Improving the Efficiency of Multi-site Web Search Engines

Guillem Francès[*]
Universitat Pompeu Fabra
Barcelona, Spain
guillem.frances@upf.edu

Xiao Bai
Yahoo Labs
Barcelona, Spain
xbai@yahoo-inc.com

B. Barla Cambazoglu
Yahoo Labs
Barcelona, Spain
barla@yahoo-inc.com

Ricardo Baeza-Yates
Yahoo Labs
Barcelona, Spain
rbaeza@acm.org

ABSTRACT

A multi-site web search engine is composed of a number of search sites geographically distributed around the world. Each search site is typically responsible for crawling and indexing the web pages that are in its geographical neighborhood. A query is selectively processed on a subset of search sites that are predicted to return the best-matching results. The scalability and efficiency of multi-site web search engines have attracted a lot of research attention in recent years. In particular, research has focused on replicating important web pages across sites, forwarding queries to relevant sites, and caching results of previous queries. Yet, these problems have only been studied in isolation, but no prior work has properly investigated the interplay between them.

In this paper, we take this challenge up and conduct what we believe is the first comprehensive analysis of a full stack of techniques for efficient multi-site web search. Specifically, we propose a document replication technique that improves the query locality of the state-of-the-art approaches with various replication budget distribution strategies. We devise a machine learning approach to decide the query forwarding patterns, achieving a significantly lower false positive ratio than a state-of-the-art thresholding approach with little impact on search result quality. We propose three result caching strategies that reduce the number of forwarded queries and analyze the trade-off they introduce in terms of storage and network overheads. Finally, we show that the combination of the best-of-the-class techniques yields very promising search efficiency, rendering multi-site, geographically distributed web search engines an attractive alternative to centralized web search engines.

[*]This research was carried out during the author's internship at Yahoo Labs Barcelona.

WSDM'14, February 24–28, 2014, New York, New York, USA.
Copyright 2014 ACM 978-1-4503-2351-2/14/02 ...$15.00.
http://dx.doi.org/10.1145/2556195.2556249.

Categories and Subject Descriptors

H.3.3 [**Information Storage Systems**]: Information Retrieval Systems

Keywords

Distributed web search; query processing; document replication; query forwarding; result caching; efficiency

1. INTRODUCTION

As the vast amount of data available in the Web and the number of users accessing it keep their steady growth, an emerging line of research has focused on the scalability problems faced by single-site, centralized web search engines [3, 10, 13, 21]. An interesting alternative, multi-site distributed web search engines were shown to reduce the resource consumption in query processing and thus the financial costs of search engines, while also decreasing query response times experienced by users [3]. In multi-site web search engines, the whole document collection is partitioned into a number of distributed indexes, usually based on the geographical proximity between web servers and users. The objective is to exploit the so-called query locality, i.e., the fact that a large fraction of queries can find their top k best-matching documents in the index of the local site to which they are issued, saving the need to process them on non-local sites and thus reducing query response latencies and query processing workloads.

Multi-site web search engines raise three major research problems regarding query processing, which forms the focus of our work. First, since each search site indexes only a subset of the documents, for some queries, some of the best-matching documents may not be present in the local index. Therefore, a query forwarding strategy is necessary to determine, every time a query is received at a local site, which non-local search sites may have indexed the missing best-matching documents for the query, so that the query can be forwarded to those sites. In order to avoid unnecessary increases in query response time, it is important to forward queries only to those sites that indexed the best-matching documents. Second, forwarding queries increases the query response time due to the network latency among sites. If we replicate and locally index those documents that are frequently fetched from remote sites, more queries will benefit from query locality, decreasing query response times.

The challenge here is to determine the right documents to replicate on the sites in order to achieve high query locality with low storage overhead. Third, as commonly used in centralized search engines, a result cache is useful for reducing query response times and query processing workload. In addition to the problems faced by result caches in centralized search engines (e.g., admission, eviction, invalidation), it is also important for a multi-site search engine to determine in which sites the results of a query should be cached.

Contribution. We propose and evaluate new strategies for each of the outlined problems (document replication, query forwarding, and multi-site result caching) and conduct what we believe is the first realistic simulation of the full stack of strategies in a multi-site search engine:

- We propose a document replication technique that relies on per-document utility to select the documents to be replicated on each site, improving the query locality of the state-of-the-art approaches by up to 5.88% with various replication budget distribution strategies.
- We propose a machine learning approach for the query forwarding problem that exploits the trade-off between result quality, on the one hand, and query response time and system workload, on the other hand. Without using any document replication or result caching techniques, our technique is able to increase the fraction of locally processed queries by up to 61% with respect to a state-of-the-art approach.
- We propose three multi-site result caching strategies that improve the query locality of the local cache approach by up to 3.4% and evaluate the trade-off in terms of network and storage overheads.
- We conduct extensive experiments with real datasets to study the interplay among the proposed techniques and show that the combination of the best-of-the-class techniques improves the average query response time of the state-of-the-art approach by up to 24% with very little sacrifice from the result quality.

Paper outline. In Section 2, we provide some background on multi-site web search engines. Section 3 describes the data and setup used in our simulations. Sections 4, 5, and 6 present the strategies we propose for document replication, query forwarding, and multi-site result caching, respectively. Section 7 evaluates the combination of the best-of-the-class techniques. Section 8 surveys the related work. The paper is concluded in Section 9.

2. MULTI-SITE WEB SEARCH

2.1 Preliminaries

Notation. A distributed web search engine is composed of a set $\mathcal{S}=\{s_1, s_2, \ldots, s_m\}$ of m geographically distributed sites. A set $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ of n documents is partitioned among these m sites. Each site s_i maintains a local index built on a disjoint subset $\mathcal{D}_i$ of documents such that $\mathcal{D} = \bigcup_{1\leq i\leq m} \mathcal{D}_i$ and $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$, for $1 \leq j \leq m$ and $i \neq j$.[1] We denote by $size(d)$ the number of postings contributed to the inverted index by document d.[2] Also, for document set $\mathcal{D}$, we define $size(\mathcal{D}) = \sum_{d\in\mathcal{D}} size(d)$. Each site s_i serves a set Q_i of queries that are issued to the site. We denote by $f_i(q)$ the number of times a query q was issued to s_i (in a given query set). Finally, we denote by $R(q) \subseteq \mathcal{D}$ the global top-k result set for query q (i.e., the result set that would be returned by a centralized search engine).[3]

Query forwarding. Upon the reception of a query q at some search site s_i, a query forwarder decides in which search sites the best-matching documents of the query might have been indexed. Based on this decision, the query is forwarded to a number of sites and processed on their indexes. The top-k result sets retrieved from the contacted sites are then merged at the local site s_i and returned to the user.[4]

Replication. The occurrence of documents in query result sets is highly skewed. Therefore, replicating a certain subset of documents (typically, the most popular documents) on search sites considerably reduces the number of forwarded queries [3, 21]. We assume that each site s_i maintains a replicated index built on a subset $\mathcal{R}_i \subseteq \mathcal{D} \setminus \mathcal{D}_i$ of documents that are not present in the local index of the site.

Caching. Finally, some form of result caching might be used in order to reduce query response times as well as the overall system workload.

2.2 Performance Metrics

Query locality. For a given combination of document replication, query forwarding, and result caching strategies, we define query locality as the fraction of queries that are answered by the local site they are submitted to without necessitating any forwarding. Increasing the query locality allows lower query response times and also helps decreasing the overall system workload.

Query forwarding accuracy. Given a query q received at site s_i, the query forwarder on that site needs to decide whether to process q on the index of each site s_j ($1 \leq j \leq m$). Each of these m binary decisions might turn out to be a true positive (TP), a true negative (TN), a false positive (FP), or a false negative (FN). If the decision is not to forward the query to s_j, we have an FN if s_j has indexed some best-matching document in $R(q)$ (and that document is not locally replicated), or a TN otherwise. If the decision is to forward the query to s_j, we have an FP if s_j has indexed no best-matching documents, or a TP otherwise. An FN degrades the quality of search results since the final result set will be missing some best-matching documents. On the other hand, it might reduce the average response time and system workload. An FP might increase the average response time (because the user might have to wait for unnecessary query processing and round-trip time due to forwarding) as well as increasing the system workload. However, it does not affect the quality of results. We report the FP and FN rates as well as the overall accuracy for different query forwarding strategies.

Result quality. We aim at generating the same result set that a centralized search engine can provide. In order to measure the degradation in the result quality (caused by the FN rate of the query forwarder), we use a number of standard metrics. These metrics compare the result sets returned by the distributed architecture against those that would be returned by a centralized architecture, using the latter as a ground truth. We measure the quality at different result ranks, considering only the first p results (for practical

[1]We assume that the partitioning of documents on sites is determined according to some external design factor.

[2]This is equal to the number of unique terms in the document.

[3]In the rest of the paper, we assume a standard value of $k=10$.

[4]In contrast to previous approaches, we allow the query forwarder not to process q on the local site s_i, as described later.

reasons, we restrict ourselves to $p \in \{1, 5, 10\}$). The three metrics we used are summarized below:

(i) *Exact match rate at rank p (EMR@p).* We define the exact match rate as the fraction of queries for which the top-k results returned by the distributed architecture are identical to the top-k results returned by the centralized architecture.

(ii) *Average overlap at rank p (overlap@p).* For a given query q, let $D_p(q)$ and $R_p(q)$ be the sets containing the first p results returned by the distributed and centralized architectures, respectively. We define the overlap at rank p for query q as the fraction of results in $R_p(q)$ that are also present in $D_p(q)$, i.e., $|D_p(q) \cap R_p(q)|/|R_p(q)|$.[5] The obtained overlap values are averaged over all queries in a test set.

(iii) *Average normalized discounted cumulative gain at rank p (NDCG@p).* Finally, we measure the average NDCG metric. Here, we assume that a document is relevant if and only if it belongs to the top-k result set of the centralized architecture. We normalize the DCG of a query by the DCG value attained by the centralized architecture.[6]

Query response time. We estimate query response times by simulating the processing of queries in a distributed web search engine and report the distribution of response times.

System workload. We estimate the workload incurred by a query as the sum of the lengths of the inverted lists that need to be traversed when processing the query. We average the workload estimates over all queries in a query set.

3. EXPERIMENTAL SETUP

Datasets. We simulate a distributed search engine composed of five sites, located in five different major continents. For each site, we sample consecutive queries from the corresponding search frontend of a commercial web search engine. Queries are normalized by case-folding, stop-word removal, term uniquing, and alphabetical ordering of query terms. The full query set is split roughly 75%–25% between a training set (5.25 million queries), which we use to compute the replication utility of the documents and to train our query forwarding models, and a test set (1.72 million queries), which we use to compute the performance metrics. The document collection contains about 200 million web pages obtained after various cleansing and filtering steps such as spam filtering. A proprietary classifier is used to assign the documents to a home country (some documents are not assigned to any of the five sites). This classifier uses features such as document language, IP address, and domain name, among others.

Simulation parameters. For each search site, we build a separate index using the documents assigned to it. Each site is assumed to have computational resources proportional to its index size. The simulated response times include query processing times and network latencies. For the former, we assume a processing cost of 200 ns per posting and a fixed preprocessing overhead of 20 ms per query, as in [13]. The top k query results are retrieved using the open-source search engine Terrier. For the latter, we considered the user-to-site and site-to-site network latencies, estimated based on the geodesic distances between the users and sites.

Baselines. We compare the performance of our query forwarder to the state-of-the-art `LP` forwarding model proposed in [13]. In order to facilitate the interpretation of our results, we often show the performance of an `ORACLE` query forwarder that always knows, for any given query, in which search sites the top k best-matching documents are indexed.

4. DOCUMENT REPLICATION

As mentioned before, document replication helps increasing the query locality in a distributed search engine. Replicating too many documents, however, defeats the purpose of a distributed search engine. Moreover, query processing time may increase when the replicated indexes are too large. Typically, the storage overhead of replicated documents are constrained by an upper-bound on the replication amount. Herein, we aim to devise a replication strategy that selects a set $\mathcal{R}_i \subset D \setminus D_i$ of documents and builds a replicated index on those documents for each site s_i such that

(i) the total size of the replicated documents remains below a given fraction b of the global document collection's size (i.e., $\sum_{1\leq i\leq m} size(\mathcal{R}_i) \leq b \times size(\mathcal{D})$), and

(ii) the query locality, i.e., the fraction of queries that can be processed locally without any quality loss (on both local and replicated indexes), is maximized.

In the rest of the section, we first categorize different document replication strategies (Section 4.1). We then propose a new document selection heuristic (Section 4.2). Finally, we show that this heuristic outperforms existing heuristics in terms of query locality (Section 4.3), assuming the presence of an `ORACLE` query forwarder.

4.1 Document Replication Strategies

Figure 1 illustrates the way different strategies replicate documents across the search sites. Figure 1(a) illustrates the case without any document replication, with each site indexing its own disjoint subset of documents.

Identical replication. A simple strategy is to replicate the same set $\mathcal{R}$ of documents on all sites (Figure 1(b)). Since part of the replicated documents may already exist in the local collection $\mathcal{D}_i$ of site s_i (the dotted regions within the replicated indexes in Figure 1(b)), only those documents in $\mathcal{R}_i = \mathcal{R} \setminus \mathcal{D}_i$ are used to build the replicated index. At query processing time, this avoids the potential redundancy due to duplicate scoring of documents that appear in both indexes. To bound the total size of replicated documents, we just need to ensure that $size(\mathcal{R}) \leq \frac{b}{m} \times size(\mathcal{D})$.

Individual replication. Identical replication possesses an obvious drawback: the documents best-suited for replication on site s_i might not be the best documents for site s_j since the users of each search site might have different information needs. A possible improvement is to replicate a distinct subset of documents on each site such that the replicated documents are selected depending on the query stream of the site. We study the following two alternatives for this kind of individual replication strategies.

(i) *Global replication budget.* In this option, the total size of the replicated indexes is constrained, i.e., $\sum_{1\leq i\leq m} size(\mathcal{R}_i) \leq b \times size(\mathcal{D})$. As shown in Figure 1(c), a site with a small local index may have a large replicated index (e.g., site 3) and a site with a

[5] We note that this metric becomes identical to the recall metric if the results in $R_p(q)$ are the only results considered as relevant.

[6] In this respect, our metric is slightly different than the standard NDCG definition in [19].

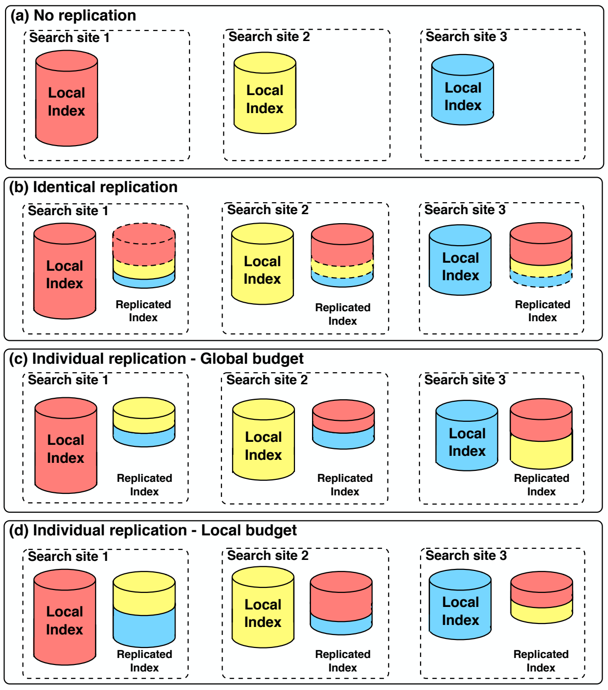


Figure 1: Different document replication strategies.

large local index may have a small replicated index (e.g., site 1).

(ii) *Local replication budget.* In a distributed search engine, the amount of storage and computational resources available in search sites may show high variation. Under a global replication budget, replicating a large number of documents on a small search site with limited resources may require redistribution of resources among the sites or an investment in new resources. A potential solution is to use a local replication budget and limit the size of the replicated index on each site s_i to a certain fraction b of the local index size, i.e., $size(\mathcal{R}_i) \leq b \times size(\mathcal{D}_i)$ (Figure 1(d)).

4.2 Document Selection Heuristics

Herein, we present a generic heuristic that selects the documents to be replicated at each site, for any given replication strategy. For a given query set Q, it has been shown that the computation of the replicated subsets $\mathcal{R}_i$ that maximize the fraction of top-k relevant documents that are indexed locally can be reduced to the 0-1 knapsack problem [14, 21],[7] which is known to be NP-hard. In our work, we rely on a greedy heuristic where the documents are sorted in descending order of their replication utility and progressively selected in this order until the replication budget is exhausted. We next define how the replication utility of a document is estimated.

For a given site s_i, let $\overline{R}_i(q) = R(q) \setminus \mathcal{D}_i$ be the set of documents in the top-k result set of q that are not indexed in the local index of s_i. We define the utility of replicating a document $d \in \mathcal{D} \setminus \mathcal{D}_i$ on site s_i with respect to query q as

$$u_{i,q}(d) = \begin{cases} \frac{1}{|\overline{R}_i(q)| \times size(d)}, & \text{if } d \in R(q); \\ 0, & \text{otherwise.} \end{cases}$$

[7]The basic idea is to assign each document a weight equal to its size and a utility proportional to the number of top-k result sets where it appears.

Given this, we define the global utility of replicating a document $d \in \mathcal{D} \setminus \mathcal{D}_i$ on site s_i as the sum of the individual utility of each query q weighted by its frequency $f_i(q)$, i.e.,

$$u_i(d) = \sum_{q \in Q_i} f_i(q) \times u_{i,q}(d).$$

The choice of $u_i(d)$ stems from the following observations: (i) replicating a non-local document in $R(q)$ is more likely to increase the query locality of site s_i if query q is frequently issued to site s_i (i.e., larger $f_i(q)$), (ii) as there are fewer non-local documents in $R(q)$, it becomes more likely that replicating one of these non-local documents will let q be processed locally, and (iii) the utility should be computed per unit document size, not per document. With this utility definition, we build the replicated document sets $\mathcal{R}_i$ for each replication strategy discussed in Section 4.1 as follows:

Identical replication. We compute the utility $u(d)$ of replicating a document as the sum of the utilities of replicating it on each site, i.e., $u(d) = \sum_{1 \leq i \leq m} u_i(d)$ and select documents in descending order of $u(d)$ values until the replication budget $\frac{b}{m} \times size(\mathcal{D})$ is exhausted.

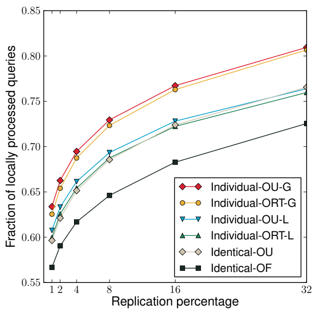
Individual replication with global budget. Since a site can replicate any number of documents as long as the global replication budget is not exhausted, we rank the pairs $\langle u_i(d), s_i \rangle$ in descending order of $u_i(d)$ and select document d with the highest $u_i(d)$ for replication on site s_i until the global replication budget $b \times size(\mathcal{D})$ is exhausted.

Individual replication with local budget. The size of replicated documents in a site should not exceed the local budget of the site. For each site s_i, we rank the documents in descending order of their $u_i(d)$ values and select them in this order until the local budget $b \times size(\mathcal{D}_i)$ is exhausted.

4.3 Experiments on Performance

In this section, we compare the performance of the proposed heuristic against the frequency-based heuristic used for identical replication in [13], where the utility of replicating a document is computed as the total frequency of the queries having the document in their top-k results divided by the size of the document. We refer to this heuristic as `Identical-OF`, where `OF` stands for optimizing frequency. Moreover, we compare our heuristic against the `ORT` (optimizing response time) heuristic used in [21] for individual replication both with global (`Individual-ORT-G`) and local (`Individual-ORT-L`) replication budgets. In [21], the utility of replicating a document, is computed as in [13], but only non-local documents are considered for replication. We refer to the three replication strategies using our heuristics as `Identical-OU`, `Individual-OU-G`, and `Individual-OU-L`, where `OU` stands for optimizing utility.

Query locality. If no documents are replicated, 44.40% of queries can be processed entirely locally (44.80% in case of unique queries). Figure 2 compares our heuristic against the three baseline heuristics under different replication strategies for varying replication budgets. We observe that (i) replicating documents significantly improves the number of locally processed queries (e.g., replicating only 1% of documents improves the query locality by 36% assuming all queries are used), (ii) individual replication outperforms identical replication, (iii) for individual replication, using global budgets outperforms using local budgets, and (iv) our heuristic (`*-OU-*`) improves the query locality metric by 0.34% to 1.32% in case of all queries and by 0.73% to 2.22%


(a) All queries.

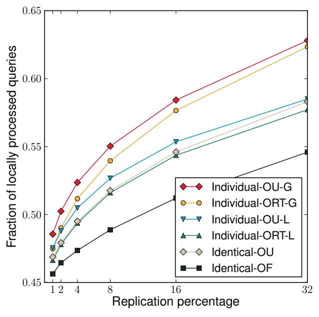
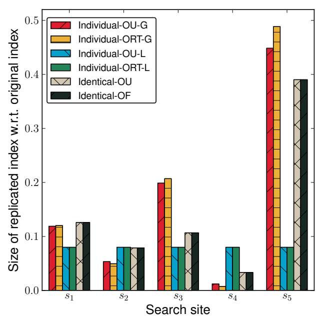

(b) Unique queries.

Figure 2: Effectiveness of document replication.


(a) Storage overhead.

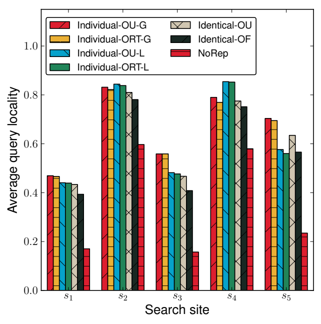

(b) Locally processed queries.

Figure 3: Load distribution among sites ($b = 8\%$).

in case of unique queries relative to the two individual replication strategies. The improvement with respect to identical replication is up to 5.88%. Interestingly, the improvement is higher in case of unique queries. This will be important once we add a cache layer (Section 6).

Load distribution. We study the load distribution among different sites by measuring the per-site storage overhead due to replication. This overhead is measured as the ratio between the replicated and local index sizes. According to Figure 3(a), individual replication with a global budget replicates more documents in sites having smaller local indexes (e.g., s_5) while individual replication with local budgets results in a more balanced replication pattern.[8] In Figure 3(b), we observe that sites with larger local indexes have higher query locality. This explains why individual replication with a global budget results in larger query locality than individual replication with local budgets.

5. QUERY FORWARDING

In this section, we describe the solution we propose for the query forwarding problem. Given m distributed search sites, we cast the problem as m^2 independent binary decision problems arising from every possible site pair. Thus, each local site s_i has m available binary classifiers $c_{i,1}, c_{i,2}, \ldots, c_{i,m}$, where $c_{i,j}$ is in charge of deciding, for any query received at site s_i, whether it is worth processing the query on the index residing at site s_j. We present the type of machine learning features we consider in Section 5.1, and we evaluate them in Section 5.2. We detail the query forwarding architecture in Section 5.3 and report on its performance in Section 5.4.

Interplay with document replication. Query forwarding in multi-site search engines typically determines where to forward a query according to the likelihood of a remote site to index some of the best-matching documents of the query [3]. In Section 4.3, we saw that individual replication outperforms identical replication. However, in the individual replication strategy, documents are usually replicated on a subset of the sites. Because of this, determining the optimal subset of sites to which the query should be forwarded becomes much more involved. Since this issue is beyond the classical query forwarding problem we aim to solve, in the rest of the paper, we assume that query forwarding is coupled with the identical replication strategy.[9]

[8]For easier visualization, Figure 3 assumes a fixed replication budget $b = 8\%$. Other values of b exhibit a similar behavior.

[9]The above-mentioned problem does not exist in case of identical replication, where a replicated document becomes locally accessible in all sites, eliminating the need for query forwarding.

Table 1: Feature categories

Type	Category	#
Pre-retrieval	Term lengths	8
	Term IDFs	20
	Term scores	32
	Query language	1
	Query popularity	15
	Query performance	12
Post-retrieval	Local query score	8
	LP forwarder decision	4

5.1 Feature Extraction

Given the model described above, each query q submitted to a site s_i can be seen as the source of m different machine learning instances $q_{i,1}, \ldots, q_{i,m}$, where the data instance $q_{i,j}$ will solely be used to train the binary classifier $c_{i,j}$ and contains features about (i) the local site s_i, (ii) the remote site s_j, (iii) some third sites s_k, where $k \notin \{i, j\}$,[10] and (iv) any meaningful combination of the previous options. Finally, the label (i.e., our ground truth) of every query instantiation $q_{i,j}$ is a binary variable indicating whether site s_j contains at least one document from $\mathcal{R}(q) \setminus (\mathcal{D}_i \cup \mathcal{R}_i)$, i.e., a document that should be in the top-k result set returned to the user and not available in neither index in site s_i.

Table 1 shows a summary of the different feature categories that we have considered. It is important to note that pre-retrieval features can be extracted prior to the local processing of the query, whereas post-retrieval features can be computed afterwards. We next give a brief description of these different categories.

Term lengths, IDFs and scores. We extract some basic features about the query and the terms in it. We also use some per-term and per-site frequency statistics as well as information on the maximum contribution of any term appearing in the site index to the score of a query containing the term.[11] As previously mentioned, we extract a number of features from that information, including features that try to capture the possible contribution of third sites to the result set.

[10] For instance, by encoding the maximum score that the lowest-scoring term is able to achieve at a third site, we expect our query forwarder to be more informed than the previous thresholding models [3, 13], which by nature only take into account information from the local-remote site pair.

[11]As pointed out in [12], the storage overhead of keeping this *site* × *term* score matrix is negligible.

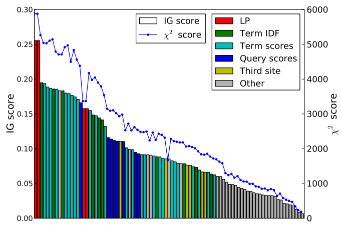


Figure 4: Individual feature scores.

Query language. Another feature which can be intuitively expected to capture the relevance of the indexes in different sites is the language of the query, especially given that documents are assigned to sites using language as a feature. We try to predict the language of the query by using a character-level, n-gram-based predictor.
Per-site query popularity. A simple set of features encoding information about whether the query belongs to the most frequent queries issued to the local site s_i, for different percentile-based definitions of most frequent queries.
Query performance. We also extract some features related to the query performance predictors: γ_1 and γ_2, which predict how good the results of a query will be on a given index [18]. These predictors are based on the distribution of IDF values over different terms of a query and can be easily computed in the pre-retrieval stage.
Local query score. This set contains a number of features related to the scores obtained after processing the query on the local index.
LP forwarder decision. We compute some features using the LP-based query forwarding model described in [13].

5.2 Feature Evaluation

We investigate the importance of individual query features by computing both the informational gain (IG) and chi-squared (χ^2) scores of each feature. IG measures the reduction of entropy when the feature value is known, and χ^2 measures the lack of independence between a feature and class values. Both metrics have been previously reported to perform well as feature selection methods [24].

Figure 4 shows IG and χ^2 scores averaged over the query instances arising from different site pairs, and are ordered by decreasing IG score types. The first thing to note is that the relative performance of individual features is roughly the same for both scores. As expected, the most informative features are those extracted from the LP formulation. A bit surprisingly, the post-retrieval query score features based on the local processing of the query are less informative than pre-retrieval features based on term scores and term IDFs. However, we note that we are only measuring the performance of features when they are considered in isolation. Therefore, the informativeness of some feature combinations might not be properly captured. It is also worth noting that features capturing information of term scores in sites other than the local-remote pair (labeled as the third site features) carry a certain amount of information. Finally, let us note that the features showing the worst performance are those related to query popularity, on the one hand, and to query language, on the other hand. This is probably due to the difficulty of accurately predicting the language of queries, given their very short length.

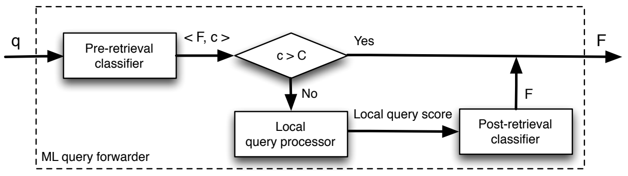


Figure 5: Architecture of the `ML` query forwarder.

5.3 Machine-Learned Query Forwarder

Figure 5 illustrates the proposed machine-learned query forwarder. The forwarder is composed of two different classifiers. On one hand, we have a pre-retrieval classifier which only uses pre-retrieval features and thus can be run right upon the reception of the query. Since this classifier dismisses post-retrieval features, however, it will predictably be less accurate than the full post-retrieval classifier, which employs all available features. In order to exploit the trade-off between the result quality and efficiency, we choose to trust the set F of forwarding decisions made by the pre-retrieval classifier only if the confidence score $c \in [0.5, 1]$ of the classifier is high enough, i.e., higher than a given threshold C.[12] The `ML` architecture is thus parametrized by the confidence threshold C, and different values of C will result in query forwarders with different properties: lower thresholds will be able to trigger the forwarding process earlier, but at the cost of being less precise and thus producing both worse results and a higher chance of unnecessarily overloading remote sites. Also, we note that this architecture gives us the option not to process the query locally, as opposed to previous approaches.

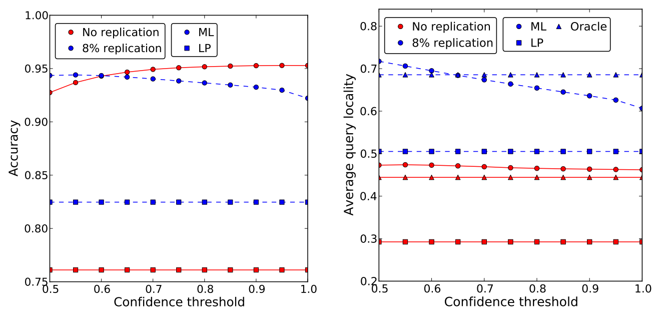
As the classification model, we choose to combine a set of 100 base decision trees through the bagging ensemble method [9]. We use an off-the-shelf implementation of both the bagging method and the fast REP classification tree provided by the WEKA package [17]. In our preliminary experiments, other machine learning approaches such as random forest classifiers showed slightly better accuracy rates, but their training time was significantly larger. Given the large number of classifiers and the size of the datasets, we chose to use the bagging classifier.

In order to assess the value of using large training sets, we analyze the accuracy of our pre-retrieval classifiers as the amount of training queries increases, in the no-replication scenario. We hold out a fraction of our training dataset for testing purposes and train the forwarders with subsets of varying size from the remaining fraction. The resulting accuracies (averaged over all different site pairs) do not attain any saturation point before reaching the maximum training set size. Increasing the number of training queries from 10K to 100K, for instance, raises the accuracy from 88.72% to 90.86%. Using all the available queries results in a further boost up to 92.75% accuracy. We thus only report henceforth on query forwarders trained with the full training set.

5.4 Experiments on Performance

We now report the accuracy and the query locality metric attained by our classification strategy and defer the discus-

[12]Note that this confidence score is actually the estimated probability of the decision to be right. Hence, since we are dealing with a binary decision problem, the score always lies in the range $[0.5, 1]$.



(a) Accuracy. (b) Query locality.

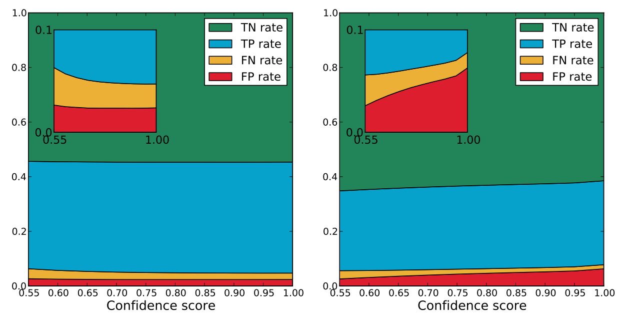
Figure 6: Performance of different query forwarders.

Table 2: Confusion matrix values (%)

	Replication budget (b)							
	0%				8%			
	TP	TN	FP	FN	TP	TN	FP	FN
ORACLE	42.97	57.03	0.00	0.00	32.22	67.78	0.00	0.00
LP	42.97	33.14	23.89	0.00	32.21	50.26	17.53	0.00
$\mathtt{ML}_{0.5}$	38.61	54.14	2.89	4.36	28.56	65.79	1.99	3.66
$\mathtt{ML}_{0.75}$	40.41	54.67	2.36	2.57	30.41	63.43	4.35	1.81
$\mathtt{ML}_{1.0}$	40.64	54.64	2.39	2.33	30.72	61.49	6.29	1.50

sion of other metrics to Section 7. Figure 6(a) compares the accuracy of our ML forwarder against that of the LP baseline, varying the confidence threshold and assuming two different replication scenarios (no replication and 8% identical replication). Table 2 further presents the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) rates ($\mathtt{ML}_C$ stands for our ML forwarder with confidence threshold C). We note that due to the false positives, in some cases the ML forwarder has higher query locality than ORACLE. In both replication scenarios, the accuracy of the ML forwarder significantly exceeds that of the LP forwarder (for any confidence threshold), due to the high FP rate of the LP model. Unexpectedly, in the replicated scenario, the accuracy decreases as the confidence threshold increases, i.e., as more queries are forwarded according to the post-retrieval query forwarder. To investigate this issue, in Figure 7, we show a graphical breakdown of the evolution of the confusion matrix values for increasing confidence threshold values. The accuracy decrease in the replicated scenario turns out to be caused by an increase in the FP rate, although the FN rate actually decreases. This might be due to the higher imbalance caused by document replication. A potential solution is to weight false positives and false negatives non-uniformly, e.g., penalizing false positives more than false negatives when training the classifiers. We explore this possibility and its outcome in Section 7.

Figure 6(b), on the other hand, compares the query locality achieved by different forwarding strategies. As expected, both LP and ML forwarders are able to exploit the higher query locality allowed by the document replication. Compared to the LP forwarder, however, the lower FP rates of our approach ensure significant increase in query locality (10%–20% more queries are locally processed). Again, we note that in the replicated scenario, the increase in the FP rate that we just mentioned has a negative impact on the query locality for high confidence thresholds.



(a) No replication. (b) 8% replication.

Figure 7: Confusion matrix values. The inset details the lower section of the graph for easier visualization.

6. RESULT CACHING

Result caching is an important technique for improving the performance of search engines. The results of recurrent queries can be served by the cache, eliminating the need to process the query on the index. In previously proposed multi-site search architectures, each site typically maintains its own result cache, which contains the results of queries issued to itself. In this section, we are interested in designing cache architectures for multi-site search engines, where the search sites coordinate among each other to decide when and where query results should be cached. The proposed architectures further reduce the need for query forwarding. We limit ourselves to infinite-capacity caches, as assumed in [1]. To maintain the freshness of results served by the cache, we expire cached entries with a simple time-to-live (TTL) mechanism.

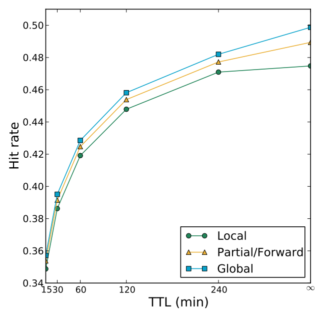
6.1 Result Caching Strategies

Local cache. A straightforward, yet common result caching approach in multi-site search architectures is to assume a local, independent cache in each search site [13]. When a site receives a query whose results are already cached, the results are served immediately by the local cache if the age of the cached results is less than a predefined TTL value. Otherwise, the query could be processed locally and also forwarded to non-local sites depending on the decision of the query forwarder. The lack of coordination between the caches cause certain queries to be processed redundantly although their results are already cached in one or more search sites.

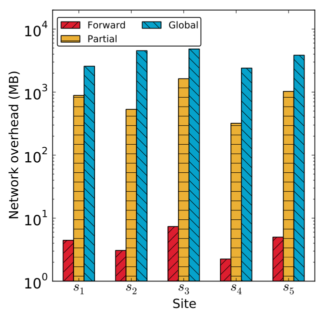
Global cache. An effective way to reduce the number of queries that require processing is to replicate cached query results on all sites. Once a query is added to the local cache of a site, it is sent to the rest of the sites and cached also remotely.[13] This approach ensures that any site can serve the query results from its cache (until the TTL expires).

Partial cache. In global caching approach, certain cache entries are replicated on sites where they will never be requested. Moreover, the storage and transmission overheads are high. To alleviate these problems, the results of a query may be replicated only on the sites where the query is forwarded during the query processing. The intuition behind

[13]This can be achieved either on a per-query basis or in batch mode, according to the desired trade-off between the hit rate and the transmission cost.

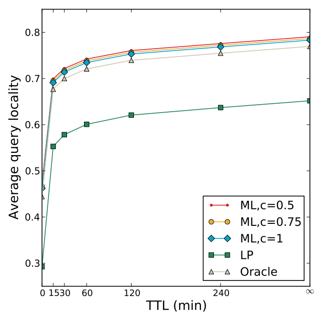


(a) Hit rate w.r.t. TTL.

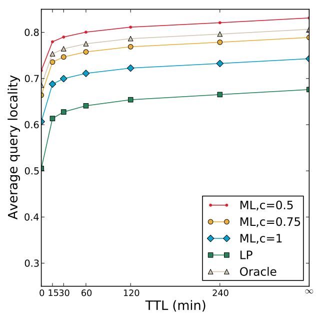


(b) Network overhead.

Figure 8: Performance of different caching strategies.



(a) No replication.



(b) 8% replication.

Figure 9: Impact of global cache on query locality.

this partial caching approach is that, since the contacted sites are more likely to index documents relevant to the query, the query will be issued to them with higher probability.

Forward cache. A variation of the above approach to reduce the transmission overhead is that, whenever a query is forwarded from site s_i to site s_j, s_j caches a pointer to site s_i. If s_j receives the same query in the future, it simply requests the results from the cache on site s_i instead of processing the query. The validity of the cached pointer entries are expired by the same TTL mechanism used before. Queries that are processed locally are maintained in the cache as in the local caching approach.

6.2 Experiments on Performance

In this section, we evaluate the performance of different result caching strategies. Figure 8(a) shows the hit rate as the value of TTL increases. As expected, the global caching approach consistently achieves the highest hit rate compared to other strategies, as the results of queries are replicated across all caches. Taking a TTL of 2 hours as an example, the average hit rate of in case of a global cache is 3.4% higher than that in case of a local cache, and 1.1% higher than those in case of partial cache and forward cache.[14] In Figure 8(b), we observe that the global cache incurs higher network and storage overheads. Nevertheless, since our caches have unlimited capacity and the transmission of cached results between sites can be performed offline, we choose to focus only on the performance of the global caching approach in the rest of the paper.

In Figure 9, we show that the query locality increases along with the TTL value, for both replication budgets (0% and 8%) and query forwarding strategies that we take into consideration. More importantly, we observe that the proposed `ML` query forwarder outperforms the baselines for all TTL values. Besides, even if the difference between not using a cache (i.e., zero TTL) and using a cache (e.g., a TTL of minutes) decreases when documents are replicated, using a cache still improves the query locality by at least 9.3% for different query forwarding strategies.

7. GLOBAL RESULTS

In this section, we report the performance in terms of result quality, query response time, and system workload. We first report on these metrics independently and then study the combined effect of different strategies.[15] Table 3 shows some result quality metrics for our `ML` forwarder assuming three different confidence thresholds $C \in \{0.5, 0.75, 1.0\}$ at different rank positions $p \in \{1, 5, 10\}$. *O@p*, *N@p* and *E@p* stand for overlap@p, NDCG@p and EMR@p, respectively. The results are consistent with what we saw in the previous sections: as the confidence threshold increases, the quality improves (according to all metrics). This is because the post-retrieval forwarder, which has a smaller FN rate, is used more often.

In Figure 10, we analyze the variation of query response time across different queries. The figure shows the cumulative percentage of queries that are answered below a certain time threshold. In the scenario with replication, for instance, 53.51% of the test queries are answered in less than 300ms when using the `LP` forwarder, whereas by using our `ML` strategy, this percentage is increased to a value in the 62.09%–71.43% range, depending on the confidence threshold.

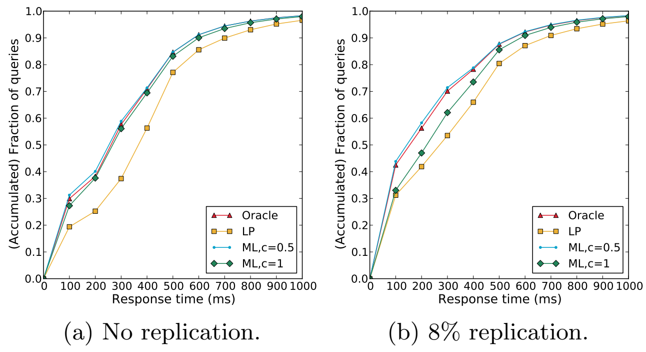
We now turn our attention to the trade-off between both metrics, this time taking into account the impact of caching strategies (Figure 11(a)). In the figure, the solid markers denote a simulation without caching while the hollow markers include the effect of a global cache with a two-hour TTL. The markers from left to right correspond to `ML` forwarders with increasing confidence threshold $C \in [0.5, 1]$. As expected, the average response time increases as the result quality increases. Without caching and taking the `LP` forwarder as a baseline, the proposed `ML` forwarder achieves a significant reduction in average response time: in the 20%–22% range (no replication) or 15%–25% (8% replication), depending on

Table 3: Quality metrics at different result positions (p)

		Replication budget (b)					
		0%			8%		
C	p	O@p	N@p	E@p	O@p	N@p	E@p
0.5	1	0.965	0.965	96.52	0.970	0.971	97.04
	5	0.957	0.959	94.41	0.964	0.965	95.33
	10	0.934	0.945	82.73	0.944	0.953	85.54
0.75	1	0.983	0.983	98.32	0.991	0.991	99.07
	5	0.978	0.979	97.07	0.987	0.988	98.21
	10	0.964	0.971	89.22	0.977	0.982	92.34
1.0	1	0.988	0.988	98.84	0.997	0.997	99.69
	5	0.983	0.984	97.61	0.994	0.994	98.92
	10	0.970	0.976	90.23	0.984	0.989	93.66

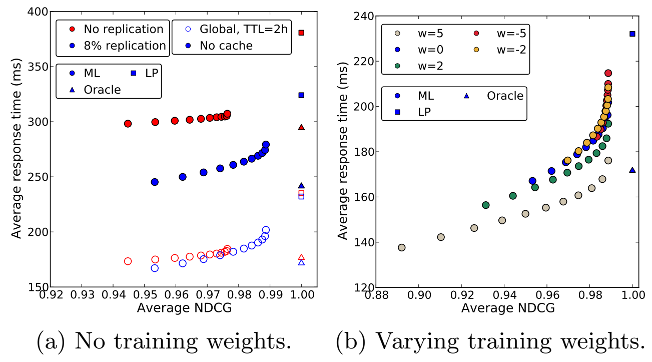
[14]We assume that the queries are forwarded by the `ORACLE` forwarder.

[15]Note that caching has no impact on the result quality.



(a) No replication. (b) 8% replication.

Figure 10: Distribution of the query response time.



(a) No training weights. (b) Varying training weights.

Figure 11: Trade-off between result quality and time.

the confidence threshold, in exchange for a small loss in result quality. If we focus on the scenario with replication and pick a confidence threshold of 0.7, our approach offers an improvement of 19.5% in the average response time while the average NDCG remains as high as 0.978, and 91.65% of queries obtain a result set identical to that of a centralized architecture. We omit the relative improvements in query processing workload since they follow a similar pattern.

Regarding the impact of the cache layer, we observe that caching neutralizes the advantages of replication as far as the response time is concerned, up to the point where larger index sizes due to replication and the induced larger query processing time outweighs the benefits achieved by the replication itself. This might be partly due to the fact that the replication strategy is specifically tailored to query frequencies in the original query stream, which is largely modified by the presence of a cache layer. To circumvent this problem, one should simulate a fixed caching strategy paired with a fixed TTL value and tailor the replication decisions to the resulting, post-cache query stream.

Varying training weights. In Section 5.4, we hinted at an alternative, orthogonal way of exploring the quality-time trade-off that consisted of modifying the balance between false positives and false negatives by weighting them non-uniformly while training the query forwarder. We show the results of applying this strategy in Figure 11(b), where for easier visualization only a scenario with 8% replication coupled with a global cache with a two-hour TTL, is depicted. $w = x$ stands for a `ML` forwarder such that during the training phase an FP is considered x times as negative as an FN (e.g., the $w = -2$ stands for those forwarders where an FN is considered twice as harmful as an FP). It can be seen that this strategy further widens the flexibility that the `ML` forwarder offers to the system designer. In exchange for a larger quality loss, the average query response time can now be decreased to as little as 137ms (a 40% reduction with respect to the `LP` forwarder). More interestingly, on the other extreme, we can reach a 0.99 NDCG quality and decrease the average response time to 176ms, which is almost as good as the `ORACLE` forwarder. This means a significant reduction of 24% with respect to the state-of-the-art `LP` forwarder.[16]

8. RELATED WORK

There is a rich literature on query forwarding, document replication, and result caching. Herein, we provide a brief survey of related work on these topics, limiting the context to multi-site web search. A broader survey on scalability and efficiency of web search engines can be found in [10].

Query forwarding. In the multi-site web search framework, the query forwarding problem is first investigated by Baeza-Yates et al. [3]. The authors propose a simple technique for estimating the maximum score a query can achieve on each search site. The forwarding decisions are made based on the outcome of a comparison between the estimated maximum scores and the lowest score in the top k result set computed by the local search site. The authors show that, with the proposed technique, many queries can be processed solely in local search sites without sacrificing the result quality. The work of Cambazoglu et al. [10] further increases the rate of locally processed queries by using a linear-programming technique. The technique exploits the co-occurrence of query terms in documents and yields tighter upper bound score estimations. In our work, we use a machine-learned query forwarding technique, which achieves further performance improvements over the linear programming model used in [13]. Finally, Baeza-Yates et al. [4] also investigate the use of machine learning techniques in a distributed search engine, although the scope of their simulation is much more limited and their focus is on two-tiered systems with high index size imbalance.

Document replication. There are two prior studies on document replication in multi-site web search engines [8, 21]. Brefeld et al. [8] employ machine learning techniques to select the data center(s) where a newly discovered document should be replicated. Since no past popularity information (i.e., the number of views in web search results) is available for new documents, the features used by the learning model are mainly extracted from the document content (e.g., the language feature). Kayaaslan et al. [21] consider a scenario where past popularity information is available for every document. They propose three algorithmic optimizations that aim to reduce the response latency, query processing workload, and the loss in result quality. The replication heuristic proposed in our work achieves further improvements over the heuristic described in [21] in terms of the query locality metric. Besides [8] and [21], Cambazoglu et al. [13] also evaluate a simple replication heuristic, where a small fraction of frequently viewed documents are replicated on all search sites. However, document replication does not form the main focus of their work. Finally, Blanco et al. [7] investigate the problem of assigning documents to search sites with the most similar content, obtaining a one-to-one mapping between the newly crawled documents and search sites.

[16]These response time gains (not shown in the figure) are virtually the same even without the cache layer.

Result caching. Result caching is long recognized as an important technique for improving search engine performance [22]. So far, various caching policies are proposed to increase the hit rates [2, 15] or to reduce the query workload of search backends [16, 23]. Recent studies mainly focus on refreshing result caches [11, 20] and cache invalidation [1, 5, 6]. To the best of our knowledge, our work is the first to investigate result caching strategies in the context of multi-site web search.

9. CONCLUSIONS

In this paper, we studied the interplay between three important mechanisms in a multi-site web search engine: document replication, query forwarding, and result caching. For each of these mechanisms, we proposed new strategies that outperform the existing state-of-the-art approaches. Most notably, we explored the fundamental trade-off between result quality, on the one hand, and query response time and system workload, on the other hand. The use of a machine-learned query forwarding strategy can allow system designers to exploit this trade-off and obtain significant improvements in query response time and overall system workload, in exchange for a little quality loss. We note that our machine learning technique can be improved through active learning on forwarding mistakes, i.e., adding false positives and false negatives to the training data.

10. ACKNOWLEDGEMENTS

This work was supported by the LEADS project (ICT-318809), funded by the European Community, and the Torres Quevedo Program from the Spanish Ministry of Science and Innovation, co-funded by the European Social Fund.

11. REFERENCES

[1] S. Alici, I. S. Altingovde, R. Ozcan, B. B. Cambazoglu, and O. Ulusoy. Timestamp-based result cache invalidation for web search engines. In *Proc. 34th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 973–982, 2011.

[2] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. The impact of caching on search engines. In *Proc. 30th Int'l ACM Conf. Research and Development in Information Retrieval*, pages 183–190, 2007.

[3] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Plachouras, and L. Telloli. On the feasibility of multi-site web search engines. In *Proc. 18th ACM Int'l Conf. Information and Knowledge Management*, pages 425–434, 2009.

[4] R. Baeza-Yates, V. Murdock, and C. Hauff. Efficiency trade-offs in two-tier web search systems. In *Proc. 32nd Int'l ACM Conf. Research and Development in Information Retrieval*, pages 163–170, 2009.

[5] X. Bai and F. P. Junqueira. Online result cache invalidation for real-time web search. In *Proc. 35th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 641–650, 2012.

[6] R. Blanco, E. Bortnikov, F. Junqueira, R. Lempel, L. Telloli, and H. Zaragoza. Caching search engine results over incremental indices. In *Proc. 33rd Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 82–89, 2010.

[7] R. Blanco, B. B. Cambazoglu, F. P. Junqueira, I. Kelly, and V. Leroy. Assigning documents to master sites in distributed search. In *Proc. 20th ACM Int'l Conf. Information and Knowledge Management*, pages 67–76, 2011.

[8] U. Brefeld, B. B. Cambazoglu, and F. P. Junqueira. Document assignment in multi-site search engines. In *Proc. 4th ACM Int'l Conf. Web Search and Data Mining*, pages 575–584, 2011.

[9] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[10] B. B. Cambazoglu and R. Baeza-Yates. Scalability challenges in web search engines. In M. Melucci, R. Baeza-Yates, and W. B. Croft, editors, *Advanced Topics in Information Retrieval*, volume 33 of *The Information Retrieval Series*, pages 27–50. Springer Berlin Heidelberg, 2011.

[11] B. B. Cambazoglu, F. P. Junqueira, V. Plachouras, S. Banachowski, B. Cui, S. Lim, and B. Bridge. A refreshing perspective of search engine caching. In *Proc. 19th Int'l Conf. World Wide Web*, pages 181–190, 2010.

[12] B. B. Cambazoglu, V. Plachouras, F. Junqueira, and L. Telloli. On the feasibility of geographically distributed web crawling. In *Proc. 3rd Int'l Conf. Scalable Information Systems*, pages 31:1–31:10, 2008.

[13] B. B. Cambazoglu, E. Varol, E. Kayaaslan, C. Aykanat, and R. Baeza-Yates. Query forwarding in geographically distributed search engines. In *Proc. 33rd Int'l ACM Conf. Research and Development in Information Retrieval*, pages 90–97, 2010.

[14] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[15] T. Fagni, R. Perego, F. Silvestri, and S. Orlando. Boosting the performance of web search engines: caching and prefetching query results by exploiting historical usage data. *ACM Trans. Inf. Syst.*, 24(1):51–76, 2006.

[16] Q. Gan and T. Suel. Improved techniques for result caching in web search engines. In *Proc. 18th Int'l Conf. World Wide Web*, pages 431–440, 2009.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[18] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *String Processing and Information Retrieval*, pages 43–54. Springer, 2004.

[19] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[20] S. Jonassen, B. B. Cambazoglu, and F. Silvestri. Prefetching query results and its impact on search engines. In *Proc. 35th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 631–640, 2012.

[21] E. Kayaaslan, B. B. Cambazoglu, and C. Aykanat. Document replication strategies for geographically distributed web search engines. *Inf. Process. Manage.*, 49(1):51–66, 2013.

[22] E. P. Markatos. On caching search engine query results. *Computer Comm.*, 24(2):137–143, 2001.

[23] R. Ozcan, I. S. Altingovde, and O. Ulusoy. Cost-aware strategies for query result caching in web search engines. *ACM Trans. Web*, 5(2):9:1–9:25, 2011.

[24] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. 14th Int'l Conf. Machine Learning*, pages 412–420, 1997.