecture.

CCS Concepts: • **Information systems → Information retrieval diversity**; **Novelty in information retrieval;**

Additional Key Words and Phrases: Web search engines, distributed result diversification

ACM Reference Format:
Kaweh Djafari Naini, Ismail Sengor Altingovde, and Wolf Siberski. 2016. Scalable and efficient web search result diversification. ACM Trans. Web 10, 3, Article 15 (August 2016), 30 pages.
DOI: http://dx.doi.org/10.1145/2907948

1. INTRODUCTION

The success of a search engine in a highly competitive market is tightly bound to how effectively its top-ranked answers satisfy the user's information need. Not surprisingly, considerable research effort is devoted to ranking candidate answers and determining the optimal top-*k* results both by academia and industrial players. A recent yet well-recognized aspect in this sense is diversifying the top search results, especially when the user's search intent is not clear, which has its roots in minimizing the risk in a

This work was partially supported by the K3 project funded by the German Federal Ministry of Education and the European Commission Seventh Framework Program under grant agreement No. 600826 for the ForgetIT project.
Authors' addresses: K. D. Naini, L3S Research Center, Appelstr. 9a, 30167, Hannover, Germany; email: naini@l3s.de; I. S. Altingovde (corresponding author), Computer Engineering Department, Middle East Technical University, 06580, Ankara, Turkey; email: altingovde@ceng.metu.edu.tr; W. Siberski (current address), NewStore, Inc., Bödekerstr. 56, 30161, Hannover, Germany; email: siberski@acm.org.

DOI: http://dx.doi.org/10.1145/2907948

financial portfolio [Markowitz 1952]. That is, just like an investor who is not sure about the future diversifies the selection of stocks in his or her portfolio, a search system that cannot predict the search intent behind a query should diversify the top search results to minimize the risk of frustrating its users [Wang and Zhu 2009].

While the classical examples for such ambiguous queries include *java* (or *jaguar*, or *apple*), where a search system should return answers related to *Java programming language* and *Java island*, it is soon realized that diversity is needed at different levels even for queries that look much less ambiguous at first glance [Santos et al. 2010a, 2011]. For instance, users submitting the query *java programming* can still have very diverse intents, such as finding an introductory tutorial, obtaining pointers to some resources like books or class notes, discovering forums, checking ads for tutors, and so on [Santos et al. 2010a].

The previous example demonstrates that most keyword queries would inherently involve some ambiguity, to a lesser or greater extent, and hence can benefit from the promises of result diversification. As this carries diversification from a niche operation to a widely used everyday task for large-scale search engines, the need for efficient and scalable algorithms becomes inevitable. Advances are required in two areas: first, the computational complexity of diversity algorithms needs to be reduced to fit in the tight budget of online query processing (usually a few hundred milliseconds), and second, these algorithms need to be adapted to the computing cluster architecture established for search engines.

Our contributions in this article are thus twofold: first, we improve the efficiency of a state-of-the-art implicit result diversification algorithm based on the desirable facility placement principle (from Operations Research) solved by a Greedy Local Search (GLS) heuristic [Zuccon et al. 2012]. Recently, this algorithm has been shown to have an impressive effectiveness for identifying relevant and novel top-k results, but its quadratic cost with the number of candidate documents renders this algorithm impractical for real-world usage. We propose simple yet effective optimizations that employ preclustering of the candidate documents for improved efficiency (i.e., linear with the number of candidate documents) without sacrificing the effectiveness. In a practical setting where the top-10 (or 20) results are selected from a candidate set of a few hundred (or thousand) documents, our optimized algorithms, so-called C-GLS and C^2-GLS, can reduce the online query diversification cost by more than 80%, and for some cases, up to 97%.

As a second contribution, we turn our attention to incorporating the diversification algorithms into a large-scale search system that would typically operate on a cluster of thousands of machines. While diversification algorithms in the literature are extensively evaluated in terms of their effectiveness, the impact of the distributed architecture on which they need to operate has not been addressed yet. In contrast, the effectiveness and efficiency of the diversification algorithms may also depend on the architecture and, more specifically, the layer where the actual diversification is realized. We introduce two possible strategies, broker-based and node-based diversification, and identify the potential effectiveness and efficiency tradeoffs for both implicit and explicit diversification algorithms. To be comparable with the previous studies in the literature, our strategies are evaluated using the standard experimental framework employed in the TREC Diversity Tasks in 2009 and 2010. To the best of our knowledge, our contribution in this direction is pioneering, as there exists no earlier work in the literature that investigates the diversification performance on top of a distributed architecture.

The rest of the article is organized as follows. In Section 2, we first provide a review of search result diversification algorithms and then outline the principles of query processing techniques in widely employed distributed search systems. Next, we briefly summarize the GLS-based algorithm from Zuccon et al. [2012] and then introduce our

more efficient variants in Section 3. Section 4 is devoted to two distributed diversification strategies, namely, broker-based and node-based diversification. In Section 5, we experimentally evaluate proposed strategies. Finally, we summarize our key findings in Section 6, and conclude and point out future research directions in Section 7.

2. RELATED WORK

Approaches for search result diversification can be categorized as either *implicit* or *explicit* [Santos et al. 2010a]. Given a set of candidate documents retrieved for a query, implicit methods aim to discover the possible different aspects from these documents in an unsupervised manner. In contrast, explicit diversification methods directly model the query aspects, exploiting external knowledge such as manually or automatically assigned query labels or query reformulations. In Sections 2.1 and 2.2, we review algorithms for implicit and explicit search result diversification, respectively. In Section 2.3, we provide a brief outline of distributed search on computing clusters, on which such diversification algorithms are intended to operate.

2.1. Implicit Result Diversification Techniques

The basic assumption of earlier IR ranking, also underlying the probability ranking principle (PRP, Robertson [1977]), assumes that the probability of a document being relevant is independent from the other documents in the result set. However, when viewing the information retrieval task as a decision process where the task is to minimize the risk that a user's information need remains unsatisfied, it becomes clear that a result set with high coverage entails a lower risk than a result set with the most relevant but also very similar documents. Zhai and Lafferty [2006] present a formal model for this risk minimization approach, where they introduce the notion of loss as the degree to which a search effort fails to satisfy a user's information needs. In this model, different diversification objectives become loss functions that probabilistically estimate the loss incurred by a user for a given result set.

Several such objectives are proposed and successfully validated. The use of Maximum Marginal Relevance (MMR) is proposed by Carbonell and Goldstein [1998] and also used in Zhai et al. [2003]. Chen and Karger [2006] maximize the probability of retrieving at least one relevant document by assuming that all documents that are ranked higher than the current one are irrelevant (i.e., as a form of negative feedback). Carterette and Chandar [2009] focus on a finer-grained level of diversification, similar to the examples given in Section 1, and introduce the so-called faceted topic retrieval. In this case, the query facets are identified using LDA and relevance modeling, and then PRP is employed while assessing the coverage of facets by the candidate documents. In a similar fashion, He et al. [2011] first cluster the candidate documents and then select the best clusters to apply diversification to improve the final result quality. Carpineto et al. [2012] compare some well-known implicit diversification techniques (e.g., based on MMR) to those that cluster the candidate documents and select the cluster representatives into the final result list. Besides other metric space-based methods, Gil-Costa et al. [2011, 2013] describe a diversification approach that again clusters the candidate documents and constructs the diversified result where the cluster centroids are placed at the top of the list. Different from all the latter works that essentially employ greedy *best-first search* diversification methods (such as the round-robin strategy or MMR) on top of the document clusters, our work presented here exploits clustering as a preprocessing stage to improve the efficiency and scalability of a greedy *local search* method, which is shown to outperform the best-first search strategies [Zuccon et al. 2012] as discussed later.

Gollapudi and Sharma [2009] show that many diversification objectives can be expressed as obnoxious facility dispersion optimization problems, that is, the task to place

facilities as far away from each other under given constraints and other optimization criteria. Building on this approach, Zuccon et al. [2012] show in that MMR, the Modern Portfolio Theory (MPT, Wang and Zhu [2009]), and the Quantum Probability Ranking Principle [Zuccon and Azzopardi 2010] can be expressed as a facility dispersion problem as well.

Additionally, Zuccon et al. [2012] introduce desirable facility placement (DES) to express a diversity objective. Desirable facility location is the task to locate a given amount of facilities such that the average distance of a customer to the nearest facility is minimized. This problem can be easily mapped to the information retrieval task by viewing the result documents as facility locations and the user's information needs as customer locations. The reported gain in retrieval effectiveness for DES is the highest of all diversification approaches proposed so far, motivating us to use DES as a base for our work. We give a more in-depth description of DES in Section 3.

While the proposed diversity objectives improve the quality of the returned results, diversification comes with a considerable computational penalty. Carterette [2009] has shown that optimal diversification is NP-hard. Consequently, for practical use, approximations and heuristics need to be used for computing a diverse result set. But even in this case, most proposed algorithms have a quadratic complexity, rendering them infeasible for the demanding efficiency constraints of online query processing. The only exceptions we are aware of are Minack et al. [2011] and Drosou and Pitoura [2009], which compute diverse set approximations over continuous data. Our aim is to overcome this limitation and propose a highly efficient diversification approach.

2.2. Explicit Result Diversification Techniques

In the explicit diversification methods, query aspects are modeled explicitly by exploiting some sort of external knowledge. The IA-Select method proposed by Agrawal et al. [2009] assumes that both queries and documents are associated with some categories from a taxonomy, and then achieves diversification by favoring documents from different categories and penalizing the documents that fall into already covered categories.

Radlinski and Dumais [2006] use reformulations of the given query to determine the candidate result set; that is, the candidate set is formed as the union of results for the original query and its reformulations, and then reranked based on the interests of the current user. The xQuAD framework by Santos et al. [2010a] also exploits query reformulations obtained from TREC subtopics and search engines and achieves diversification by identifying the relevance of candidate documents to these subqueries and favoring documents that cover those aspects not yet covered in the current result set. This approach is found to be a top performer in earlier TREC campaigns (e.g., Clarke et al. [2009, 2010]). In Santos et al. [2011], both xQuAD and IA-Select are employed in an intent-aware diversification framework, which specifically takes into account the user query intent (such as navigational, informational, etc.). Chapelle et al. [2011] also take into account query intents that are defined within a shopping scenario and develop a specific ranking function per intent. Capannini et al. utilize query logs to decide when and how query results should be diversified and present the new algorithm OptSelect that takes into account the popularity of query reformulations in the log [Capannini et al. 2011]. Vargas et al. [2012] propose to reformulate xQuAD and IA-Select by incorporating a formal relevance model. Vallet and Castells [2012] further extend the latter two approaches to obtain personalized diversification of search results.

Being inspired from the electoral process used in some countries, Dang and Croft [2012] introduce a novel explicit strategy that takes into account the proportionality of the votes given to the query aspects. In a follow-up work, Dang and Croft [2013] further argue that an explicit aspect need not be represented in the form of a set of

terms, but that considering each such term as a separate aspect is equally useful, or even better. Wu and Huang [2014] propose to combine multiple retrieval results (from different systems) for a given query with the expectation of having a more diversified list at the end. Liang et al. [2014] leverage a similar idea, but in their work, after the data fusion stage, the latent aspects are discovered by the LDA algorithm that also takes into account the retrieval scores of the fused list. Finally, the combined list and discovered aspects are all fed to the explicit diversification algorithm of Dang and Croft [2012]. Note that theirs is not considered as an implicit approach as the initial data fusion stage operates over the result lists that have been already diversified using some explicit strategy. In contrast to the latter two approaches that fuse different retrieval results for the *same query*, Ozdemiray and Altingovde [2015] adopt score-based (such as CombSUM and CombMNZ) and rank-based (such as Borda voting and fusion approaches with Markov chains) aggregation methods to merge the multiple rerankings of candidate documents for *different query aspects*. Ozdemiray and Altingovde [2014] also propose strategies to determine the aspect weights during explicit diversification and present gains in the diversification effectiveness of almost all of the state-of-the-art explicit methods.

Note that while explicit diversification approaches are more effective than implicit methods (e.g., see Capannini et al. [2011]), the best-performing methods (e.g., Santos et al. [2010a] and Capannini et al. [2011]) essentially exploit external sources of information such as query logs, and can identify a query aspect only after the reformulations appear in the logs. This works well for popular queries, but queries in the long tail of the popularity distribution may not benefit from such diversification approaches [Chapelle et al. 2011]. In such cases, implicit diversification methods are still applicable, and this underlies our motivation in this article to improve the efficiency of one of the most effective implicit strategies based on DES and GLS, as we describe in Section 3.

2.3. Distributed Search on Computing Clusters

Nowadays, the usage of computing clusters with thousands of nodes for processing of search requests on large document collections is firmly established. In such an infrastructure, the index needs to be partitioned. In document-based partitioning, each node in a cluster of servers is responsible for an index that is created for a particular subset of documents. In term-based partitioning, each node stores the fragment of an index corresponding to a subset of terms in the collection. The former approach is usually preferred by large-scale systems [Dean 2009], due to its lower usage of resources (like network bandwidth) and better load balancing.

In a document-based partitioning architecture, the query processing is said to be embarrassingly parallel. In the basic query processing workflow (e.g., Ozcan et al. [2012]), the processing task is split up between broker nodes that coordinate query processing, and search nodes that hold index partitions. First, the search engine front end forwards the query to one of the broker nodes in the search cluster. The broker node in turn distributes the query to the search nodes that access their index files, compute partial top-k results, and send them back to the broker. Then the broker determines the global top-k results and contacts document servers to obtain snippets for result presentation. Note that documents can reside at the search nodes or in a separate set of servers [Dean 2009]. Similarly, it is also possible to let each physical node function as both a search node and broker [Ozcan et al. 2012].

While several aspects of query processing are investigated over the distributed search architecture (such as query forwarding [Cambazoglu et al. 2010], caching [Ozcan et al. 2012], etc.), to the best of our knowledge, no previous study addresses how online result diversification algorithms can operate on such an architecture and how their results might be affected according to the layer where the diversification takes place.

3. EFFICIENT GREEDY LOCAL SEARCH FOR DES

As we also outlined in the previous section, casting the problem of search result diversification as a DES problem from Operations Research yields high-quality retrieval results [Zuccon et al. 2012]. In this case, the driving motivation is that each document in the top-k search results represents a facility at a different region and every customer (i.e., each possible query intent) should be sufficiently close to one of these facilities. This goal is achieved by choosing the facility locations (i.e., the top-k documents) such that the total distance to all other documents is minimized. As usual for diversification, this coverage criterion is balanced with document relevance to determine the final result set. Formally, the optimal set of top-k results for a query q is given by

$$S^* = \arg\min_{\substack{S \subset D \\ |S|=k}} f(S) \tag{1}$$

$$f(S) = -\lambda \sum_{d \in S} r(d) + (1-\lambda) \sum_{d' \in D \setminus S} \left(\min_{d \in S} w(d, d') \right), \tag{2}$$

where D is the set of candidate documents (i.e., an initial retrieval result for q), $f(S)$ the diversification objective function, $r(d)$ the relevance score of document d for query q, and $w(d, d')$ the distance between two documents d and d'.

Zuccon et al. [2012] proposed an approximate solution for computing DES using Greedy Local Search (GLS, see Algorithm 1). GLS starts with placing the most relevant k documents of N candidate documents (denoted with set D) into the set S, that is, top-k search results. At each round of the algorithm, the documents in S are replaced with documents that are not in S, to optimize the objective function score. If the latter score does not change anymore, the algorithm terminates, as it has reached a global or local minimum. The authors have also shown that framing result diversification approaches such as MMR, MPT, and QPRP into DES and using the GLS approximation is quite effective, and indeed yields considerably better diversification performance than applying the traditional Best First Search heuristic.

ALGORITHM 1: GLS: Diversification as DES using Greedy Local Search [Zuccon et al. 2012].

```
Input: D, k, f
Output: S
S ⇐ {d_1, ..., d_k}
repeat
    for d ∈ S do
        for d' ∈ D\S do
            S' ⇐ (S\{d}) ∪ {d'}
            if f(S') < f(S) then
                S ⇐ S'
            end
        end
    end
until S does not change
```

While exhibiting an impressive diversification performance, GLS is computationally expensive. Each round of Algorithm 1 requires swapping each document in S with each document in $D\setminus S$, which means $k \times N$ calls for computing the objective function, $f(S)$. As shown in Equation (2), the computation of $f(S)$ also requires finding the closest document in S to each document in $D\setminus S$, and this costs another $N \times k$ operations to

compute pairwise distances. Thus, in total, the computational complexity of a single round of the algorithm is $O(N^2k^2)$. Assuming that the algorithm converges in I_{GLS} rounds, the overall complexity is $O(N^2k^2I_{GLS})$.

Note that if the distance $w(d_i, d_j)$ is computed on the fly, this cost would further increase by the complexity of the document similarity measure (e.g., for the cosine-based similarity $O(\min(|d_i|, |d_j|))$. Therefore, we assume that all pairwise distances among the candidate documents (typically top-100 or top-1,000) are computed once (in a preprocessing stage) and then cached till the end of processing (as in Minack et al. [2011]). In Table I, we provide the breakup of CPU and memory consumption for this latter case (i.e., with cached distances). Given that there is no bound on the convergence of the algorithm and it might take many rounds until it converges (as we illustrate in our experimental evaluations), the algorithm is expensive for practical scenarios.

Clustering-GLS (C-GLS) algorithm. In this article, we propose to improve the efficiency of the GLS solution by employing clustering to approximate the distance computation stage for the objective function computation (see Algorithm 2). In particular, we form a clustering of the documents in D such that each document falls into a single cluster C with centroid $ctr(C)$. Instead of storing pairwise distances, $w(d_i, d_j)$, we store for each document the distance to all cluster centroids, $w(d_i, ctr(C_j))$. Then, while computing the distance of documents that are in $D\backslash S$ to those in S, instead of considering every single document in $D\backslash S$, we only consider cluster centroids, that is, as an approximation of the document space $D\backslash S$ (compare Figure 1(a) and (b)). As our result set can represent at most k different query intents, it is sufficient to split D into k different clusters C_i. With $\mathcal{C} = \{C_1, \ldots, C_k\}$, the cluster-based objective function becomes

$$f(S) = -\lambda \sum_{d \in S} r(d) + (1 - \lambda) \sum_{C \in \mathcal{C}} \left(\min_{d \in S} w(d, ctr(C)) \right). \tag{3}$$

As Algorithm 2 shows, computing this objective function has a complexity of $O(k^2)$ instead of $O(k \cdot N)$. Since the objective function is called $k \times N$ times, the overall complexity of the algorithm becomes $O(Nk^3)$ (per round).

In the clustering stage, we essentially employ the k-means clustering algorithm. This indeed is a natural choice for the GLS problem; as already pointed out in Zuccon et al. [2012], for $\lambda = 1$, the objective function (Equation (2)) leads to k-medoids clustering of D as a result. In our evaluations, we also experiment with a single-pass approach

ALGORITHM 2: Objective Function for C-GLS

```
Input: S, λ
Output: score
relscore ⇐ divscore ⇐ 0
for d ∈ S do
    relscore ⇐ relscore − λr(d)
end
for C ∈ C do
    minDist ⇐ 1
    for d ∈ S do
        minDist ⇐ min(minDist, w(ctr(C), d))
    end
    divscore ⇐ divscore + (1 − λ)minDist
end
score ⇐ relscore + divscore
```

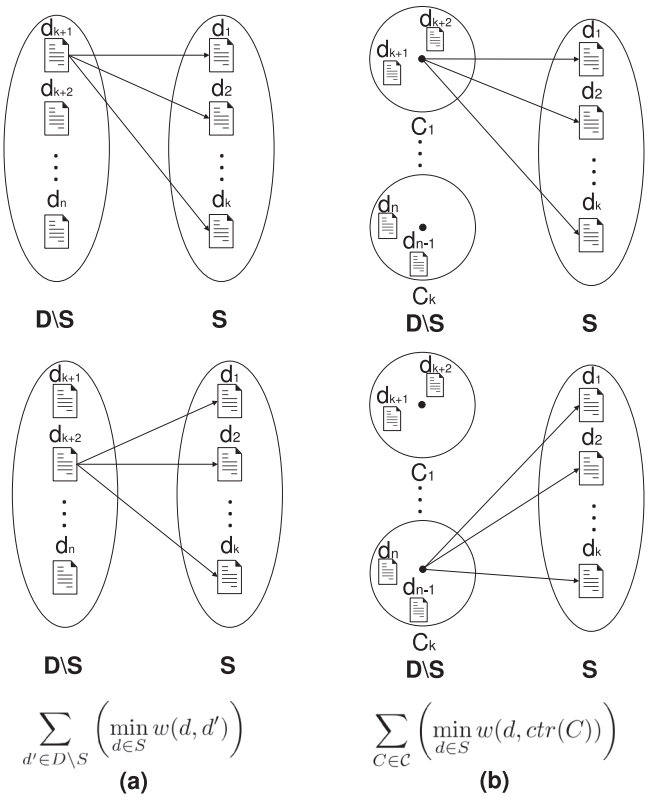


Fig. 1. Objective function computation for (a) GLS (i.e., for each document d' in $D\backslash S$, we compute the distance $w(\bullet)$ to each document d in S) and (b) Clustering-GLS (i.e., for each cluster centroid C in $D\backslash S$, we compute the distance to each document d in S).

based on the *list of clusters* idea introduced by Chávez and Navarro [2005] and adopted for diversification by Gil-Costa et al. [2013].

Clustering^2-GLS(C^2-GLS) algorithm. When choosing reasonable λ values, that is, values that lead to a balance between relevance and diversity, it can be observed that documents with a low relevance score don't end up in the final top-k result set, regardless of the chosen DES variant. In a sufficiently large document collection, it is safe to assume that for several documents matching a specific query intent, the more relevant ones will be preferred (although such a selection might not lead to the perfect result set composition). It is therefore very ineffective to try replacement candidates d' at random (Algorithm 1, Line 6); instead, we should identify the most promising candidates for each query intent and limit the greedy search procedure to this selected set.

We exploit clustering *again* to identify these candidates as well as using the objective function shown in Algorithm 2 (and hence we call this strategy C^2-GLS). We sort the documents of each cluster by relevance and define our candidate set *TopC* as the union of the top-r documents from each cluster C_j. By taking only these most relevant documents into account and with $R = |topC|$, we achieve a complexity of $O(Rk^3)$ (per round), which is a further improvement over $O(Nk^3)$ given that $R \ll N$.

Table I. Complexity of Diversification Algorithms (I_C Denotes the Number of Rounds for Clustering; CPU Complexity Costs for the Actual Diversification Stage Are in Terms of the Number of Distance Computations *per Round*)

Algorithm	CPU (Preprocessing)	CPU (Diversification)	Memory
GLS	$O(N^2)$	$O(N^2k^2)$	$O(N^2)$
C-GLS	$O(NkI_C)$	$O(Nk^3)$	$O(Nk)$
C^2-GLS	$O(NkI_C)$	$O(Rk^3)$	$O(Nk)$

Table I shows the cost breakup of the so-called Clustering-GLS (C-GLS) and Clustering^2-GLS (C^2-GLS) strategies with the k-means clustering algorithm. The pairwise distance computation cost now reduces to $O(NkI_C)$, where I_C is the number of iterations for k-means clustering. Note that the clustering algorithm is expected to converge in a few iterations as N is at most a few thousands in practical settings. More crucially, since the computation of $f(S)$ now involves comparison of cluster centroids to the top-k documents, it has $O(kN)$ complexity, reducing the overall complexity of Algorithm 1 from $O(N^2k^2)$ to $O(Nk^3)$ and $O(Rk^3)$ (per round) for the C-GLS and C^2-GLS strategies, respectively. Given that k is usually one or two orders of magnitude smaller than N (i.e., $k = 10$ while N is either 100 or 1,000), this is a crucial improvement in terms of efficiency. Furthermore, our C-GLS and C^2-GLS strategies have a memory footprint linear in the document collection size, that is, $O(Nk)$ instead of $O(N^2)$, as we only need to store for each document the distance to the cluster centroids. Last but not the least, the use of cluster centroids also induces a smoothing effect on the distance values, and the proposed algorithms converge in a smaller number of rounds (as shown in the experiments).

4. DISTRIBUTED DIVERSIFICATION

While the effectiveness and efficiency of the diversification algorithms have received serious attention in the literature, to the best of our knowledge, the architecture over which these algorithms would be employed has not been considered. In this section, we introduce two distributed diversification approaches and discuss their pros and cons in a realistic setup.

In practice, all large-scale search engines operate on a number of geographically distributed computing clusters, each with tens of thousands of servers (nodes).[1] Each node in a cluster stores an inverted index that corresponds to a randomly partitioned subset of the entire collection, as well as other data statistics (such as document lengths, etc.) required for query processing. A search cluster has one or more broker nodes that are responsible for forwarding the query to all search nodes, gathering the top-k partial results from these nodes, and merging the partial results to obtain global top-k results.

In this section, we investigate diversification of search results in a typical search cluster as described earlier. We envision that result diversification for a given query can take place either in the broker node or in the indexing nodes and define corresponding strategies as follows:

Broker-based diversification (BB-Div). This is the straightforward case where the broker runs a diversification algorithm once it collects and merges all the partial results from the search nodes (see Algorithm 3). Assuming that each of the P nodes returns the top-k (partial) results computed on its local collection D_P, the broker will have a set of $P * k$ documents. Then, it can apply the diversification algorithm on these $P * k$ documents or further restrict this initial set to the top-N results with the highest relevance scores (for the sake of efficiency). We opt for the latter option as it is

[1]In this work, we focus on distributed query processing within a single search cluster that can usually capture a replica of the entire web index.

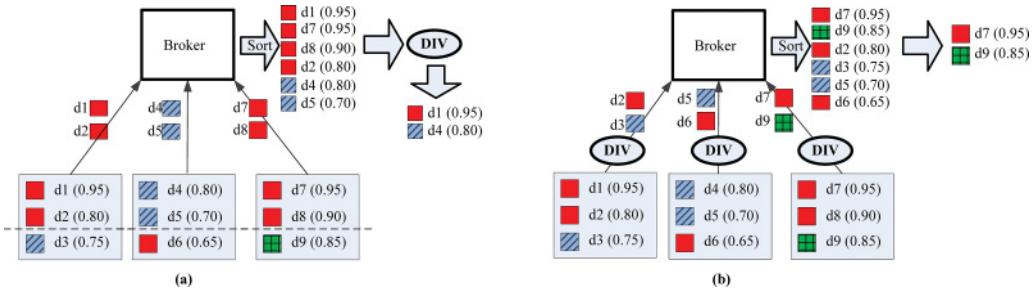


Fig. 2. A toy example for (a) broker-based and (b) node-based diversification strategies (all nodes and the broker return top-2 results). Note that the final diversified results differ.

ALGORITHM 3: BB-Div

Node $D_{P,k} \Leftarrow Top(D_P, k)$ `// Compute local top-`k` result`

Broker
for $P \in \mathcal{P}$ **do**
  $D \Leftarrow D \cup D_{P,k}$ `// Merge top-`k` from nodes`
end
$D \Leftarrow$ `Top`(D, N) `// Keep top-`N` results`
$S \Leftarrow$ `Diversify`(D, k)

more likely in practice (otherwise, for a typical cluster of 50,000 servers, the candidate set would be huge). Besides, as we discuss in the next section, fixing the candidate set size allows a fair comparison of the distributed diversification approaches in our simulations. Thus, in Algorithm 3, the broker first calls the function $Top(.)$ to construct the candidate set D of size N, and then invokes the diversification algorithm to obtain the final result set S (of size k). Note that if the diversification algorithm imposes an order on the documents selected into S (e.g., by generating a score), the results are presented in this order. Otherwise, for the algorithms (like DES) where the output S is a set but not a ranked list, the final top-k results are still presented in the order of their initial relevance scores. The BB-Div strategy is illustrated in Figure 2(a) for a toy scenario with the latter assumption. In this scenario, d_1 and d_4 are sent to the broker as they achieved the highest relevance scores at their nodes; at the end of the diversification applied at the broker, we assumed both appeared in the final result, because they are both relevant and also different from each other (as denoted by different color codes and textures in the figure).

The advantage of the broker-based diversification strategy lies in its simplicity and practicality; it can be directly coupled with an existing search system. On the other hand, there are a couple of drawbacks that should be taken into account in terms of the efficiency and effectiveness of the diversification: First, as each server returns its top-k results (where k is usually 10), the candidate list to be diversified can miss results that are related to different interpretations of a query, especially if a particular interpretation dominates the result set. For instance, for the query *Java*, partial top-10 results from each node can rank only those documents related to Java as a programming language, so that diversification in the broker by setting N to 100 or 1,000 (or even using all $P * k$ results) might be suboptimal, simply because the candidate set is not large enough to encounter the documents that cover different query intents. We illustrate this case also in Figure 2(a): the third node returns its top-2 results (d_7 and d_8), both of which have the same pattern/color, and hence misses the third result, d_9, which is different from the first two documents. This might be remedied by increasing

the partial result set size if the query can be identified as ambiguous beforehand, which is a nontrivial issue on its own (e.g., see Capannini et al. [2011]).

As a further yet related efficiency problem, the broker-based diversification requires the document vectors for the top-N documents to be transferred to the broker node, as most of the implicit and explicit diversification approaches need the document vectors. For instance, all implicit diversification approaches we focus on in this article would need vectors for pairwise distance computations. Indeed, even for an explicit diversification technique like xQuAD [Santos et al. 2010a], the document vectors may still be required to compute the similarity of explicit query intents to each candidate document in the broker.[2] Moving document vectors among the nodes incurs some network cost even if each document resides in a different node and transfers can be processed in parallel. This cost would further increase for larger values of N and/or partial result set size. In the next section, we provide a detailed analysis of the network cost for this scenario. Note that once document vectors are transmitted to the broker, the pairwise distances (or coverage of explicit query aspects) have to be computed on the fly, since the top-N candidate documents will be almost always compiled from different nodes (due to partitioning of the collection uniformly at random to the nodes), and hence, these distances cannot be computed a priori.

Node-based diversification (NB-Div). An alternative strategy is applying the result diversification in each search node and combining the partial top-k results that are already diversified (see Algorithm 4). In this case, each of the P nodes first obtains the candidate set $D_{P,N}$ (of size N) on its local collection D_P and then calls the diversification algorithm to select the diversified top-k set into $S_{P,k}$. The generated partial results are simply merged at the broker based on their relevance scores (or diversification scores, if available) to create the global top-k answer. This case is illustrated in Figure 2(b). In this case, the top-ranked document d_1 is eliminated at the node on which it is hosted, as its successors in the list d_2, which has a lower relevance score, is assumed to have a much higher dissimilarity to the third document in the list (d_3) (to illustrate how NB-Div differs from BB-Div), and hence, these two documents, d_2 and d_3, are sent to the broker.

ALGORITHM 4: NB-Div

```
Node   D_{P,N} ⇐ Top(D_P, N)
       S_{P,k} ⇐ Diversify(D_{P,N}, k)
Broker for P ∈ 𝒫 do
       |  S ⇐ S ∪ S_{P,k}                    // Merge Div-k from nodes
       end
       S ⇐ Top(S, k)                        // Keep top-k results
```

As the indexed documents can be stored along with the index at each node [Ozcan et al. 2012], the node-based diversification strategy has no transfer costs to access the documents. Furthermore, similarities between a certain subset of documents (such as those with the highest PageRank scores or highest frequency in the query results) can be precomputed and cached, to be used for different queries. This strategy also allows considering a deeper pool of documents while still executing the diversification algorithm for the top-N candidates. That is, for a fixed value of N, the cost of running

[2]Alternatively, each query aspect has to be sent back to the nodes and the relevance scores for each such aspect and every document in the candidate set need to be computed using the inverted index. We further investigate the efficiency of this approach in the next section.

a diversification algorithm at the broker (excluding the document transfer times as discussed earlier) is the same as running the same algorithm at the nodes, but since each node has its own top-N set, it is more likely to encounter documents with diverse interpretations (e.g., in the *Java* example earlier). However, this might also work in the reverse direction: when each partial result set is locally diversified, some of the results in partial top-k can indeed have a very low global rank in terms of relevance; that is, some potentially more relevant results might be sacrificed at each node for diverse yet very low-ranked results. This is an interesting tradeoff that might require adaptive solutions based on the query properties, such as the degree of ambiguity [Santos et al. 2010b].

A particular disadvantage of this algorithm is the lack of global knowledge during the diversification stage. That is, since each list is diversified locally, the relative order of diverse intents might be similar (e.g., all top-1 documents from each node might be relevant for the programming language intent for the *Java* query) and the simple relevance-based merging at the broker can easily end up with less diversification than desired.

In the following section, we evaluate both the GLS-based implicit diversification algorithms and xQuAD, a well-known explicit diversification algorithm, in a distributed environment; observe how the aforementioned tradeoffs affect their effectiveness and efficiency; and draw conclusions for designing and employing diversification algorithms in large-scale search systems.

5. EXPERIMENTS AND RESULTS

5.1. Experimental Setup

Document collection. In this article, following the practice in the diversification field, we use one of the largest available web datasets with expert assessments, namely, the Clueweb09 Part-B collection[3] that includes around 50 million web pages. We index the collection using the Zettair IR system,[4] with the "no stemming" option. All stop words and numbers are included in the index, yielding a vocabulary of around 160 million terms.

Queries and initial retrieval. We use the query topics and relevance judgments released for TREC 2009 and 2010 Diversity Task[5] that include 50 and 48 topics, respectively. For each topic, there are a number of predefined subtopics (between one and eight) that are used during the judgment process, so that each document annotated as relevant is also associated with one or more subtopics from this list. The TREC queries are generated automatically using the title fields of the topics.

Additionally, we employ a third and larger set (denoted as *Q1000*) that includes 1,000 queries that are sampled from the AOL 2006 query log [Pass et al. 2006] and can retrieve nonempty results over our collection. This latter set is only used for evaluating efficiency but not effectiveness, as there are no available relevance judgments for the queries in this set. A similar approach is also followed in Gil-Costa et al. [2013] for evaluating diversification efficiency.

We used our own IR system (also employed in Ozdemiray and Altingovde [2015]) to process the queries over the index. As the relevance model, we use a variant of the well-known Okapi-BM25. We set the free parameters k_1 and b to 1.2 and 0.75, respectively.

[3]http://www.lemurproject.org/clueweb09.php/.
[4]www.seg.rmit.edu.au/zettair/.
[5]http://trec.nist.gov/data/web09.html.

Document similarity computation. We employ the usual cosine similarity of document vectors with tf-idf weights for computing the similarity $s(d, d')$ between two documents d and d'.

Diversification algorithms. In addition to GLS and its proposed variants the C-GLS and C^2-GLS methods, we involve List of Clusters Diversification (LCD) and xQuAD as further baselines from the implicit and explicit diversification literature, respectively. We summarize their implementation and parameters as follows.

GLS, C-GLS, and C^2-GLS. As discussed in the previous sections, we focus on desirable facility placement (DES) approaches and evaluate three approximate solutions: GLS from Zuccon et al. [2012], C-GLS, and C^2-GLS. In Zuccon et al. [2012], it is also shown that well-known diversification techniques like MMR, QPRP, and MPT can be all modeled within the DES framework, by choosing corresponding relevance and distance measures in Equation (2). Their experiments further reveal that the best-performing strategy in the DES+GLS framework is MPT, outperforming MMR and QPRP. Therefore, in this article, we instantiate the objective function in Equations (2) and (3) for the MPT approach as in Zuccon et al. [2012] for all three diversification algorithms. So, $r(d)$ is set to the document's BM25 score (normalized per query by dividing relevance scores by the maximum BM25 score for a given query) and

$$w(d, d') = 2 \times b \times \sigma^2 \times w_{d'} \times (1 - s(d, d')), \tag{4}$$

where $w_{d'}$ is the importance weight of the rank of d' in S, computed in the same fashion to discounting factors of the nDCG metric [Järvelin and Kekäläinen 2002], and b and σ^2 are treated as parameters for MPT following the practice in Zuccon et al. [2012]. We experiment with values of b in the range $[1, 10]$ with increments by 1, and σ^2 in the range $[10^{-6}, 10]$ incremented by orders of 10. Finally, for our C^2-GLS algorithm, we set r to 5; that is, we consider the top-5 most relevant documents of each cluster in the algorithm.

LCD. As a further baseline, we involve another implicit approach, namely, the List of Clusters Diversification (LCD) algorithm introduced by Gil-Costa et al. [2013]. As discussed in the related work section, this latter work also aims to improve the efficiency of implicit diversification and proposes three different methods that utilize metric spaces for efficient computation of the pairwise distances. Among these methods, we choose LCD as the baseline for two reasons. First, the LCD method applies clustering for diversification, so it is methodologically close to our C-GLS and C^2-GLS methods that also employ clustering, albeit as a preprocessing stage. Second, according to their experimental results, when the underlying retrieval model is BM25 (which is also the case in our article), the best-performing method is LCD (please see Table I in Gil-Costa et al. [2013]).

In a nutshell, the LCD algorithm works as follows. The document with the highest retrieval score is selected as the center of the first cluster, and the distance of all the other documents to the center is computed. Then, a fixed number, say, r, of the documents that are nearest to this first center are assigned to its cluster and removed from the candidate set. The next cluster center is chosen as the one that maximizes the sum of distances to the previous center(s), and again, the r-nearest documents are assigned to this cluster. The process continues until all the candidate documents are clustered and results in a List of Clusters (LC) [Chávez and Navarro 2005]. Finally, the LCD algorithm ranks the cluster centers at the top of the final result list (in the order of the discovery) and then lists their members as blocks (in the order of relevance) in the corresponding order of their centers. In our implementation of LCD, we set the cluster size, r, as the value that yields the highest diversification performance for each query topic set.

xQuAD. While our cluster-based solutions are intended to improve the efficiency of GLS as an effective implicit diversification algorithm, the strategies proposed for a scalable distributed architecture are independent of the diversification algorithm; that is, any kind of diversification approach can be incorporated into BB-Div and NB-Div strategies. Therefore, we do not restrict our evaluations over the distributed architecture to only implicit diversification algorithms, but also employ xQuAD as a further baseline. We believe that xQuAD is a good representative of the class of explicit diversification algorithms, as it is placed among the top performers in the diversity tasks of TREC from 2009 to 2012. Furthermore, most recent explicit diversification algorithms, namely, IA-Select [Agrawal et al. 2009], xQuAD [Santos et al. 2010a], PM2 [Dang and Croft 2012], and ranking aggregation-based methods proposed in Ozdemiray and Altingovde [2015], all need to compute the relevance scores of the candidate documents for the query aspects, which means that they would incur the same cost in terms of the network communication in a distributed setup. Hence, xQuAD serves as an adequate representative also from the latter perspective.

We implemented xQuAD following the common practice in the earlier works [Santos et al. 2010a; Dang and Croft 2012; Ozdemiray and Altingovde 2015] to simulate an ideal setup, that is, with the perfect knowledge of the query aspects. To this end, explicit query aspects are generated using the official TREC subtopic descriptions for each topic. In our experiments, we test all values of the tradeoff parameter λ in the [0,1] range with a step size of 0.01 and report the results for the λ that optimizes α-NDCG@20. Note that, in a more realistic scenario, the query aspects could be obtained from the query suggestions of a search engine [Santos et al. 2010a] and λ values could be learned within a machine-learning setup as in Santos et al. [2010b]. However, we prefer to use the best-performing setup for xQuAD as our goal here is not comparing the effectiveness of implicit and explicit diversification algorithms, but providing an in-depth investigation of how these algorithms perform when they are incorporated into our distributed diversification strategies.

Clustering algorithms. As discussed in Section 3, we essentially employ the standard k-means clustering algorithm for the preprocessing in the C-GLS and C^2-GLS methods. As a further baseline, we employ the LC approach [Chávez and Navarro 2005] described before in the context of LCD, as it is a single-pass algorithm and found to be efficient when the target number of clusters is small [Gil-Costa et al. 2013].

Evaluation metrics. We use the evaluation software *ndeval* provided as part of the TREC Diversity Task. We report effectiveness results using α-NDCG [Clarke et al. 2008], ERR-IA [Chapelle et al. 2009], and subtopic recall (S-recall) [Zhai et al. 2003], which are widely used in the literature. To evaluate efficiency, we report the elapsed time for the preprocessing (i.e., pairwise distance computations for GLS, and the cost of k-means or LC clustering for the proposed methods C-GLS and C^2-GLS) and actual diversification stages, per query. To facilitate the reproducibility of our results by others, we also report machine-independent measures, namely, the average number of rounds till convergence, average number of times for invoking the objective function (in each round), and average number of lookups for pairwise distances (between two documents or between a document and a cluster centroid) in each call of the objective function. Finally, for the distributed diversification setup, we evaluate the performance in terms of the network communication cost, namely, total volume of the transferred data (in bytes) and transfer time (in milliseconds).

5.2. Evaluation of the C-GLS and C^2-GLS

As our first goal is improving the efficiency of the GLS solution for the DES approach in result diversification, we compare the effectiveness and efficiency of GLS to the C-GLS and C^2-GLS strategies that are proposed in this article. In this set of experiments, we

Table II. Retrieval Effectiveness of the Diversification Algorithms. Type Field Denotes Implicit or Explicit Diversification

			ERR-IA			α-NDCG			S-recall		
TREC 2009											
Algorithm	Preproc.	Type	@5	@10	@20	@5	@10	@20	@5	@10	@20
Baseline		None	0.120	0.134	0.142	0.183	0.217	0.250	0.256	0.346	0.414
LCD		Imp.	0.120	0.134	0.142	0.183	0.217	0.250	0.256	0.346	0.414
GLS		Imp.	0.150*	0.162*	0.168*	0.228*	0.248	0.269*	0.286*	0.347	0.391
C-GLS	k-means	Imp.	0.155*	0.164*	0.171*	0.233*	0.248*	0.272*	0.316*	0.353	0.420
C^2-GLS	k-means	Imp.	0.153*	0.163*	0.169*	0.230*	0.247*	0.270	0.313*	0.381	0.435
C-GLS	LC	Imp.	0.156*	0.169*	0.175*	0.227*	0.250*	0.276*	0.299*	0.351	0.424
C^2-GLS	LC	Imp.	0.146*	0.158*	0.165*	0.220*	0.244*	0.270	0.291	0.368	0.426
xQuAD		Exp.	0.158	0.175	0.183	0.226	0.264	0.298*	0.288	0.390†	0.460*†
TREC 2010											
Baseline		None	0.141	0.156	0.165	0.182	0.214	0.245	0.276	0.377	0.452
LCD		Imp.	0.141	0.156	0.165	0.182	0.214	0.249	0.276	0.377	0.478
GLS		Imp.	0.138	0.156	0.166	0.184	0.221	0.255	0.276	0.386	0.483
C-GLS	k-means	Imp.	0.168*†	0.187*†	0.195*†	0.209*†	0.249*†	0.278*†	0.287	0.408	0.507*
C^2-GLS	k-means	Imp.	0.164*†	0.175	0.184*	0.210†	0.231*†	0.265*†	0.310	0.385	0.506
C-GLS	LC	Imp.	0.151*	0.162*	0.169*	0.208*	0.233*†	0.266*†	0.278	0.356	0.432
C^2-GLS	LC	Imp.	0.159*†	0.176*†	0.185*†	0.202*†	0.238*†	0.270*†	0.298	0.395	0.502
xQuAD		Exp.	0.177	0.194	0.200	0.234	0.272*†	0.293*†	0.390*†	0.488*†	0.526*†

The superscripts (∗) and (†) denote a statistically significant difference at the 0.05 level from the baseline and GLS algorithms, respectively. The xQuAD algorithm that utilizes explicit knowledge of aspects is included only for reference, to be considered in the evaluation of the distributed framework.

assume a single-node architecture as is typical in the literature; that is, for a given query, we retrieve the top-N documents (D set) from the entire collection and then rerank these candidate documents to obtain the final top-k results (S set) using GLS, C-GLS, and C^2-GLS with MPT instantiation. We set $k = 20$ and $N = 100$.

Effectiveness evaluation. In Table II, we compare the effectiveness of the baseline and proposed diversification algorithms and the standard BM25 baseline, that is, retrieving the top-20 most relevant documents without any diversification, for the TREC 2009 and 2010 topic sets. First of all, we observe that the nondiversified baseline using BM25 is much better than the language model baseline reported in Zuccon et al. [2012]. For instance, while NDCG@5, @10, and @20 are found to be 0.105, 0.150, and 0.207 in Zuccon et al. [2012], our baseline yields 0.183, 0.217, and 0.250, respectively. Capannini et al. [2011] also report values closer to ours, namely, 0.190, 0.212, and 0.240, respectively, for NDCG@5, @10, and @20 while they employed the Divergence from Randomness (DFR) model for the retrieval. We believe that these differences can be caused by different choices that might be made during document processing, indexing, and/or query processing, as each of these stages involves several parameters that can affect the final result.

Table II shows that xQuAD, in its best-performing setup, can beat both the standard baseline and implicit diversification approaches, a finding that confirms the previous results in the literature. As we have pointed out, our goal in this work is improving the efficiency of GLS as an implicit diversification solution, as such approaches are viable/helpful for a wide range of practical cases where the query aspects cannot be known in advance. Therefore, here we provide the effectiveness values for nondistributed xQuAD only for reference, to enable the analysis of its performance within the distributed framework in the following section.

For the implicit strategies, our experiments reveal that GLS can outperform the standard baseline, although the gains are not as pronounced as those reported in Zuccon et al. [2012]. In contrast, the alternative implicit diversification baseline, LCD, can

Table III. Diversification Performance (α-NDCG@20) Versus the Number of Clusters (k)

Topic set	Algorithm	Preprocessing Algorithm	No. of Clusters (k) 5	10	15	20	25
TREC'09	C-GLS	k-Means	0.266	0.266	0.270	**0.272**	0.267
	C^2-GLS	k-Means	0.263	0.270	0.258	**0.270**	0.267
	C-GLS	LC	0.274	**0.276**	0.270	**0.276**	0.272
	C^2-GLS	LC	0.259	0.265	0.268	**0.270**	0.258
TREC'10	C-GLS	k-Means	0.264	0.247	0.261	**0.278**	0.259
	C^2-GLS	k-Means	0.258	0.255	0.256	**0.265**	0.256
	C-GLS	LC	0.265	**0.266**	0.262	**0.266**	0.263
	C^2-GLS	LC	0.258	0.266	0.266	**0.270**	0.260

improve the nondiversified BM25 ranking only for a couple of cases for the TREC 2010 topic set. Note that the latter finding is not far from the earlier results reported in Table I of Gil-Costa et al. [2013], where an improvement of at most 0.0067 is observed for LCD using similar metrics at the cutoff value 20. These results also justify our goal of improving the efficiency of GLS in this article, due to its high effectiveness as an implicit diversification algorithm.

Our proposed methods, C-GLS and C^2-GLS, are evaluated using two different clustering algorithms, namely, k-means and LC. Table II shows that, especially for the cases with the k-means algorithm, our methods have no adverse impact on the diversification effectiveness, and indeed, at almost all rank cutoffs, both of the proposed strategies are better than the baseline and perform comparable to (or, especially for the TREC 2010 set, even better than) the GLS algorithm. The latter finding on the TREC 2010 dataset can be explained by the smoothing effect of the clustering strategies; that is, our algorithm avoids considering very diverse candidates with very low relevance scores. We also observe that using k-means for the preprocessing yields higher effectiveness than using LC for the majority of the cases (e.g., C^2-GLS with LC is inferior to the version with k-means for the TREC 2009 set for almost all metrics). This is also expected, as the multipass nature of the k-means algorithm may yield a better clustering of the candidate documents.

Impact of the parameters N and k. We also analyze the sensitivity of our methods for the candidate result set size (N) and the number of clusters (k). For the former parameter, earlier studies using a similar TREC setup report that smaller values (such as 50 or 100) are better [Dang and Croft 2012, 2013; Ozdemiray and Altingovde 2015]. A possible explanation for this observation is that the documents that are ranked too low are more likely to be irrelevant yet diverse, and hence their inclusion in the final result reduces the effectiveness. In our case, we also experimented for $N \in \{50, 100, 200, 500, 1000\}$ and found out that a candidate set of 100 consistently yields the best scores for more than half of the cases on the TREC 2009 and 2010 sets for GLS, as well as the C-GLS and C^2-GLS methods (with the k-means clustering).

Another important parameter is the number of clusters, k, that is intuitively set to the final result set size (i.e., 20), as it is possible to represent at most 20 different clusters (and, equivalently, intents) in the final query result. We also experimented for $k \in \{5, 10, 15, 20, 25\}$. In Table III, we only report the results in terms of the α-NDCG metric at the cutoff value of 20, as the results for the other metrics exhibit completely similar trends and are discarded for the sake of brevity. These experiments reveal that for the majority of the cases, setting k as 20 yields the best effectiveness score in this setup, a finding that justifies our intuitive choice.

Finally, we investigate the stability of the performance of our methods when the k-means algorithm is initialized differently, due to random selection of the seeds (note

Table IV. Statistics of the Diversification Performance for 10 Different Clustering Structures Produced by the k-Means (GLS Scores Are Provided for Easy Comparison)

		ERR-IA			α-NDCG			S-recall		
TREC 2009										
Algorithm		@5	@10	@20	@5	@10	@20	@5	@10	@20
GLS		0.150	0.162	0.168	0.228	0.248	0.269	0.286	0.347	0.391
C-GLS	AVG	0.152	0.165	0.171	0.225	0.248	0.273	0.285	0.350	0.416
	MIN	0.143	0.159	0.166	0.213	0.243	0.269	0.263	0.337	0.380
	MAX	0.161	0.172	0.178	0.239	0.255	0.282	0.316	0.365	0.433
	STDEV	0.006	0.005	0.005	0.009	0.004	0.005	0.011	0.009	0.019
C^2-GLS	AVG	0.154	0.165	0.172	0.225	0.247	0.272	0.291	0.355	0.420
	MIN	0.150	0.161	0.169	0.218	0.239	0.264	0.272	0.335	0.390
	MAX	0.161	0.170	0.178	0.235	0.252	0.283	0.319	0.389	0.446
	STDEV	0.003	0.003	0.003	0.005	0.004	0.005	0.008	0.010	0.017
TREC 2010										
GLS		0.138	0.156	0.166	0.184	0.221	0.255	0.276	0.386	0.483
C-GLS	AVG	0.159	0.174	0.183	0.203	0.234	0.267	0.295	0.385	0.498
	MIN	0.150	0.162	0.169	0.183	0.221	0.254	0.247	0.334	0.432
	MAX	0.171	0.182	0.194	0.219	0.247	0.278	0.342	0.424	0.530
	STDEV	0.005	0.004	0.002	0.009	0.006	0.002	0.026	0.012	0.015
C^2-GLS	AVG	0.166	0.181	0.188	0.211	0.241	0.265	0.312	0.389	0.472
	MIN	0.162	0.175	0.184	0.202	0.231	0.259	0.286	0.373	0.432
	MAX	0.170	0.188	0.194	0.216	0.253	0.271	0.333	0.403	0.506
	STDEV	0.003	0.004	0.003	0.004	0.006	0.003	0.011	0.010	0.006

that this is not an issue for LC as it always chooses the same seeds in the same order in a deterministic manner). To this end, for each algorithm and topic set, we applied the k-means algorithm 10 times, yielding different clustering structures, which are then used in the diversification stage. In Table IV, we present the statistics for the diversification performance of each algorithm with these 10 clustering structures in terms of the minimum, maximum, and average scores for each metric. We also report the effectiveness of the original GLS for each case, which is repeated from Table II to facilitate the comparison. Our findings show that the effectiveness scores of the algorithms are stable, as the standard deviation for each metric is very low. Furthermore, a comparison of the average scores to the corresponding GLS row shows that our methods consistently perform as well as the original GLS on average, even with different clustering structures.

Efficiency evaluation. Being convinced of the effectiveness of our methods C-GLS and C^2-GLS, we turn our attention to their efficiency, which is our main focus here. We report the preprocessing and diversification costs in terms of the CPU processing time. While doing so, we discard the time for generating the candidate result set (D) and retrieving the document vectors, as these stages have the same cost for all compared approaches. Our implementations are single threaded and hence executed on a single CPU, although we use a server with 31 Intel Xeon processors and a total of 32GB of RAM, and running CentOS Linux 6.6 distribution.

Our results in Table V reveal that the implicit diversification baseline, LCD, is extremely fast. Indeed, it is even faster than xQuAD, which is reported here only for the sake of completeness, as most explicit approaches have lower computational complexity due to their prior knowledge of the query aspects (e.g., see Ozdemiray and Altingovde [2015]). However, as discussed before, the effectiveness of LCD is only slightly better than the nondiversified BM25 ranking, rendering it rather useless in a realistic setup. For the GLS algorithm with higher effectiveness, the online diversification stage takes more than 500 milliseconds, which is again impractical.

Table V. Processing Time of the Diversification Algorithms (Per Query) (the Last Column Denotes the Improvement over GLS with Respect to the Total Processing Time)

Topic Set	Algorithm	Preprocessing Algorithm	Preprocessing Time (*ms*)	Diversification Time (*ms*)	Total Time (*ms*)	Impr. over GLS
TREC 2009	GLS		72.081	747.938	820.019	-
	LCD	N/A	-	5.604	5.604	99%
	C-GLS	k-Means	138.244	38.375	177.149	78%
	C^2-GLS	k-Means	138.244	0.486	138.730	83%
	C-GLS	LC	19.104	38.905	58.009	93%
	C^2-GLS	LC	19.104	1.441	20.545	97%
	xQuAD	N/A	-	9.375	9.375	98%
TREC 2010	GLS		92.395	770.687	863.082	-
	LCD	N/A	-	8.175	8.175	99%
	C-GLS	k-Means	146.750	37.908	184.658	79%
	C^2-GLS	k-Means	146.750	0.481	147.231	83%
	C-GLS	LC	21.314	38.366	59.680	93%
	C^2-GLS	LC	21.314	1.703	23.017	97%
	xQuAD	N/A	-	9.591	9.591	98%
Q1000	GLS		70.100	544.776	614.876	-
	LCD	N/A	-	4.899	4.899	99%
	C-GLS	k-Means	132.787	27.123	159.910	74%
	C^2-GLS	k-Means	132.787	0.338	133.125	78%
	C-GLS	LC	16.248	27.244	43.492	93%
	C^2-GLS	LC	16.248	1.100	17.348	97%

Fortunately, Table V demonstrates that our approaches (and especially C^2-GLS) reduce the actual diversification time of GLS by up to three orders of magnitudes (e.g., 747.938 vs. 0.486ms for GLS and C^2-GLS with k-means on the TREC 2009 topic set, respectively). In terms of the preprocessing time, when we employ the k-means algorithm, the clustering overhead of our approaches seems to be larger than the cost of computing all pairwise distances for GLS. However, this is observed in a setup where we use a straightforward implementation of the k-means algorithm (without any efforts for optimization) and the parameters k and N are set to rather close values, that is, 20 and 100, respectively. When we employ LC for the preprocessing stage, the clustering overhead is significantly reduced, but in return for some reduction in the effectiveness (cf. Table II). In practice, some search engines may even prefer to precategorize the documents in its collection according to a taxonomy (as suggested in Agrawal et al. [2009]) for various purposes (like improving the result relevance), and in this latter case the preprocessing cost of clustering can be totally avoided.

Nevertheless, even when the preprocessing times are included, C-GLS (C^2-GLS) with the k-means preprocessing yields an *overall* efficiency improvement of 78% (83%), 79% (83%), and 74% (78%) over GLS for TREC 2009, 2010, and Q1000 topic sets, respectively. Remarkably, the overall processing time for the C-GLS (C^2-GLS) algorithm drops under, respectively, 200 (150) milliseconds, which makes it possible to satisfy the demanding requirements of online query processing in real-life search systems. Furthermore, under heavy workloads, the search engines may even switch to a less effective yet more efficient preprocessing technique as a compromise, such as the LC method, which yields an overall processing time of less than 25ms with C^2-GLS (an improvement of 97% over GLS) for all three topic sets.

In Table VI, we report the counts of key operations to shed light on the efficiency gains provided by our methods. Note that, unlike the processing time, these operation counts are independent of the experimental architecture, and hence, allow a more general comparison among the algorithms. Table VI shows that the proposed approaches

Table VI. Breakup of the Diversification Cost in Terms of the Key Operation Counts (Per Query)

Topic set	Algorithm	Preproc. Algorithm	No. of Rounds	No. of Calls to $f(S)$	Time in $f(S)$	No. of Iterations & Lookups *per Call*	Time for Lookups
TREC 2009	GLS		54.220	8,288.100	741.108	1,600.0	97.936
	C-GLS	k-Means	10.740	1,703.380	34.460	400.0	4.820
	C^2-GLS	k-Means	3.388	23.320	0.473	400.0	0.015
	C-GLS	LC	10.964	1,743.184	35.669	400.0	5.051
	C^2-GLS	LC	8,327	70,469	1.377	400.0	0.076
TREC 20010	GLS		41.400	8,308.020	765.259	1,600.0	118.217
	C-GLS	k-Means	10.313	1,679.300	34.189	400.0	7.371
	C^2-GLS	k-Means	3.958	23.813	0.468	400.0	0.047
	C-GLS	LC	11.313	1,704,854	34.658	400.0	6.784
	C^2-GLS	LC	7.813	66.479	1.647	400.0	0.290
Q1000	GLS		19.875	5,526.915	539.655	1,600.0	109.588
	C-GLS	k-Means	8.534	1,276.450	23.303	400.0	1.793
	C^2-GLS	k-Means	1.034	17.694	0.321	400.0	0.026
	C-GLS	LC	8.605	1,280.213	23.456	400.0	1.855
	C^2-GLS	LC	2.291	51.433	1.031	400.0	0.139

significantly reduce the average number of calls for computing the objective function and number of rounds till convergence, per query. Furthermore, the number of iterations within the objective function per call (which is also equal to the number of distance lookups) is also reduced: as also illustrated in Figure 1, while C-GLS and C^2-GLS make only 400 iterations (and lookups) per call, GLS requires 1,600 iterations.

We also investigate the relationship between the operation counts and diversification time. It turns out that the majority of the diversification time is spent for computing the objective function (cf. Table V). Therefore, the reductions provided by our methods in two ways, namely, in the number of calls for the objective function and number of iterations within the function, are almost exactly reflected to the diversification time. For instance, for the TREC 2009 set, GLS calls the objective function 8,288 times, and in each call of the function, the loop iterates 1,600 times (cf. Algorithm 2); while for C-GLS (with k-means), these numbers are 1,703 and 400, respectively. Thus, in terms of the operation counts, C-GLS should be 19.5 times faster than GLS, which is actually reflected in the diversification times of 34.460ms and 741.108ms, respectively (i.e., implying a speedup of 21.5 times). A similar proportionality is also observed between C-GLS and C^2-GLS: both methods iterate the same number of times (namely, 400) in the objective function; however, the former calls it 1,703 times, whereas the latter calls only 23 times (again for the TREC 2009 dataset), a reduction of almost 74 times, which is almost perfectly reflected in the diversification times (i.e., 34.460ms. vs. 0.473ms, indicating a speed-up of 73.2 times).

Finally note that the distance lookups during the computation of the objective function take a nonnegligible portion of the online diversification time (i.e., up to 20% for different algorithms and datasets). Our methods again significantly reduce the time for the lookups, as shown in Table VI. While we cache all the distances during the preprocessing for all the methods compared here, for the scenarios where such a caching may not be possible and the lookups have to be replaced by the actual distance computations, our reductions for the online diversification time would be more emphasized. Overall, all these findings confirm that the reductions shown in the algorithmic complexities (cf. Table I) for the proposed methods are also reflected in the actual performance.

5.3. Evaluation of the Distributed Strategies

In this section, we investigate the impact of a distributed query processing architecture on the diversification effectiveness and efficiency of the implicit (i.e., GLS, C-GLS, and

Table VII. Retrieval Effectiveness of Distributed Diversification Algorithms for TREC 2009 and 2010 Topic Sets

			ERR-IA			α-NDCG			S-recall		
Topic Set	Algorithm	Dist. Strategy	@5	@10	@20	@5	@10	@20	@5	@10	@20
TREC'09	GLS	BB-Div	0.152	0.164	0.169	0.228	0.249	0.270	0.285	0.348	0.393
		NB-Div	0.124	0.138	0.147	0.187	0.219	0.254	0.257	0.351	0.423
	C-GLS	BB-Div	0.155	0.164	0.171	0.233	0.248	0.272	0.316	0.353	0.420
		NB-Div	0.142	0.159	0.167	0.214	0.248	0.281	0.280	0.361	0.459
	C^2-GLS	BB-Div	0.153	0.163	0.169	0.230	0.247	0.270	0.313	0.381	0.435
		NB-Div	0.133	0.147	0.156	0.195	0.227	0.265	0.258	0.345	0.421
	xQuAD	BB-Div	0.152‡	0.169‡	0.178‡	0.220‡	0.256‡	0.294‡	0.299‡	0.395‡	0.461
		NB-Div	0.136	0.149	0.160	0.202	0.229	0.275	0.267	0.347	0.465
TREC'10	GLS	BB-Div	0.138	0.156	0.166	0.184	0.221	0.255	0.276	0.386	0.483
		NB-Div	0.145	0.162	0.171	0.188	0.223	0.257	0.289	0.394	0.485
	C-GLS	BB-Div	0.168	0.187	0.195	0.209	0.249	0.278	0.287	0.408	0.507
		NB-Div	0.165	0.177	0.188	0.214	0.238	0.277	0.324	0.392	0.513
	C^2-GLS	BB-Div	0.164	0.175	0.184	0.210	0.231	0.265	0.310	0.385	0.506
		NB-Div	0.153	0.168	0.178	0.193	0.228	0.264	0.281	0.386	0.482
	xQuAD	BB-Div	0.177‡	0.194‡	0.200‡	0.234‡	0.272‡	0.293	0.390‡	0.487	0.526‡
		NB-Div	0.154	0.176	0.191	0.204	0.256	0.312	0.319	0.469	0.631

The cases where the result of the BB-Div strategy differs significantly (at 0.05 level) from that of the NB-Div strategy are denoted with ‡.

C^2-GLS with the k-means) and explicit (xQuAD) diversification approaches, in a setup that employs either broker-based (BB-Div) or node-based (NB-Div) diversification, as proposed in Section 4. To this end, we assume a simulated search cluster with $P = 10$ nodes. Given that the total collection is around 50 million documents, we believe the choice of cluster size is realistic, as each node is typically expected to include a few million documents (see, for instance Ozcan et al. [2012] and Cambazoglu et al. [2010] and the industrial practice[6]). In our simulation runs, we first retrieve the top-1,000 documents for a given query q and randomly distribute[7] these results to each node so that each node stores 100 documents (i.e., $N = 100$). We repeated each simulation run five times and report the average results. As in the previous section, k is set to 20.

For broker-based diversification, each node returns its local top-20 results based on the relevance scores, resulting in up to 200 documents. From these documents, the broker selects the most relevant 100 documents (as we keep $N = 100$ through all experiments for the sake of comparability) and executes the diversification algorithm on the top-100 set to create the final result set of 20 documents.

In the case of node-based diversification, each node applies the diversification algorithm to determine its local diversified result set of size 20 from its own 100 candidates. These diversified sets are merged at the broker and the global top-20 results are returned.

Effectiveness evaluation. In Table VII, we provide the evaluation results using TREC 2009 and 2010 topics for all four algorithms combined with each of the two distributed diversification strategies. We see that broker-based and node-based diversification strategies exhibit a similar effectiveness, but the former is slightly better for the majority of algorithms and evaluation metrics. The differences between two distributed strategies are more visible for GLS (on TREC 2009 topics) and for xQuAD (on both topic sets) and found to be statistically significant (at the 0.05 level using one-way ANOVA) for the latter algorithm.

[6]http://www.searchtechnologies.com/enterprise-search-scalability.html.

[7]We discuss alternative document partitioning strategies later in the subsection entitled *Critical Assumptions*.

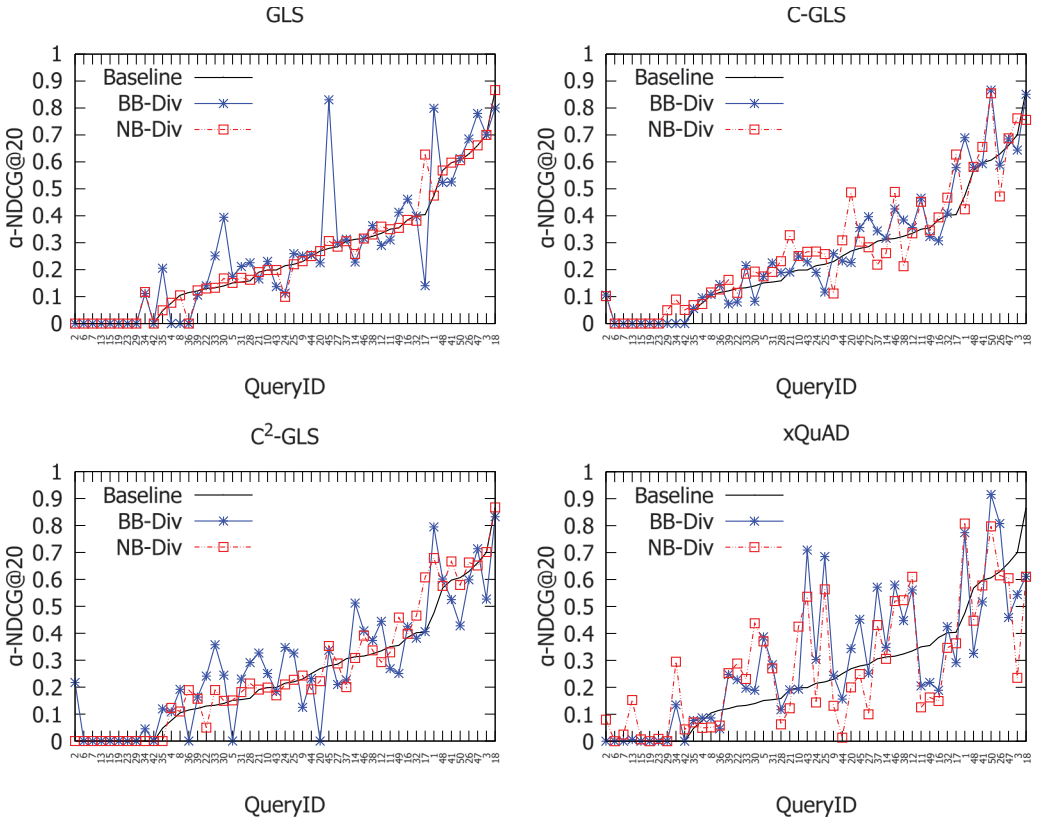
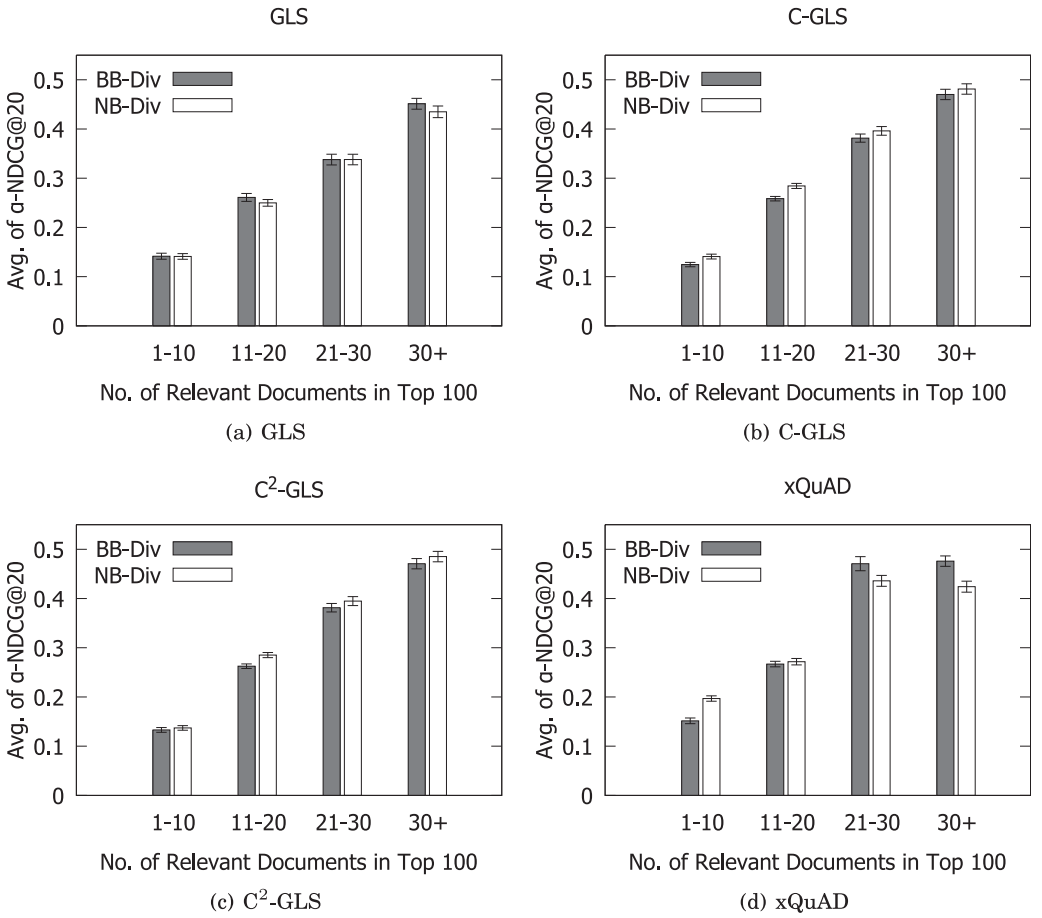


Fig. 3. Query-wise α-NDCG@20 scores for BB-Div and NB-Div using GLS, C-GLS, C^2-GLS, and xQuAD diversification algorithms (query ids sorted in ascending order of $\alpha-$NDCG@20 scores for the baseline).

In addition to average values, for each diversification strategy, we also provide a query-wise breakup of performances in Figure 3 for each of the algorithms, namely, GLS, C-GLS, C^2-GLS, and xQuAD, respectively. In the plots, queries are sorted in the order of increasing α-NDCG@20 score obtained for the nondiversified baseline. We see that, in line with the general trends, diversification algorithms are usually outperforming the baseline. Interestingly, regardless of the layer of diversification (i.e., either at the broker or nodes), both xQuAD and GLS exhibit a more volatile behavior in that they improve certain queries a lot but also hurt some others a lot. On the other hand, our algorithms (especially C-GLS) behave more conservatively, but while they usually improve the result diversity, the gains can be rather small for many queries. Nevertheless, this is a positive finding for our cluster-based approaches as their diversification performance seems to be more robust over a set of queries.

Next, we concentrate on the possible causes of the performance differences between the distributed diversification strategies, especially for the GLS and xQuAD algorithms. For both of the latter algorithms, BB-Div seems to outperform NB-Div for the queries with higher α-NDCG scores (see corresponding plots in Figure 3). For a better insight, in Figures 4(a) to 4(d), we show the effectiveness with respect to the number of relevant documents (based on the relevance judgments) encountered in the broker's candidate set D (i.e., top-100 results) for each query. The figure shows that, especially for xQuAD and GLS, the gap between BB-Div and NB-Div widens as the



(a) GLS (b) C-GLS

(c) C^2-GLS (d) xQuAD

Fig. 4. Effectiveness of distributed diversification strategies versus the number of relevant documents in the top-100 results.

number of relevant documents in the top-100 increases. This implies that, while the NB-Div strategy works on a deeper pool (as each node operates on a different set of candidate documents), its pool may not include as many relevant documents as that of the broker. Supporting this latter hypothesis, for each query, Figure 5 shows the percentage of relevant documents in the top-100 and top-1,000 results, that is, the broker's candidate set and the union of the nodes' candidate sets, respectively (as we distribute the top-1,000 results of a query among 10 nodes in the simulation runs). We see that while for some queries the percentage of relevant documents reaches up to 75% in the top-100, the percentage of relevant documents in the top-1,000 does not increase proportionally, and indeed, it remains mostly less than 10%. In other words, the biggest possible advantage of NB-Div, being able to consider a deeper pool of documents, does not necessarily help in this setup, as the majority of the documents in the candidate sets of the nodes are indeed irrelevant.

In this sense, the potential of the NB-Div strategy should not be underestimated: although it encounters many more irrelevant documents than the BB-Div strategy in the current TREC evaluation setting, it can still provide comparable results to the BB-Div strategy. We presume that in a setup where each query has more relevant

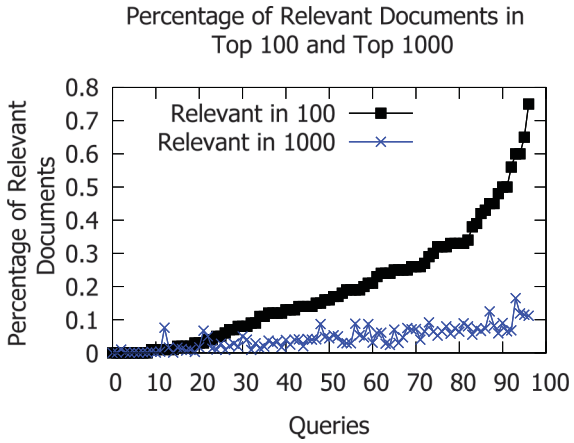


Fig. 5. Percentage of documents judged as relevant in the top-100 and top-1,000 results for TREC 2009 and 2010 queries. Note that the x-axis represents the queries sorted wrt the number of relevant documents in the top-100 for a more clear visualization.

documents and the initial retrieval strategy can retrieve more relevant documents in the top-1,000, NB-Div might outperform BB-Div. However, further investigation would require a test collection with a much higher number of relevance judgments. We identify the latter point as a limitation of the experimental framework provided by the TREC Diversification Task: the majority of the queries have a small number of relevant documents (i.e., on average, there are around 200 relevant documents per query in the TREC topic sets released between 2009 and 2012) and they are distributed quite unevenly among the subtopics.

Efficiency evaluation. In this section, we compare the efficiency of BB-Div and NB-Div in terms of the network communication costs. As a particular diversification algorithm is always executed on a candidate set of the same size either at the broker or at the nodes (and since the latter execution takes place in parallel), the processing cost of the algorithm itself (as extensively discussed in Section 5.2) would be the same for BB-Div and NB-Div. For this reason, our discussion in this section focuses only on the network communication costs. In our analysis, as in the previous section, we employ GLS (and its variants) and xQuAD as the representative methods for the implicit and explicit diversification approaches, respectively. However, the cost formulas developed here are applicable to any implicit diversification method that needs to access the actual document vectors of the candidate documents (e.g., our C-GLS and C^2-GLS, MMR [Carbonell and Goldstein 1998], etc.), and any explicit method that needs to compute the scores of the candidate documents for each (known) query aspect (e.g, PM2 [Dang and Croft 2012] and aggregation-based methods in Ozdemiray and Altingovde [2015]). In this sense, our analysis sheds light on the general efficiency figures of the typical implicit and explicit diversification approaches on a distributed setup. Therefore, in the following cost formulas and experimental results, we prefer to label the cases as "implicit" or "explicit" diversification, rather than using particular algorithm names.

For the more straightforward NB-Div strategy, there is no additional network communication, assuming that each node stores the document vectors for the set of documents that are assigned to it, as well as the index on these documents,[8] which is

[8]We discuss alternative storage strategies for document vectors later in the subsection entitled *Critical Assumptions*.

Table VIII. Parameters for the Network Cost Computations

Parameter	Description
$\|d\|$	No. of distinct terms in a document
s	Size of an entry in the document vector (in bytes)
$\|A\|$	No. of aspects for a query
$\|a\|$	Size of a query aspect (in bytes)
f	Size of an entry in the vector of aspect scores (in bytes)
O_j	Set of documents hosted at the node j
$\|V\|$	No. of nodes that include at least one candidate document
T	Transfer rate of the network (MB/s)

Table IX. Network Communication Costs for the Implicit and Explicit Diversification Approaches with BB-Div Strategy

Algorithm Type	Communication Volume	Communication Time
Implicit	$\sum_{d_i \in D} \|d_i\| \times s$	$\max_{j \in \{1,\ldots,P\}} \frac{\sum_{d_i \in D \cap O_j} \|d_i\| \times s}{T}$
Explicit	$\left(\sum_{a_i \in A} \|a_i\| \times \|V\|\right) + (N \times \|A\| \times \|f\|)$	$\max_{j \in \{1,\ldots,P\}} \frac{\sum_{a_i \in A} \|a_i\| + \sum_{d_i \in D \cap O_j} (\|A\| \times f)}{T}$

a practical assumption also employed in earlier works (e.g., Ozcan et al. [2012]). In contrast, the BB-Div strategy requires that the necessary information for the diversification stage, which naturally depends on the type of the utilized algorithm, should be transferred to the broker. In what follows, we analyze this latter case in detail.

In Table VIII, we list the parameters and their symbols to be used in the cost formulas, and in Table IX, we provide the formulas for the network cost of the implicit and explicit algorithms when employed together with the BB-Div strategy. Note that the communication volume is computed by summing the total amount of data (in bytes) that needs to be transferred on the network. In contrast, network communication time is computed as the time to transfer a data package to the broker from the node that sends the *maximum* amount of data (note that the number of candidate documents stored in a node and their total size may differ among the nodes).

We derive the formulas in Table IX based on the following facts. If BB-Div employs an implicit diversification algorithm, it will need to fetch the document vectors of the candidate results (so that the pairwise document similarities, as in GLS or MMR [Carbonell and Goldstein 1998], or document-cluster similarities, as in the case of our C-GLS and C^2-GLS methods, can be computed). Hence, the network communication volume is simply the sum of the lengths of the document vectors for all candidate documents in D. In contrast, for the communication time, we compute the transfer time from each node to the broker, which is based on the total document length of the candidate documents hosted at a node, and take the *maximum* of these transfer times, as the broker needs to wait until all data from the nodes arrive.

In case of an explicit algorithm, such as xQuAD, the broker should compute the score of each query aspect for each candidate document using a retrieval model. This latter score can be computed by either transferring the document vectors to the broker, of which cost was already discussed for the BB-Div case, or, more conveniently, sending the query aspects to those nodes that host the documents in the candidate set. In the communication volume formula for the explicit case in Table IX, the first summation

Table X. Network Cost in Terms of the Communication Volume (in Bytes) and Time (in Milliseconds) per Query for the BB-Div Strategy

Topic Set	Div. Type	Comm. Volume	Comm. Time
TREC'09	Implicit	391,360	5.159
TREC'10	Implicit	433,658	5.749
TREC'09	Explicit	5,455	0.066
TREC'10	Explicit	4,845	0.058

represents the size of the query aspect strings (in bytes), which is the amount of data that is sent to each node including a document in D. Then, each such node computes the aspect-document score by using its local inverted index and sends back to the broker a vector that involves the score of each candidate document for each aspect. The overall communication volume incurred by this latter stage is $N \times |A| \times f$, where A denotes the set of query aspects, f denotes the size of a score value, and N is the size of the candidate set, as before.

To apply the cost formulas in Table IX in our experimental setup, we set the parameter s as 8 bytes, assuming that each entry in the document vector will include two integer values, a term id and its frequency in the document. We also set the parameter f again as 8 bytes, assuming that the score of an aspect for a document is stored as a double value. The transfer rate of the network, assuming a LAN, is set to 11MB/s. Note that we discuss other possible values for these parameters in the following subsection.

Table X reveals that, as expected, explicit diversification algorithms incur network costs that are two orders of magnitude smaller than those incurred by the implicit algorithms; hence, if query aspects are available beforehand, employing an explicit algorithm on top of the BB-Div strategy is more efficient. In contrast, for the practical scenarios where no aspects are known and implicit methods need to applied, the BB-Div strategy incurs some overhead in terms of the communication volume and time. However, we envision that these additional costs are still affordable. For instance, in a real-life setting, a typical query of 15 characters on average (e.g., see Kamvar and Baluja [2006]) needs to be sent to several thousands of nodes, say, 50,000, at a particular data center (this is a moderate estimation given that Microsoft had more than 1 million servers in 2013[9]). With a back-of-the-envelope computation, we see that even forwarding a query string to the nodes in the latter setup causes a communication volume of 750,000 bytes, which is larger than the data volumes shown in Table X (e.g., 391,360 and 433,658 bytes per query for the TREC 2009 and 2010 sets, respectively). Furthermore, the communication volume formula in Table IX only involves N, the size of the candidate set, as a parameter, but not the number of nodes; hence, the overhead will be the same regardless of the number of nodes for a fixed N. Regarding the network communication time, the cost is around 5ms using a moderate parameter for the network transfer rate (i.e., 11MB/s) and should be clearly affordable in a real-life setup. Therefore, we conclude that, when the candidate set D is small, the network costs seem to be an affordable overhead, and BB-Div remains as a viable option for applying implicit diversification in a distributed setup.

Critical assumptions. In our evaluation of the diversification algorithms in a distributed setup, we have some critical assumptions that are essentially based on the common practice for web search setup and may not hold in different scenarios. We list and discuss these assumptions as follows:

—Distribution of the collection: As discussed in the related work section, it is usually assumed that the state-of-the-art method used in partitioning the collection of

[9]http://www.datacenterknowledge.com/archives/2013/07/15/ballmer-microsoft-has-1-million-servers/.

a search engine is document partitioning, where each node (and its possible replicas) in the system is responsible for a disjoint subset of the collection and the corresponding index. In the literature, different approaches for assigning documents to the nodes are proposed. The most straightforward approach, as we also assume here, is a random allocation. In contrast, alternative partitioning approaches usually aim to store the similar documents at the same node, which can be achieved via unsupervised clustering, using semantic catalogues, or exploiting previous query results in the log (e.g., Puppin et al. [2006] and Cambazoglu and Baeza-Yates [2011]). Despite its simplicity, the random document partitioning is attractive in many ways: First, its implementation is practical, as a hash function can be used to quickly assign each document to a node. In contrast, alternative approaches require running an algorithm to determine the document's node. Second, random partitioning achieves good load balancing. In contrast, alternative document partitioning models usually suffer from load imbalance; that is, some nodes need to answer a large number of queries while some others stay idle. Finally, in the random partitioning, fault tolerance may be simply achieved by having a fixed number of replicas for each node. For the alternative approaches, this issue is also more complicated due to the load imbalance problem; that is, the nodes that include the documents from the most popular sites may be accessed more and need to have a larger number of replicas. In the light of these discussions, we believe that the random document partitioning is the most practical approach and hence employed in large-scale search engines, as also stated in Feldman et al. [2011]. Nevertheless, for the scenarios where topically similar documents are assigned together to a single node (e.g., a metasearch scenario as in Kulkarni and Callan [2010]), the query processing model would also change (i.e., include a resource selection stage instead of forwarding the query to all the nodes), and hence, our findings discussed in the previous sections may not hold. Clearly, such scenarios are not in the scope of our article and can be investigated in the future work.

—Index and document servers: In this work, following the practice in the earlier studies [Ozcan et al. 2012], we assume that a particular node stores both a disjoint subset of the collection and its index. It is also possible that the document subset and its corresponding index are stored in physically different servers, that is, at a document and index server, respectively (e.g., see Dean [2009]). In this case, implicit diversification methods with the NB-Div strategy would need to fetch the document vectors from another server, yielding network costs similar to those in the BB-Div case. For the explicit diversification methods with the NB-Div strategy, there would be still no network costs, as these approaches use the index to compute the score of each candidate document for each query aspect.

—System parameters: In our cost formulas shown in Table IX, we set the size of a document vector entry as 8 bytes, assuming that such a vector is composed of integer pairs that represent a term identifier and its frequency in the document. In practice, such a vector can have additional information and/or can be stored in a more efficient way, that is, in a compressed form. We also assume a network speed of 11MB/s, which might be very moderate for connecting the servers in a data center. Nevertheless, we believe that replacing such parameters with more realistic values will not change the trends reported in this article.

6. SUMMARY OF THE KEY FINDINGS

Our key findings in this article can be summarized as follows:

—Explicit diversification approaches are both effective and efficient, as xQuAD, a representative explicit approach, spends around only 10ms for diversification, whereas

the implicit algorithms are either efficient yet less effective (as in the case of LCD) or effective yet inefficient (as in the case of the original GLS). Given that the explicit aspects of a query may not be always available in practical scenarios, this finding also justifies our first goal in this article, namely, improving the efficiency of GLS as a promising implicit algorithm.

—Using clustering as a basis of the diversification on its own does not yield high-quality results, as LCD is found to be only slightly more effective than the nondiversified baseline. In contrast, our methods C-GLS and C^2-GLS that employ the clustering as a preprocessing stage for GLS are found to be both effective and efficient. In particular, when the k-means algorithm is used for the preprocessing, their effectiveness is comparable to (or sometimes better than) GLS, whereas the overall diversification time is reduced by more than 80%. It is also possible to improve the diversification efficiency by employing a cheaper preprocessing algorithm, namely, LC, that yields slightly inferior diversification quality in return to higher efficiency. For this latter case, C^2-GLS takes at most 23ms, which means a 97% improvement over GLS. These findings mean that the proposed algorithms can be utilized in real-world scenarios with strict budgets for query processing.

—Our experiments on a distributed setup show that running the diversification algorithms (of either implicit or explicit type) at the broker (i.e., using BB-Div) yields higher effectiveness scores than applying diversification at each node (i.e., using NB-Div). Our detailed analysis reveals that the ineffectiveness of NB-Div might be caused by the relatively small number of relevant documents per query in the TREC datasets. Because of this, the candidate sets at the nodes include a larger number of irrelevant documents and, hence, lead to inferior diversification effectiveness.

—We also show that NB-Div is relatively cheap, as it incurs no network communication overhead in a typical distributed setup. In contrast, BB-Div has additional overhead in terms of the network costs (especially for the implicit diversification algorithms); however, these costs, namely, network communication volume and time, seem to be affordable in a practical web search setup.

In light of these findings, we can claim that in a setup that needs an implicit diversification algorithm, the proposed methods C-GLS and C^2-GLS (with the k-means or LC preprocessing) can be safely utilized as effective and efficient variants of GLS. Furthermore, if a given query is expected to return a relatively small number of relevant documents, it may be better to apply these algorithms at the broker (as the network communication overhead seems to be affordable in a realistic setup); otherwise, applying the diversification at the nodes would be a more efficient choice.

7. CONCLUSION

For practical application of diversification in a large-scale setting, two requirements need to be met. First, we need an algorithm with low computational complexity to satisfy the demanding efficiency requirements of online query processing. Second, the diversification process should be executable on a computing cluster where each node holds a collection partition, because larger collections cannot be maintained on one central node.

In this work, we presented C-GLS and C^2-GLS, two greedy algorithms that perform an initial document clustering to reduce the GLS complexity from quadratic to linear (with the number of candidate documents). We show that the proposed approaches can reduce the online diversification cost by more than 80% and up to 97% while achieving comparable or even better effectiveness than the GLS solution.

We also studied how distribution of the diversification process affects its result quality and efficiency. In our experiments, diversification on the broker with the BB-Div

strategy yielded better result quality than diversification on the nodes with the NB-Div strategy; however, there were also cases where both strategies performed equally well. While evaluating their efficiency, we found that the diversification algorithms with the BB-Div strategy incur additional costs for the network communication (while NB-Div incurs no network costs); fortunately, this seems to be an affordable overhead in real-life settings.

These two contributions pave the way for scalable distributed diversification of search results for web-scale document collections. We also anticipate that our work may lead the community interest toward the development and evaluation of diversification algorithms on distributed architectures, which we believe to be the next and natural testbed for evolving research in this field.

In our future work, we plan to evaluate the distributed diversification for other scenarios that employ alternative document allocation policies. We also plan to investigate approaches to further reduce the network communication costs when diversification is applied at the broker.

REFERENCES

Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *Proceedings of the 2nd International Conference on Web Search and Web Data Mining*. 5–14.

Berkant Barla Cambazoglu and Ricardo Baeza-Yates. 2011. Scalability challenges in web search engines. In *Advanced Topics in Information Retrieval*, Massimo Melucci and Ricardo Baeza-Yates (Eds.). The Information Retrieval Series, Vol. 33. 27–50.

Berkant Barla Cambazoglu, Emre Varol, Enver Kayaaslan, Cevdet Aykanat, and Ricardo A. Baeza-Yates. 2010. Query forwarding in geographically distributed search engines. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 90–97.

Gabriele Capannini, Franco Maria Nardini, Raffaele Perego, and Fabrizio Silvestri. 2011. Efficient diversification of web search results. *PVLDB* 4, 7 (2011), 451–459.

Jaime G. Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 335–336.

Claudio Carpineto, Massimiliano D'Amico, and Giovanni Romano. 2012. Evaluating subtopic retrieval methods: Clustering versus diversification of search results. *Inf. Process. Manage.* 48, 2 (2012), 358–373.

Ben Carterette. 2009. An analysis of NP-completeness in novelty and diversity ranking. In *Proceedings of the 2nd International Conference on the Theory of Information Retrieval*. 200–211.

Ben Carterette and Praveen Chandar. 2009. Probabilistic models of ranking novel documents for faceted topic retrieval. In *Proceedings of the 18th ACM Conf. on Information and Knowledge Management*. 1287–1296.

Olivier Chapelle, Shihao Ji, Ciya Liao, Emre Velipasaoglu, Larry Lai, and Su-Lin Wu. 2011. Intent-based diversification of web search Results: Metrics and algorithms. *Inf. Retr.* 14, 6 (2011), 572–592.

Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. 621–630.

Edgar Chávez and Gonzalo Navarro. 2005. A compact space decomposition for effective metric indexing. *Pattern Recog. Lett.* 26, 9 (2005), 1363–1376. DOI:http://dx.doi.org/10.1016/j.patrec.2004.11.014

Harr Chen and David R. Karger. 2006. Less is more: Probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 429–436.

Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. 2009. Overview of the TREC 2009 web track. In *Proceedings of the 18th Text Retrieval Conference*.

Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Gordon V. Cormack. 2010. Overview of the TREC 2010 web track. In *Proceedings of the 19th Text Retrieval Conference*.

Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 659–666.

Van Dang and W. Bruce Croft. 2012. Diversity by proportionality: An election-based approach to search result diversification. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 65–74.

Van Dang and W. Bruce Croft. 2013. Term level search result diversification. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 603–612.

Jeffrey Dean. 2009. Challenges in building large-scale information retrieval systems: Invited talk. In *Proceedings of the 2nd International Conference on Web Search and Web Data Mining*. 1.

Marina Drosou and Evaggelia Pitoura. 2009. Diversity over continuous data. *IEEE Data Eng. Bull.* 32, 4 (2009), 49–56.

Moran Feldman, Ronny Lempel, Oren Somekh, and Kolman Vornovitsky. 2011. On the impact of random index-partitioning on index compression. *CoRR* abs/1107.5661 (2011). http://arxiv.org/abs/1107.5661.

Veronica Gil-Costa, Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2011. Sparse spatial selection for novelty-based search result diversification. In *Proceedings of the 18th International Symposium on String Processing and Information Retrieval*. 344–355.

Veronica Gil-Costa, Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2013. Modelling efficient novelty-based search result diversification in metric spaces. *J. Discrete Algorithms* 18 (2013), 75–88.

Sreenivas Gollapudi and Aneesh Sharma. 2009. An axiomatic approach for result diversification. In *Proceedings of the 18th International Conference on the World Wide Web*. 381–390.

Jiyin He, Edgar Meij, and Maarten de Rijke. 2011. Result diversification based on query-specific cluster ranking. *JASIST* 62, 3 (2011), 550–571.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

Maryam Kamvar and Shumeet Baluja. 2006. A large scale study of wireless search behavior: Google mobile search. In *Proceedings of the 2006 Conference on Human Factors in Computing Systems (CHI'06)*. 701–709.

Anagha Kulkarni and Jamie Callan. 2010. Document allocation policies for selective searching of distributed indexes. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*. 449–458.

Shangsong Liang, Zhaochun Ren, and Maarten de Rijke. 2014. Fusion helps diversification. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 303–312.

Harry Markowitz. 1952. Portfolio selection. *J. Finance* 7, 1 (1952), 77–91.

Enrico Minack, Wolf Siberski, and Wolfgang Nejdl. 2011. Incremental diversification for very large sets: A streaming-based approach. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 585–594.

Rifat Ozcan, Ismail Sengör Altingövde, Berkant Barla Cambazoglu, Flavio Paiva Junqueira, and Özgür Ulusoy. 2012. A five-level static cache architecture for web search engines. *Inf. Process. Manage.* 48, 5 (2012), 828–840.

Ahmet Murat Ozdemiray and Ismail Sengor Altingovde. 2014. Query performance prediction for aspect weighting in search result diversification. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 1871–1874.

Ahmet Murat Ozdemiray and Ismail Sengor Altingovde. 2015. Explicit search result diversification using score and rank aggregation methods. *JASIST* 66, 6 (2015), 1212–1228. DOI:http://dx.doi.org/10.1002/asi.23259

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems*.

Diego Puppin, Fabrizio Silvestri, and Domenico Laforenza. 2006. Query-driven document partitioning and collection selection. In *Proceedings of the 1st International Conference on Scalable Information Systems*. 34.

Filip Radlinski and Susan T. Dumais. 2006. Improving personalized web search using result diversification. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 691–692.

Stephen E. Robertson. 1977. The probability ranking principle in IR. *J. Document.* 33 (1977), 294–304.

Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2010a. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th International Conference on World Wide Web*. 881–890.

Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2010b. Selectively diversifying web search results. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*. 1179–1188.

Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. 2011. Intent-aware search result diversification. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 595–604.

David Vallet and Pablo Castells. 2012. Personalized diversification of search results. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 841–850.

Saul Vargas, Pablo Castells, and David Vallet. 2012. Explicit relevance models in intent-oriented information retrieval diversification. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 75–84.

Jun Wang and Jianhan Zhu. 2009. Portfolio theory of information retrieval. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 115–122.

Shengli Wu and Chunlan Huang. 2014. Search result diversification via data fusion. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 827–830.

ChengXiang Zhai, William W. Cohen, and John D. Lafferty. 2003. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 10–17.

ChengXiang Zhai and John D. Lafferty. 2006. A risk minimization framework for information retrieval. *Inf. Process. Manage.* 42, 1 (2006), 31–55.

Guido Zuccon and Leif Azzopardi. 2010. Using the quantum probability ranking principle to rank interdependent documents. In *Proceedings of the 32nd European Conf. on IR Research*. 357–369.

Guido Zuccon, Leif Azzopardi, Dell Zhang, and Jun Wang. 2012. Top-k retrieval using facility location analysis. In *Proceedings of the 34th European Conference on IR Research*. 305–316.

Received December 2014; revised March 2016; accepted March 2016