Assigning Documents to Master Sites in Distributed Search

Roi Blanco
Yahoo! Research
Barcelona, Spain
roi@yahoo-inc.com

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

Flavio P. Junqueira
Yahoo! Research
Barcelona, Spain
fpj@yahoo-inc.com

Ivan Kelly
Yahoo! Research
Barcelona, Spain
ivank@yahoo-inc.com

Vincent Leroy
Yahoo! Research
Barcelona, Spain
leroy@yahoo-inc.com

ABSTRACT

An appealing solution to scale Web search with the growth of the Internet is the use of distributed architectures. Distributed search engines rely on multiple sites deployed in distant regions across the world, where each site is specialized to serve queries issued by the users of its region. This paper investigates the problem of assigning each document to a master site. We show that by leveraging similarities between a document and the activity of the users, we can accurately detect which site is the most relevant to place a document. We conduct various experiments using two document assignment approaches, showing performance improvements of up to 20.8% over a baseline technique which assigns the documents to search sites based on their language.

Categories and Subject Descriptors

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

General Terms

Design, Experimentation, Performance

Keywords

Multi-site web search engine, distributed index, document assignment

1. INTRODUCTION

Web search requires building very large data centers, comprising a large number of computer resources [2]. As the Web grows in size and extent, it becomes necessary for a search engine to increase its capacity, and one typical solution is to rely upon more resources of an existing data center. One important drawback of this solution is requiring larger data centers over time, which presents severe limitations regarding space, energy provision, and connectivity. A viable alternative is to design the search engine to enable new, smaller data centers to be incorporated when more capacity becomes necessary. This design has the advantage of not requiring data centers to grow over time. Designing a search engine to distribute its functionality across multiple data centers, however, is not trivial since most techniques used in commercial search engines to date rely upon tightly coupled computer resources to operate efficiently. Recent work has focused on moving away from this paradigm and devising techniques that enable the design of efficient distributed search engines [1, 6].

A distributed search engine is comprised of geographically distributed data centers (sites), and users submit queries to the closest location, where closest can be either with respect to geographic or network distance. Upon receiving a user query, a site either processes a query locally or forwards it to other sites for processing. Techniques for deciding when to forward a query are consequently critical to the performance of such a distributed search engine [1, 7].

Forwarding queries, however, introduces extra latency to the processing of a query, so ideally a site is able to process many of its received queries locally to avoid the extra latency of processing remotely. To enable sites to process many queries locally, it is necessary to select documents to index in a site that are among the top results of queries that are submitted to the site. At the same time, to avoid replicating all documents across all sites, which leads to sites growing with the Web, it is necessary that we carefully select the documents that a site indexes. Such a mechanism to select documents for a site must use some metric of relevance to determine the importance of documents to a given geographic region. Some previous work has proposed techniques for distributed index construction [12, 14] and partitioning [10, 11, 13, 15] over a cluster of computers within a search site, thus targeting a local setting. The only work we are aware of that targets distributed settings is the one of Brefeld *et al.*, which uses machine-learning techniques to replicate documents across sites and are able to forward only 4% of the queries while indexing in each site 55% of documents, on average [4].

Contributions. We argue for an approach consisting of two steps for the construction of the distributed indexes. The first step is assigning a document to exactly one site,

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

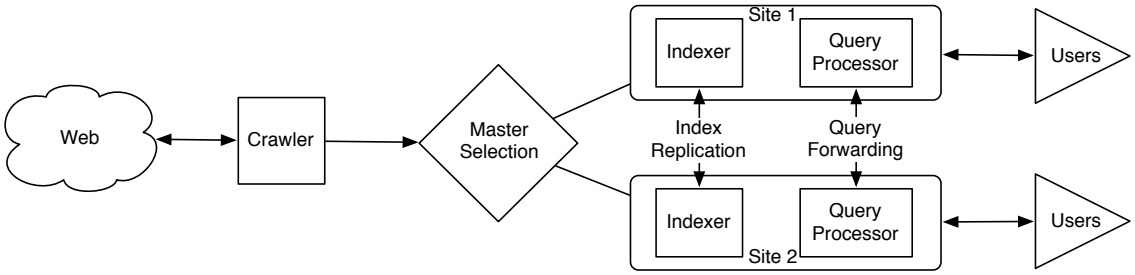

Figure 1: Architecture of a distributed search engine

the document's *master site*. The master site is responsible for indexing the document, which ensures that each crawled document exists in the global search index. The second step consists of enabling sites to replicate documents in a site that is not the master site to enhance the performance of the search engine. Having a master for each document simplifies the design of the replication strategy: each site can make local decisions about replication and document masters guarantee that at least one site indexes a given crawled document. As our experiments show, assigning documents to sites using document content alone is difficult and consequently popularity information is key for efficient replication. Separating mastership and replication, consequently, leads to a more efficient utilization of indexer resources, since sites replicate documents only once they have sufficient evidence of interest from their local users.

This work focuses on the selection of document masters and replication techniques are outside the scope. An efficient master selection strategy assigns documents to sites where users are most likely to access them. Having master sites also enables other sites to arbitrarily add and delete documents for which they are not masters without compromising the coverage of the search engine. Since master sites index the documents assigned to them, we guarantee that every crawled document is indexed by at least one site.

In our evaluation, we consider three approaches to partition the index across master sites. The first one assigns documents to sites using their language, and we use it as a baseline. The second approach uses KL divergence to evaluate the similarity between the distribution of terms in the new document and a description of each site. This description can be either extracted from the queries of the users, or from the documents given as results. Finally, we consider a metric based on cache invalidation to estimate the impact of the document on each site of the search engine. We evaluate these techniques over a large document collection and a query set obtained from a commercial search engine. Our results indicate a 20.8% overall improvement over our baseline, as well as up to 88% of the optimal performance under some document popularity conditions. This result provides evidence that a tight integration with the search engine with access to user feedback significantly improves the accuracy of documents' placement.

Roadmap. The remainder of this paper is organized as follows. Section 2 discusses the high-level architecture we consider for distributed search engines. In Section 3, we present the document assignment techniques we evaluated. We describe the experimental setup in Section 4 and report our results in Section 5. Section 6 contains a discussion of practical implementation aspects. Finally, we discuss related work in Section 7 and conclude the paper in Section 8.

2. ARCHITECTURE

We consider a distributed search engine deployed in several regions across the world. We refer to the sites of a distributed search engine as $\mathcal{S}$, while $\mathcal{D}$ is the set of documents it processes. Fig. 1 presents an overview of the search engine. We now describe the different components of the search engine as well as their interactions.

2.1 Crawler

A search engine discovers and acquires content through a crawler. The main research challenge in multi-site Web crawling is to partition the Web document collection across multiple search sites so that pages can be crawled faster, due to the improved network proximity between the search sites and Web servers. Gao *et al.* formulates this problem as a geographically focused Web crawling problem [8].

In this paper, we do not evaluate crawling performance. Hence, for the sake of simplicity, we consider the case of a single crawler. Once a document has been fetched by the crawler, it is sent to an indexer in order to be added to the inverted index of the search engine. The transition from the crawler to an indexer is the main focus of this paper. The index is divided among the sites of the distributed search engine. Hence, the crawler has to decide which sites should index the document. In Brefeld *et al.*'s work [4], the crawler selects multiple sites for each document through a machine learning model. In this paper, we propose an approach to select a unique site for each document. We call this site the master site of the document. We formally define this problem in Section 3.1.

2.2 Query processor

Processing queries in a distributed search engine is a challenging task. The result quality is crucial to ensure user satisfaction. The results computed by the search engine should not be affected by the distribution of the index among several sites. The main technique investigated so far is query forwarding [1, 7]. A search site first computes results using its local index and then determines through heuristics if the other sites may have different documents that would be ranked higher. If that is the case, the query is forwarded in order to compute more accurate results. The heuristic may overestimate the scores of the other sites to achieve a result similar in quality to that of a centralized setup.

While forwarding a query can improve the quality of the results, this may increase the response time of the search

engine as well as the computational load. It is therefore crucial to reduce the number of forwarded queries. This can be done both by improving the accuracy of the score estimation heuristics and by ensuring that documents relevant to the users of a site are present in its local index. In this paper, we assume the existence of a query forwarding algorithm with an associated heuristic [1, 7]. The content of the index of a site is the direct consequence of the master selection performed by the crawler. Hence, we evaluate the accuracy of the master selection process by computing the proportion of results served by a site that was present in its local index.

2.3 Indexer and replicator

An indexer extracts terms from documents and generates an inverted index in order to match documents to queries. Each site has its own indexer which receives documents from the crawler according to the master selection process. An indexer has a maximum capacity, due to storage, memory, or computational limits. Artificially limiting the size of an index can also be an efficient way of reducing the query response time, as smaller indexes are faster to process. Each document is assigned to a master which is responsible for keeping the document in its index. The forwarding mechanism implemented by the query processor ensures that a document matching a query is always taken into account in ranking, even when the document is indexed in another site.

The popularity of Web pages follows a power law, a few Web pages are extremely popular and constitute most of the search results. If these documents were indexed only by their master site, any query matching one of them and issued at another site would always have to be forwarded. This would degrade the performance of the search engine. As a consequence, the master selection process is used in conjunction with a replication algorithm. After indexing the documents that are assigned to itself, each site uses its remaining index capacity to index other documents that are assigned to other sites but are frequently accessed by its users. This process replicates popular documents at each site. Still, contrary to Baeza-Yates *et al.*'s approach [1], the set of replicated documents is not statically determined and common to all sites. Each site can locally decide which documents to replicate, and this decision takes into account the activities of the users of that site in particular. The replication decisions are local. As each document has a master that maintains it in its index, a replicated document can be deleted without synchronization with the other sites, it will not be evicted from the distributed search engine.

This paper focuses on the problem of selecting a master for each document. While the replication mechanism is crucial for the performance of the search engine, we leave this study as part of future work. Since the master of a document is in charge of maintaining it, it is important to accurately assign it to the site where users are most likely to access it.

3. DOCUMENT ASSIGNMENT

3.1 Problem definition

We consider the problem of selecting exactly one site for each document while crawling. This site is referred to as the *master* of the document, and it is responsible for keeping this document in its index. The existence of a master site guarantees the recall of the search engine and simplifies the index replication process.

$$master : \mathcal{D} \rightarrow \mathcal{S}$$

Without loss of generality, defining the choice of a master site is equivalent to providing the crawler with a function *mScore* such that:

$$mScore : \mathcal{D} \times \mathcal{S} \rightarrow \mathbb{R}$$
$$master(D_j) = \arg\max_{S_i \in \mathcal{S}} mScore(D_j, S_i)$$

The goal of the master selection process is to place documents in sites where they are likely to be requested by users. Therefore, $mScore(D_j, S_i)$ should produce higher scores when it predicts that the document D_j matches the interests of the users of site S_i.

3.2 Document language

The language of a document is a good indicator of which region could be interested in it. Regional documents are more likely to be requested from the countries whose language is the same. Brefeld *et al.* [4] use the languages as one of the features of a machine learning algorithm. They observe that the language of a document has a large impact in the precision of the document's placement. In our experiments, we use a commercial language classifier to assign a language to each document.

Using past search engine activity, we compute the following conditional probability: given that a document in a language L_j is returned as a result to a user, what is the probability that the user is from site S_i? The index partitioner computes the language of new documents and assigns each of them to the site with the highest probability. In this case, we approximate *mScore* with $p(S_i|L_j)$. In the evaluation, we refer to this solution as LANG.

3.3 Document assignment by likelihood

While the language of a document leverages, to some extent, the content of the document, the language is a coarse-grain information. It would be possible to achieve a better precision by taking into account all the terms in the document. Hence, we propose to compute *mScore* using a probability estimation. In general, the optimum placement for a document D_j at a master site S_i depends on the queries that will be issued in the future Q_i^f in which D_j appears among the top results. We make use of two sources of information: C_i is a random vector that represents the content of site S_i, and Q_i is a random vector representing the most recent recent query stream processed by S_i. Both these informations can be extracted from the cache of the search engines or its query logs. That is, we want to select the most probable site maximizing the following log-likelihood:

$$\log p(S_i|D_j) = \int \log p(S_i|Q_i^f, D_j) p(Q_i^f|D_j) dQ_i^f$$
$$= \int \log p(C_i, Q_i|D_j, Q_i^f) + \log p(Q_i^f|D_j) dQ_i^f .$$

In order to estimate Q_i^f, we make use of Q_i as an approximation of the future query stream, and we have

$$\log p(S_i|D_j) \approx \int \log p(C_i, Q_i|D_j, Q_i) + \log p(Q_i|D_j) dQ_i \quad (1)$$
$$\approx \log p(C_i|D_j) + \frac{1}{|Q_i|} \sum_{q_z \in Q_i} \log p(q_z|D_j) \ . \quad (2)$$

Here, we assume that the queries in the stream are conditionally independent given a document.

This formulation has two components: one that uses the contents of a given site and its relatedness to the document $p(C_i|D_j)$ and another one that uses the available query stream Q_i as a source of evidence. We relate the former probability with the *term distribution* of the site, and the latter as the fraction of queries that are *invalidated* by a given document, as explained next. We experiment how both probabilities perform for the problem of master assignment independently and we combine them later on.

3.3.1 Document as a query

Our first classification method stems directly from the estimation of $p(C_i|D_j)$ (Eq. (2)), which assigns a document D_j to a master site S_i based on the likelihood of observing the content C_i at the site S_i given the content of D_j. It is possible to compute the likelihood of the content to the document using a similarity function, in the same way standard language models for IR score documents with respect to their likelihood to a query. In this case, the whole contents of the site plays the role of the document and the document to be indexed as the query. We employ a bag of words representation for the document to be indexed where C_i is the concatenation of all content of S_i. The rationale is that if the contents of the documents returned by a site matches the queries it receives, we should assign a new document to the data center that has the closest term distribution.

To estimate $p(C_i|D_i)$, we employ the language-model-based KL divergence [16], which measures how similar a certain distribution is (query) with respect to a reference distribution (document). We smooth the unigram language model counts of both the document and site contents using Dirichlet priors smoothing [17]. This ranking function contains a tunable parameter μ that controls the amount of probability mass that is assigned to unseen terms:

$$mScore(D_j, S_i; \mu) = KL(D_j||C_i; \mu).$$

In our experiments, we consider two different ways to represent the content of a search site. For the first one, which we call KL-D, we use the content of the documents returned as results to the users of the site. The second approach, KL-Q, directly uses the term distribution of user queries to represent the content of the site, as an approximation of the most recent term distribution. It is important to notice that both of these approaches are directly driven by user activity and not influenced by the master selection process. An alternative approach could be to use the documents indexed by a site to represent its content. A drawback of this approach is that previous indexing decisions would influence the future ones and mis-classification error would affect future decisions degrading the performance over time. We therefore discarded this possibility in our experiments and focus on using the term distribution in the cache contents only.

3.3.2 Cache invalidation

The second classification method estimates the probability of $p(Q_i|D_j)$ (Eq. (1)). The likelihood of the query stream to the document has been studied in the past in the context of *cache invalidation*. Search engines cache the results served to the users in order to avoid computing multiple times the answer to a given query. While this increases the performance of the search engine, it also raises the problem of results staleness [**?**], especially in the case of incremental indexing. Cache entries need to be invalidated in order to reflect the changes in the index. Blanco *et al.* [3] experiment different cache invalidation policies to strike a good balance between the risk of serving stale results and the hit-rate of the cache. In practice, upon adding a document to the index, the search engine matches the terms in the document with queries present in the cache and if the document is expected to be in the top-k of a query, its cache entry is invalidated. Then, we define $p(Q_i|D_j)$ as the fraction of queries in the cache of S_i that would be invalidated by D_j, that is, D_j would be in the top-k results of those queries:

$$mScore(D_j, S_i) = \frac{1}{|Q_i|} \sum_{q_z : Q_i} I(q_z||D_j) \ ,$$

where $I(q_z||D_j)$ is 1 if D_j would be in the top-k results for q_z and 0 otherwise.

The query cache of a search site is a good indicator of the interests of the users in its region. Hence, we propose to use the number of cache invalidation caused by a document at a given site as an indicator of its relevance to this site. If the document would have had a high impact on the results of the past queries, it is likely to appear in the future results as well. The main advantage of this approach is that, contrary to the approach presented in Section 3.3.1, it does not concatenate all the queries into a bag of words. Since the co-occurrence of terms is preserved, we expect cache invalidation to have a slightly better performance. In the remainder of this paper, we refer to the assignment based on cache invalidation as CACHE.

4. EXPERIMENTAL SETUP

4.1 Search engine

4.1.1 Distributed search engine

We consider a distributed search engine, such as the one described in Section 2. We devise a configuration in which the search engine is deployed on five sites $\mathcal{S} = \{S_1 \dots S_5\}$, distributed over different continents. While the sites are located in different countries, some countries have people speaking the same language, which complicates the assignment of documents. We assume that the query processor of the search engine implements a query forwarding scheme (Section 2). As a consequence, given a query and a document collection, the results computed by the distributed search engine are equal to the ones returned by a centralized one, and do not depend on the site that the query originates from. Thus, we perform our experiments using a centralized commercial search engine, but label each query with the site it originates from and label each document with the site that is selected as the document's master.

4.1.2 Ranking function

The ranking function of a search engine is a key component, as it directly impacts the quality of the results returned to the users. While the main focus or our work is not related to the relevance of the results of the search engine, it is important to consider a realistic ranking function because it impacts the distribution of the documents' popularity. We rely on the following ranking function to assign a score to a

document d with respect to a query q:

$$score(d, q) = \lambda \times relevance(d, q) + (1 - \lambda) \times quality(d)$$

The ranking function assesses both the relevance of the document and its quality, and weights them using the λ parameter. We use BM25 to compute the relevance, while the quality of a document is evaluated through a link-based metric. We optimize its parameters using a separate training dataset that contains queries with relevance information and keep the best parameters for our own test evaluation. While BM25 and KL divergence are both effective ranking functions and therefore, exhibit correlations in their behavior, it is important to notice that we use a completely different setup to rank documents in the search engine and to assign documents to sites. Hence, our results are not specific to a ranking function and can be applied in more general cases.

4.2 Dataset

4.2.1 Documents

We use a random sample of 31,599,910 Web pages obtained from the Web crawl of a commercial search engine. We randomly split the documents into a training and a testing set, holding 75% and 25% of the documents, respectively.

4.2.2 Queries

We rely on query logs to simulate user activity. The log consists of consecutive queries issued in a single day to different frontends of a commercial search engine. We only consider queries of less than 20 terms where the users clicks a result on the first page of results. The queries are converted to lower case, and duplicate terms are removed. Each query is labeled with the country of the user who issued the query. We generate a collection of queries for each site of the search engine by keeping queries originating from the country of the site and, when this is not sufficient, from neighboring countries. We sample a total of 7,023,102 queries, which we equally split for training and testing for each region. As queries are sorted in arrival order, the first half of the queries constitute the training set, while the rest forms the test set.

4.2.3 Document popularity

In our experiments, we assume that the top-10 documents returned by the search engine to a query are all relevant. We evaluate the training queries on an index containing the training documents. The popularity of a document is the number of times it occurs in the results of a query. We display the distribution of the popularity of the documents in the training set in Fig. 2. As already observed in many search engine studies (e.g., by Brefeld *et al.* [4]), the distribution follows a power law: a few documents are very popular while many documents have a very low frequency in top search results. From all the documents in the training set, only 4,370,608 appear at least once in the results of the training queries. The documents in the test set exhibit a similar popularity pattern, where 2,729,205 of them appear in the results of the test queries. Given the large amount of diversity in document popularity, it seems that a reactive replication scheme may be more efficient than a static replication scheme, determined at the discovery time of the document. Precisely predicting the popularity of a new document is indeed a difficult problem. In the approach we propose, the search engine first selects a single master site for a

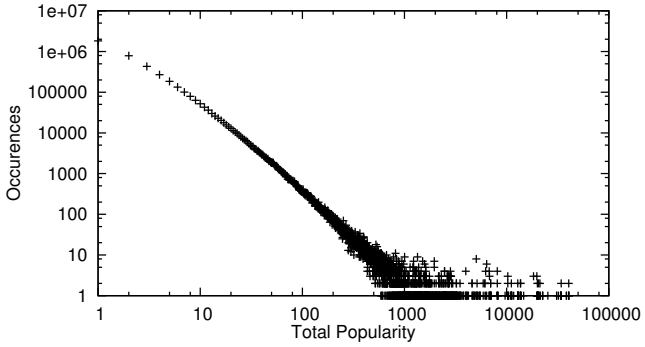

Figure 2: Document popularity distribution in the training set

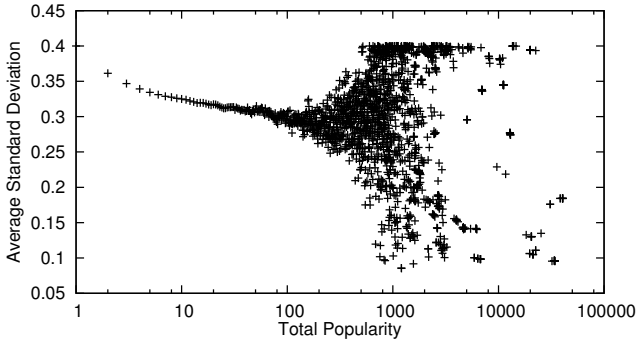

Figure 3: Divergence of document popularity distribution in the training set

document, and then replicates the documents depending on the observed popularity of the document.

As each query is labeled with the search engine site it originates from, it is also possible to study the popularity of documents with respect to the users of each site. In particular, we are interested in knowing if documents are equally popular among all the sites, or if they only of interest to users from a particular site. For each document, we compute the normalized standard deviation of its popularity among the five sites of the search engine. We average the divergence of the documents that exhibit the same total popularity and display the results in Fig. 3. Given the power law distribution of the document popularity, the right part of the graph contains a lot of noise, as the averages are computed from very few samples. Nevertheless, the results show a clear trend: popular documents present a lower deviation in their popularity among the different sites. This means that popular documents are more likely to be queried by users from different sites. Still, the average standard deviation remains high (above 0.2), even for very popular documents. This shows that, while some documents are accessed uniformly by users from all sites, many popular documents are accessed by users of a given site in particular. This observation confirms the validity of our approach, the access pattern of documents shows that they indeed have a natural master site. Furthermore, as some documents, while being very popular, remain local to one site, this also argues in favor of local replication decisions, as opposed to using global popularity statistics.

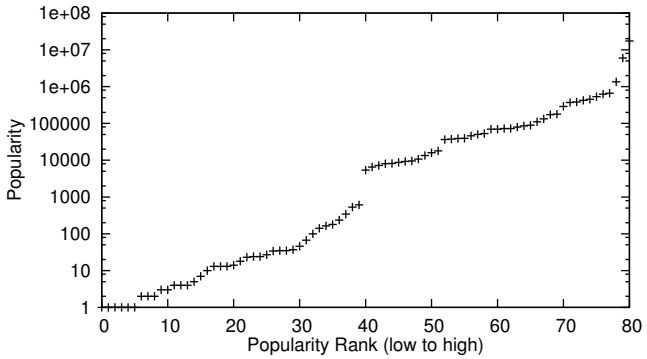

Figure 4: Popularity of the languages

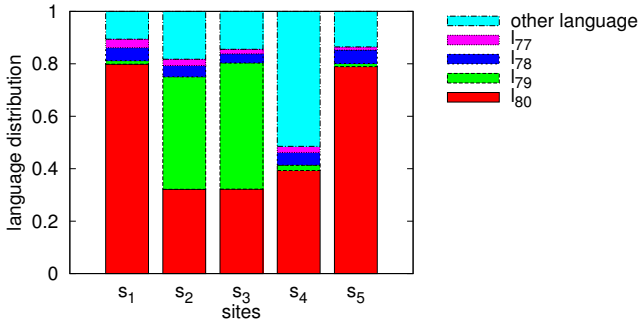

Figure 5: Distribution of the most popular languages at the different sites

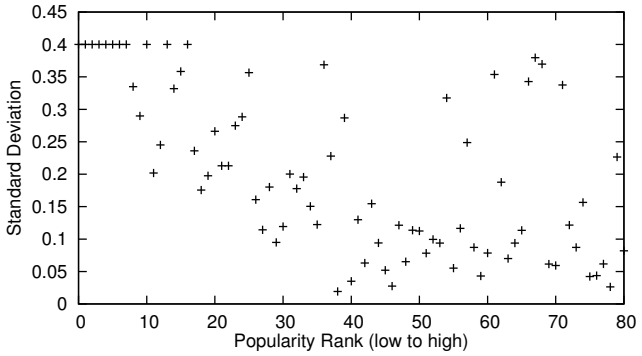

Figure 6: Divergence of the languages with respect to their popularity

5. EXPERIMENTS

5.1 Metrics and theoretical baselines

The main focus of our work is the selection of a master site for each document indexed by the search engine. We rely on existing query forwarding algorithms to process the queries (Section 2). Therefore, we do not evaluate the quality of the results returned by the search engine. As explained by Brefeld *et al.* [4], distributed search engines should achieve a good trade-off between the amount of data indexed by each site and the proportion of forwarded queries. In the approach we propose, a document is only indexed by its master site. Hence, the total size of the indexes is fixed. Our main evaluation criterion is the locality of the results delivered by a search site. The first metric we use to evaluate the quality of our results is the proportion of documents returned as results by a site that are part of its index:

$$locality(S_i) = \frac{1}{\sum_{Q_j \in \mathcal{Q}_i} |result(Q_j)|} \sum_{Q_j \in \mathcal{Q}_i} |results(Q_j) \cap \mathcal{D}_i|,$$

where $\mathcal{D}_i$ is the set of test documents assigned to S_i, $\mathcal{Q}_i$ is the set of test queries submitted to S_i, and $results(Q_j)$ is the set of documents that the search engine returns as a result to query Q_j.

This metric reflects the activity of the search engine, as each query is taken into account with the same weight. However, making an error in the assignment of a popular document has more impact, since it affects more queries. We define a second metric that takes into account each document with an equal weight. For each document, we sort the sites according to the popularity of the document at each one of them. A perfect master assignment would index the document at the first site of this list. We compute the rank of the site selected by the feature in this list and score it using NDCG [9]. We then average this value over the collection of documents.

From this information, we also compute theoretical baselines in order to facilitate the interpretation of our results: BEST always indexes a document at the site where it is most popular, WORST indexes at the site where the document is least popular, and RAND selects a site at random.

5.2 Language

We observe 81 different languages in our training data. In Fig. 4, we display the popularity of each language in the query results. The most popular language represents 58% of the results of the queries and the second one has a 20% share. This clearly demonstrates an imbalance in the popularity of the languages, i.e., some languages are dominant and widely used while others are very rare. The language classifier assigns the *unknown* label to 2.2% of the results.

We examine the distribution of languages in the results of each site in Fig. 5. The overall most popular language, l_{80}, represents a large fraction (over 30%) of the results returned by each site. However, the second most popular language, l_{79} is mostly used by the users of the sites s_2 and s_3. Over 50% of the results returned by s_4 are not in one of the most frequent languages. This suggests that users in these region tend to use multiple less common languages.

For each language, we compute the probability that the query originated from each site, as explained in Section 3.2. Fig. 6 displays the divergence of these probabilities for the five sites, while Fig. 7 compares the highest probability with the others. These results show that while some languages are clearly localized and match a particular site, many of the most popular languages have a more uniform usage. The lower the divergence of the popularity at each site, the more difficult it is to precisely assign documents to sites solely based on their language. This is problematic as popular languages are difficult to assign, and they represent most of the documents in the Internet.

5.3 Document as a query

We evaluate the performance of KL divergence in assigning documents to sites using different smoothing parameters μ. In Fig. 8, we display the performance of the assignment using the locality metric. It appears that computing KL divergence with respect to the terms in the result documents,

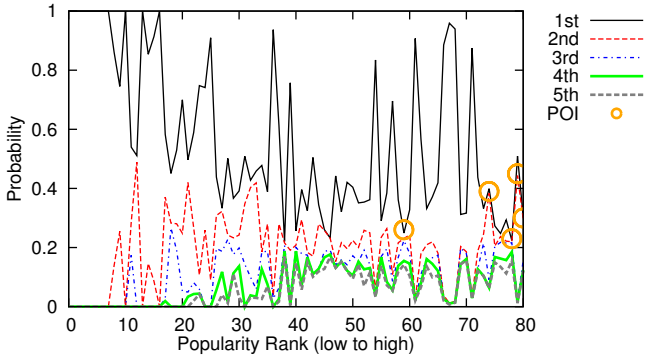

Figure 7: Probabilities of the languages with respect to their popularity

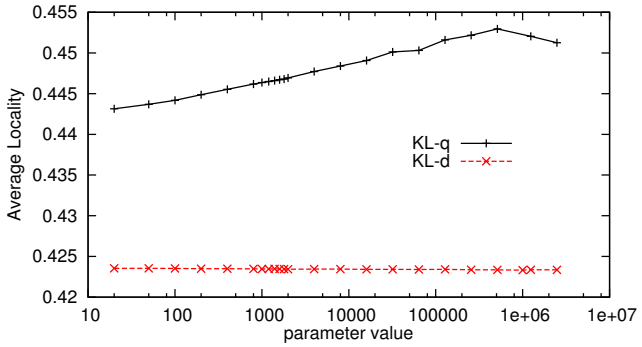

Figure 8: Influence of the μ parameter in KL divergence performance

KL-D, is not affected by the parameter value and maintains a constant performance. On the contrary, KL-Q, which takes into account the terms in the queries, exhibits the maximum performance for $\mu \approx 500{,}000$. We confirm these results by cross-validating on different subsets of the test collections. The performance measurements through NDCG lead to the same conclusions. Hence, we do not report them here. Note that when comparing the performance of KL divergence with other approaches, we only consider the best parameter.

5.4 Cache invalidation

We generate a cache of results at each site using the training queries and documents. For each query, this cache records the number of results as well as the ranking score obtained by the last result. We then simulate the addition of the test documents to the index of each site. Following the strategy of Blanco *et al.* [3], we count the number of cache entries invalidated at each location, weighted by the number of occurrences of the query. Fig. 9 depicts the distribution of the number of invalidations. These results are very similar to the distribution of document popularity and clearly indicate a power-law distribution. This result is not surprising, as the cache invalidation process is very close to the ranking process of the search engine.

We evaluate the divergence of the number of invalidations at each site and present the results in Fig. 10. Once again, this distribution resembles the divergence of the popularity of documents. The similarity between cache invalidation and query processing is encouraging: if the query distribution of the users does not vary too much, the distribution of

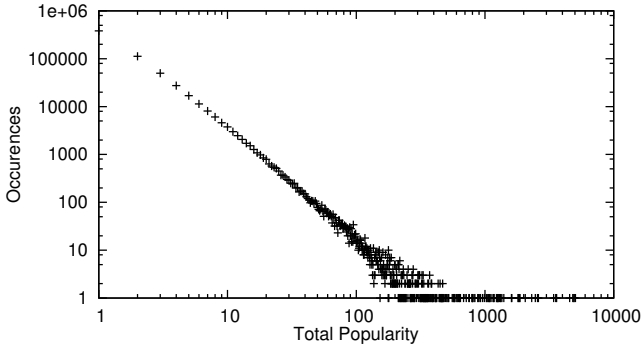

Figure 9: Distribution of the number of invalidations

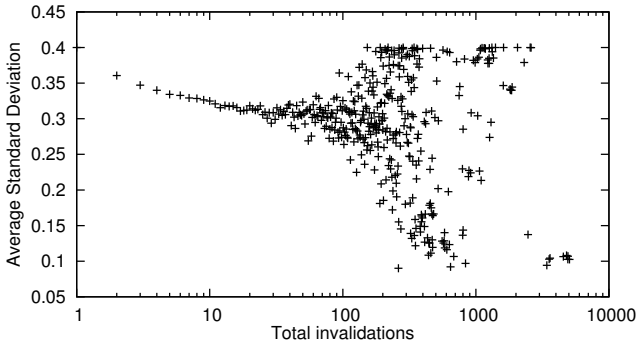

Figure 10: Divergence of cache invalidation distribution

cache invalidations should be a good indicator of the popularity of documents for future queries. However, cache invalidation suffers from the same problem as query processing: some documents have a very low popularity and are never returned in query results. Similarly, a large fraction of documents do not generate any cache invalidation. In our experiments, 648,184 documents cause cache invalidations in at least one site. This means that CACHE can be used to infer a prediction for 24% of the test documents. From these, we also need to remove the documents that do not match any query of the test set although they invalidate cache entries in the training set. This leaves us with 531,646 documents to measure the performance of CACHE.

5.5 Comparison

Table 1 (left) presents the performance of all the features on the full document test dataset. As CACHE cannot be applied on all documents, we perform a separate comparison. The first observation is that the best locality achievable is 72.9%. As the most popular documents are queried from different sites, it is not possible to obtain 100% locality without any replication. Still, this score, while impossible to achieve in practice, remains very high. A random assignment provides 20% locality, since the search engine is deployed on five sites. As few very popular documents are queried from all five sites of the search engine, the worst possible score is not 0%, but 1.2%. LANG, in spite of being a simple feature, achieves 38, 1%. KL-D and KL-Q achieve better performance, with 42, 3% and 45, 3% locality, respectively. While these three features all leverage the content of the document to compute a prediction, KL divergence is more reliable as it learns directly from the behavior of the users. In particular,

feature	All documents		Invalidation subset	
	locality	NDCG	locality	NDCG
BEST	72.9%	0.500	70.6%	0.500
WORST	1.2%	0.031	1.4%	0.031
RAND	20.0%	0.194	20.0%	0.194
LANG	38.1%	0.344	39.5%	0.342
KL-D	42.3%	0.355	46.2%	0.359
KL-Q	45.3%	0.362	48.7%	0.368
CACHE	NA	NA	52.0%	0.368

Table 1: Overall performance

as LANG mostly relies on a dictionary, it does not benefit from particular words such as names and locations. KL-Q outperforms KL-D, which indicates that mining the queries of the users is more reliable than mining the results. This may seem counterintuitive, as this metric is applied on the terms of a new document to index. Hence, it would seem more natural to compare it with terms in other documents. Nevertheless, the queries seem to contain less noise, which increases the precision of the assignment.

Table 1 (right) shows results computed on the set of documents that trigger at least one cache invalidation. These documents have higher quality and are more likely to be accessed in several sites. This explains why BEST scores are lower while all other features, except for RAND, obtain higher results. CACHE obtains the best locality score with 52%. This can be explained by the similarity of the cache invalidation process to the computation of search results.

In Figs. 11 and 12, we evaluate the locality achieved by the features with respect to the popularity of the document. In order to avoid noise, we group together different ranges of popularity. The more popular a document is, the lower BEST scores. Indeed, popular documents have a higher probability to be accessed from different locations. On the left part of the plot, we can see that while LANG has a constant behavior, KL-D, KL-Q and CACHE obtain better results on popular documents. One of the explanations could be that popular documents have a higher quality, which provides these last three features with more terms to leverage. Unpopular documents have more random access patterns and are less reliable due to the lack of samples of users accessing them. Interestingly, around a popularity of 1,000, CACHE achieves a performance very close (88%) to the optimal BEST. The right part of the plot, for very high popularity, exhibits a different behavior. All features but BEST obtain lower scores. On of the possible explanations to this phenomenon is that very popular documents are more universal and are accessed from all locations. Hence, this could be caused by a temporary event that would have reflected in the queries.

5.6 Combination

Our experiments show that CACHE generates the most accurate document placement. However, it cannot be applied on all documents, as some of them do not generate any query invalidation. One simple possibility to increase the precision of master selection overall is to rely on CACHE when this information is available, and otherwise use a more general method, such as KL-Q. This leads to a locality performance of 46.0%, which constitutes an improvement of 20.8% over the language baseline.

As explained in Section 3.3, KL-Q, KL-D and CACHE can be interpreted as probabilities and can therefore be combined with different coefficient. We evaluate this possibility over a wide range of coefficients. Nevertheless, as shown in Fig. 12, these three predictors exhibit a similar behavior and only little gain can be achieved by combining them. We obtain a maximum locality of 52.6% on the documents that generate cache invalidations by combining CACHE and KL-Q. This is unexpected, as both these metrics are based on terms in the queries. More interestingly, combining them has a higher impact on the NDCG performance. This means that combining predictors has a smoothing effect, as fewer false decisions are made.

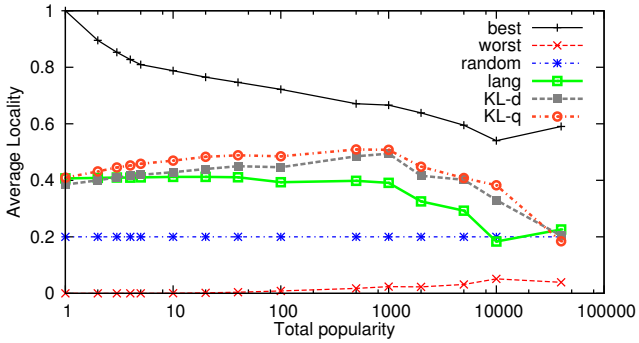

Figure 11: Performance with respect to document popularity (all documents)

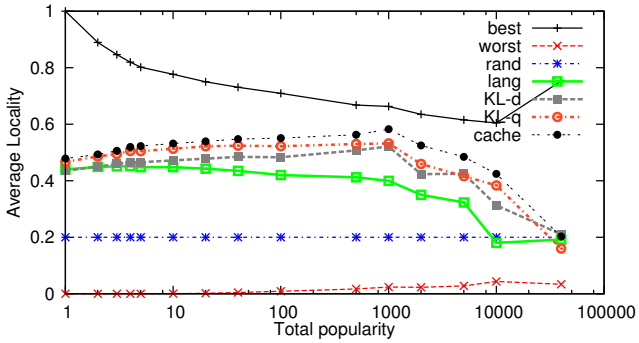

Figure 12: Performance with respect to document popularity (invalidation subset)

6. PRACTICAL CONSIDERATIONS

Throughout this paper, we consider different metrics to assign sites to new documents. We evaluate them on a large collection of data to assess their efficiency. In this section, we discuss the practical implementation aspects.

6.1 Language

The document placement policy is by far the easiest to implement. The only requirement is the presence of a classifier that assigns a document to a language depending on the terms it contains. Most of this information is static. It may be interesting to regularly enrich it with new popular terms, in particular if they differentiate users from different regions. Given its simplicity, this method achieves a reasonable precision. It can be used in contexts where there is little access to the information of the data centers, as reasonable approximations about the language statistics of regions can be obtained from demographic studies. However, if more in-

formation can be obtained from the search engine, it is more cost efficient to implement more complex measurements as mistakes in document placement cause query forwarding.

6.2 Document as a query

The computation of KL divergence requires statistics about the content of the different search sites. In this paper, we propose two sources of information: the queries of the users and the content of the results. Aggregating the terms of queries is simpler than extracting data from the documents. Indeed, the later would be a two steps process, in which the list of results is used to extract the documents and obtain their content. To reduce the cost of extracting this information, it is possible to leverage the text from the snippets computed by the search engine. This information does not contain the full document and could constitute an interesting trade-off between document information and query information. Furthermore, snippets of documents are usually cached by search engines, making their access faster.

Our experiments show that using queries to summarize the content of a search site is more efficient. In practice, each search site needs to transfer term statistics about its query log to the crawler. This data should be updated regularly to reflect changes of user interest. This is not problematic, as this data is small, typically in the order of hundreds of megabytes, and can be easily compressed.

6.3 Cache invalidation

Contrary to the other approaches, the number of cache invalidations caused by a document cannot be computed locally by a crawler. A search engine cache typically contains millions of entries, which makes its replication on the crawler impractical [3]. Consequently, the crawler has to transmit the document to each search site to determine how many invalidations this causes. Blanco *et al.* [3] propose an approach to summarize the content of a document and reduce the amount of data transmitted to the cache. Using term frequency statistics, they select the terms that have the highest impact in the ranking function of the search engine. At the cost of a few invalidations misses, this significantly reduces the size of the data representing a document. In this paper, we do not evaluate the impact of such techniques on the index partitioning. Nevertheless, this constitutes a potential optimization for the system, as it would reduce the size of the data transmitted to computed the invalidation scores.

While relying on the number of cache invalidations to estimate the relevance of a document may seem to generate more communications than the other approaches, it is important to notice that cache invalidation is a desirable mechanism in a search engine. This is particularly true if the index is updated incrementally. Therefore, if such a mechanism is already in place, it is possible to use invalidation statistics without generating any additional traffic.

Our experiments demonstrate that metrics based on cache invalidation provide the most accurate information to place documents among several search sites. The main goal of a placement strategy is to increase the locality of the computation of results. Hence, even a small improvement in locality can significantly reduce the traffic caused by query forwarding, as well as the additional processing it incurs. It is difficult to accurately evaluate these costs without implementing a real prototype. Depending on the amount of queries with respect to the number of new documents, the cache invalidation mechanism may compensate part of its bandwidth usage, or even save on network usage overall. Finally, forwarding queries also increases the latency observed by the users. Thus, any gain in document placement accuracy translates into an improvement in quality of service.

7. RELATED WORK

The inverted index partitioning problem is considered by many works in literature [10, 11, 13, 15]. The two commonly used techniques are to partition the index based on document ids or term ids. The main challenge addressed by these works is to distribute the index over a parallel computing system such that both the storage and query processing loads are evenly distributed. All these works assume that the index is distributed over a search cluster located within a single search site. Our work investigates the problem in a geographically distributed setting, where the main problem is to reduce the workload of the search system rather than establishing a load balance across search sites.

In practice, in a multi-site web search engine, partitioning of the index can be performed based on the language or the region of documents. So far, all research works investigated the very simple region-based index partitioning strategy [1, 7]. In these works, the index in each data center is built over the documents that are in the geographical neighborhood of the data center with partial replication of popular documents on all sites. Our technique goes one step beyond these works by taking into account the past query distributions observed in local search sites.

The closest work to ours is the work of Brefeld *et al.*, which proposes a machine learning model to replicate Web documents across data centers [4]. The main difference between this work and our approach is that we aim to achieve an effective one-to-one mapping between documents and sites, whereas the model in [4] replicates a document on many sites. Brefeld *et al.* rely mostly on the language and the region of a document to assign it to several sites. As shown in our experiments in Section 5.2, this can be problematic, as popular languages are used by many users in different regions. In our dataset, the most popular language constitutes over 30% of the results of each search site. An approach purely based on languages is likely to index all documents in this language in all different sites. As a consequence, a large proportion of the indexes will be used to reference low quality documents, solely on the basis of their language. We believe that it is preferable to rely on observed documents popularity before replicating it on many location. Hence, we propose a two phase approach, in which we first select a master, and then replicate the document using real popularity observations. Assigning a master site to each document ensures that it remains in the index, while giving the possibility to the other sites to replicate the document without synchronizing. Our experiments confirm that the language of a document is indeed quite accurate for assigning masters to documents. However, we go one step further by using the distribution of terms in the document to build a more accurate predictor. Finally, our work is more practical in that it does not rely on a learning model, but instead uses some simple statistical evidence about user interests.

In this paper, we rely on KL divergence [17] to evaluate the similarity between a document and the content of a site. In the early stages of our work, we also considered alternative approaches. Callan *et al.* use inference networks to select

collections of documents relevant to a given query [5]. In the context of distributed search, this avoid forwarding a query to a site that does not contain any potential result. This algorithm could also be used to determine which site would be most likely to contain a document, and therefore would be a suitable master. We performed evaluations using inference networks. The performance is quite close to the one of KL divergence. Hence, we do not report on these numbers.

8. CONCLUSION

We evaluated two different document assignment techniques for multi-site distributed Web search engines. Compared to a naive baseline, which assigns documents to sites based on documents' language, our techniques achieved a large improvement (20.8%) in increasing the locality of documents. Using cache invalidation knowledge, we achieve up to 88% of the optimal performance when documents are sufficiently popular. This directly impacts the amount of query forwarding needed to compute results, thus improves the quality of service and reduces the cost of the search engine.

The next steps involve designing a dynamic index replication scheme. This work will be facilitated by the presence of a master for each document. This setup favors dynamic policies, in which documents can be easily added and removed from the index to reflect users' interests. Depending on the replication mechanism, it may also be possible to design a master migration procedure, to transfer the responsibility of a document from the master site to a site hosting a replica. We believe that this approach is quite promising as, by adaptively re-partitioning our index, we will be able to capture the temporal patterns of document accesses.

Acknowledgement

This work has been partially supported by the COAST project (ICT-248036), funded by the European Community.

9. REFERENCES

[1] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Plachouras, and L. Telloli. On the feasibility of multi-site web search engines. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 425–434, 2009.

[2] L. A. Barroso, J. Dean, and U. Hölzle. Web search for a planet: the Google cluster architecture. *IEEE Micro*, 23(2):22–28, 2003.

[3] R. Blanco, E. Bortnikov, F. P. Junqueira, R. Lempel, L. Telloli, and H. Zaragoza. Caching search engine results over incremental indices. *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 82–89, 2010.

[4] U. Brefeld, B. B. Cambazoglu, and F. P. Junqueira. Document assignment in multi-site search engines. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 575–584, 2011.

[5] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, 1995.

[6] B. B. Cambazoglu, V. Plachouras, and R. Baeza-Yates. Quantifying performance and quality gains in distributed web search engines. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 411–418, 2009.

[7] B. B. Cambazoglu, E. Varol, E. Kayaaslan, C. Aykanat, and R. Baeza-Yates. Query forwarding in geographically distributed search engines. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 90–97, 2010.

[8] W. Gao, H. C. Lee, and Y. Miao. Geographically focused collaborative crawling. In *Proceedings of the 15th International Conference on World Wide Web*, pages 287–296, 2006.

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

[10] C. Lucchese, S. Orlando, R. Perego, and F. Silvestri. Mining query logs to optimize index partitioning in parallel web search engines. In *Proceedings of the 2nd International Conference on Scalable Information Systems*, pages 1–9, 2007.

[11] A. MacFarlane, J. A. McCann, and S. E. Robertson. Parallel search using partitioned inverted files. In *Proceedings of the 7th International Symposium on String Processing Information Retrieval*, pages 209–220, 2000.

[12] B. Ribeiro-Neto, E. S. Moura, M. S. Neubert, and N. Ziviani. Efficient distributed algorithms to build inverted files. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 105–112, 1999.

[13] B. A. Ribeiro-Neto and R. A. Barbosa. Query performance for tightly coupled distributed digital libraries. In *Proceedings of the 3rd ACM Conference on Digital Libraries*, pages 182–190, 1998.

[14] B. A. Ribeiro-Neto, J. P. Kitajima, G. Navarro, C. R. G. Sant'Ana, and N. Ziviani. Parallel generation of inverted files for distributed text collections. In *Proceedings of the 18th International Conference of the Chilean Computer Science Society*, pages 149–157, 1998.

[15] A. Tomasic and H. Garcia-Molina. Query processing and inverted indices in shared: nothing text document information retrieval systems. *The VLDB Journal*, 2(3):243–276, 1993.

[16] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 403–410, New York, NY, USA, 2001. ACM.

[17] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22:179–214, April 2004.