Document Assignment in Multi-site Search Engines

Ulf Brefeld
Yahoo! Research
Barcelona, Spain
brefeld@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

ABSTRACT

Assigning documents accurately to sites is critical for the performance of multi-site Web search engines. In such settings, sites crawl only documents they index and forward queries to obtain best-matching documents from other sites. Inaccurate assignments may lead to inefficiencies when crawling Web pages or processing user queries. In this work, we propose a machine-learned document assignment strategy that uses the locality of document views in search results to decide upon assignments. We evaluate the performance of our strategy using various document features extracted from a large Web collection. Our experimental setup uses query logs from a number of search front-ends spread across different geographic locations and uses these logs to learn the document access patterns. We compare our technique against baselines such as region- and language-based document assignment and observe that our technique achieves substantial performance improvements with respect to recall. With our technique, we are able to obtain a small query forwarding rate (0.04) requiring roughly 45% less replication of documents compared to replicating all documents across all sites.

Categories and Subject Descriptors

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

General Terms

Design, Performance, Experimentation

Keywords

Multi-site web search engines, document replication, classification

1. INTRODUCTION

As the Web grows in size and extent, web search engines require an increasing amount of compute power to keep up with user expectations. More compute power is necessary, for example, to crawl more documents, compute more expensive features, and serve more users faster. Recently, commercial engines have dealt with this problem by building larger data centers, and centralizing all search engine infrastructure into one of those data centers. There is a clear limitation in this method, however, since building even larger data centers becomes increasingly difficult due to space and power pressure.

An appealing alternative to such a centralized approach to search is to split the functionality of an engine across multiple data centers or *sites*. Different from *federated search* [8], this design option implies complete control over the functionality of each site, instead of using existing engines to process queries. To date, some commercial engines have taken the approach of using multiple sites in a replicated fashion to prevent local disasters from disrupting the search service. This replicated approach consists of having multiple data centers crawling and indexing approximately the same collection of web documents. Such a replicated solution, however, does not mitigate the problem of having to build even larger data centers in the future. Consequently, a successful multi-site design depends upon the ability to split the functionality of an engine without requiring sites to be full replicas of each other.

Being able to divide roles and tasks across a number of sites efficiently has a number of advantages. An important advantage is scalability: it enables a new data center to become part of the system at any time and to increase its overall capacity. It also enables other benefits such as geodiversity, which leads to lower costs [1] and more efficient edge processing [14]. Despite such a set of appealing benefits, no such practical solution exists, to the best of our knowledge, and there are still several challenges to overcome before we are to see such multi-site search engines commonly in practice.

One such challenge is assigning documents to sites effectively. Since we have chosen by design not to have all sites crawling and indexing the same set of documents, we need a mechanism to determine which site crawls and indexes which set of documents, perhaps even allowing some degree of replication for performance. Assigning documents to sites, however, is not a straightforward task. There is no clear set of features that provides a good assignment, and a poor assignment might lead to poor utilization of storage, compute, and communication resources, possibly defeating the purpose of having a multi-site engine.

Our goal therefore is to propose a solution with desirable properties to the problem of assigning documents to sites in multi-site engines. We assume that sites crawl pages regularly from the Web, and once they fetch a new page they execute a procedure to determine which site should index such a page, possibly one or more remote sites. Our strategy builds upon multi-class classification techniques, and shows that it is possible to improve the overall user

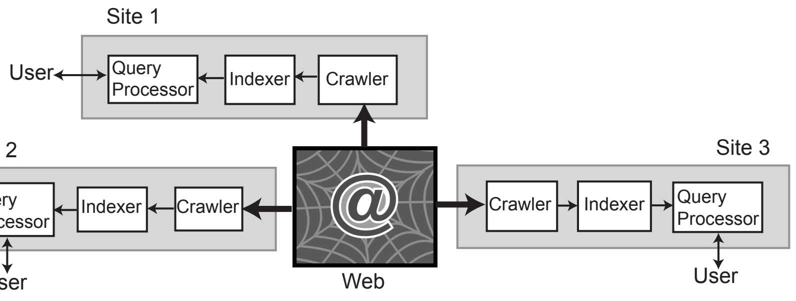


Figure 1: System overview, using three sites.

experience compared to baseline strategies. There are two main use cases we foresee for our strategy:

Query processing and indexing: Suppose a multi-site search engine such that users always submit queries to the closest site with respect to network distance. If such a site indexes most documents requested by its users, then we are able to reduce query processing latency substantially on average. Reducing query processing latency is critical for user satisfaction [28].

Crawling: Suppose that we have a procedure that is able to decide based on the URL which site or set of sites should index this document. In such a scenario, sites mostly fetch documents they index, and for optimization purposes, sites can tell each other URLs they discover and other sites should index.

Contributions.

We propose a machine learning technique for document assignment in multi-site search engines. We use query logs, documents, and region/language classifiers from a commercial search engine to evaluate our technique. Our main conclusion is that this technique enables significant performance improvements with respect to user-observed latency and significant improvements with respect to system resources such as storage, processing, and network bandwidth. Compared to baselines that classify documents based on region and language only, we are able to reduce substantially the rate of query forwarding (from 0.65 to 0.04) without requiring that all sites index all documents (roughly 45% less replication).

Roadmap.

The rest of this paper is structured as follows. Section 2 describes our problem setting in greater detail. Section 3 characterizes the data and discusses the features we use for the machine learning approaches that are presented in Section 4. We present our empirical results in Section 5, and Section 6 reviews related work. We discuss our findings in Section 7, and Section 8 concludes the paper.

2. PROBLEM DESCRIPTION

A search engine has essentially three main components: a crawler, an indexer, and a query processor. The crawler fetches documents from the Web, and in the process it finds new pages to crawl by parsing fetched pages. Once the crawler builds a document collection, the indexer processes the document collection and creates an inverted file, which is used to process queries. The query processor receives queries from users and processes them using the inverted file. All these components can be implemented with multiple processors in a parallel fashion, and we assume in our architecture that each site contains processors implementing crawling, indexing and query processing.

In our setting, we need a document assignment mechanism for two reasons: crawling and indexing. A crawler implemented with multiple processors requires a way to split the crawling work across all participating processes. In a multi-site scenario, splitting the work across processors becomes even more difficult because not all processors are co-located and selecting processors closer to the document is critical for efficient utilization of network bandwidth [12]. Consequently, we need a mechanism to determine which site or set of sites should crawl a given document. Given such an assignment mechanism, there are some trivial strategies that could be implemented, such as having a single site fetching documents and distributing synopses to others, assuming a replicated document.

Since all sites index documents, it is necessary for each site to determine which set of documents it should index. Ideally, a site indexes all documents necessary to process queries it receives directly from users. In practice, this is simpler to achieve when fully replicating documents. The major drawback of fully replicating is the potential increase on the amount of infrastructure (e.g., hardware, power, personnel) required for each site. Consequently, we assume that sites are not necessarily approximate images and are able to forward queries to each other in the case they are not able to process queries locally. Sites can use, for example, thresholding techniques such as those proposed in [1] and [13] to forward queries. It is desirable, however, to reduce the amount of forwarded queries, due to both network utilization and query processing latency. Hence, we target an assignment mechanism that enables us to reduce the amount of forwarded queries between sites. The actual query forwarding mechanism is out of the scope of this paper.

More specifically, the problem we target is the one of processing a document and deciding which subset of sites should index it. Let $\mathcal{S}$ be the set of sites of our system and $\mathcal{D}$ be the set of documents on the Web. Note that the set $\mathcal{D}$ is not necessarily known. We seek a function $g : \mathcal{D} \rightarrow 2^{\mathcal{S}}$ that is able to assign documents accurately and efficiently. Accurately means that for every site S, a significant fraction of the queries received in S are processed locally in S and not forwarded. Efficiently means that it is not computationally costly to find which sites should crawl and index a given document. In the remainder, we will focus on an application with 27 sites.

3. FEATURE EXTRACTION

We take a machine learned document classification approach as solution for our document assignment problem. We first produce a list of features that may correlate well with the occurrence counts of documents in search results. For every document in a given training set, we compute the values for each of the features in our list. From the same training set, we also compute the view information (the number of times a document is displayed to users) for documents. This information constitutes our ground truth.

In our solution, each document that appears in the top k results of a user query issued to a search site forms a classification instance, and it is labeled with the search site at which the document is displayed. Since a document may be requested by different queries issued to different search sites, a document may have multiple class labels. Given the features and class labels of training documents, we train classifiers using machine learning techniques detailed in Section 4. Then, using these models, the labels are assigned to unseen documents in a given test set of documents. A document is assigned to all sites corresponding to the predicted labels.

In our work, we evaluate two main features: document region and language, which are expected to be highly correlated with document access patterns in a geographically distributed query processing setting. We also evaluate a number of supplementary features, relatively much less correlated. In what follows, we describe the representative features extracted. Note that an alternative to our set of features would be a standard approach to document classification using a bag-of-words or tf-idf representation. We refrain from this approach as it is computationally expensive and does not translate to our use cases as the document content may not be available at all (e.g., classification of URLs discovered during web crawling).

3.1 Main Features

Region. We represent the geographical region of the web site from which the document is crawled by the `F-region` feature. It may not always be possible to accurately obtain this information, however. In our case, we use a document classifier, actively used by a commercial search engine to predict the geographical regions of documents. This classifier combines various features such as the URL domain to assign a region to every document. This feature is expected to be an important feature as it is known that a high fraction of user queries are regional queries that seek documents that are in geographical neighborhood of users [2]. In our data, the `F-region` feature can take 238 different values.

Language. A traditional technique in document classification is to use the bag of words in the document content. However, in a practical setting, this approach may be computationally expensive. Herein, we use a single content-based feature, `F-language`, which represents the language of the document. To determine the language of our documents, we use another text classifier, which is in production in a commercial search engine. This feature is also expected to be important as queries are much more likely to match documents in the same language. In our data, the `F-language` feature can take 83 different values, i.e., we observed 83 different languages in our document collection.

3.2 Supplementary Features

Document quality. The quality of a document may be computed in many different ways, using the information extracted from its content (e.g., spam classification) or by using the link information in the web graph (e.g., PageRank). Herein, we evaluate two quality metrics referred to as `F-linkQuality` and `F-hostQuality`. The first is a metric that uses the incoming links of the document to compute a quality value. The second computes the quality based on the host of the document. Both metrics are proprietary, and hence we do not disclose the exact formulas herein.

URL. In certain cases, the only information available about the document is the URL. There are several reasons for this. First, the document content may not have been crawled yet, but only a link to the document (i.e., its URL) is discovered. Second, classifiers may not be available to compute more complex features such as region, language, and quality. Third, the document may have an empty content and consist of only a URL. As representative features, we extract four features from the URL: `F-length`, `F-port`, `F-query`, and `F-depth`. The `F-length` feature is simply the length of the URL. `F-port` represents the port number of the HTTP server from which the document is fetched. `F-query` is a categorical feature taking the values of 1 or 0, depending on whether the URL has a query component[1] or not. Finally, `F-depth` indicates the depth of the document in the storage hierarchy, i.e., it is simply the number of slashes in the path component of the URL.

Size. We use two features, `F-htmlSize` and `F-textSize`, which represent the size of the document before and after removing the HTML tags, respectively. We also use two other features, referred to as `F-termCount` and `F-uTermCount`, which represent the number of terms and the number of unique terms in the document, respectively.

3.3 Feature Characteristics

In this section, we first investigate the correlation between our main features and document views in search results. We adopt the following strategy. For the `F-region` and `F-language` features, we count the number of documents having a certain feature value and how many of those documents are requested by at least one query, i.e., appear in the top k search results (in our case, $k = 10$). The ratio of the two numbers gives us the likelihood for those documents to be viewed at least once. Any feature value for which the number of documents is less than 0.1% of the collection size is omitted as the number of samples is low.

Fig. 2 shows properties of the `F-region` and `F-language` features, where the values are sorted in increasing order of probabilities. According to Fig. 2 (top), documents of certain regions are more likely to be viewed. We similarly observe that documents of certain languages are more likely to be viewed (Fig. 2, bottom).

In Fig. 3, we visualize the correlation between sites (x-axis) and features (y-axis). In this figure, correlation is encoded by color, that is, darker colors indicate higher correlations and lighter colors indicate lower correlations, e.g., white indicates statistically uncorrelated features. Unsurprisingly, we observe that `F-region` and `F-language` are the two features that correlate most with all sites. Depending on the site, the strength of the correlation varies. For instance, sites 4, 13, 15, and 27 highly correlate with regions and languages while sites 6, 7, 8, 19, and 26 show only weak correlations across all features. Most importantly, however, none of the supplementary features show high correlation throughout the analysis. Consequently, we discard them in the remainder and focus only on `F-region` and `F-language` features.

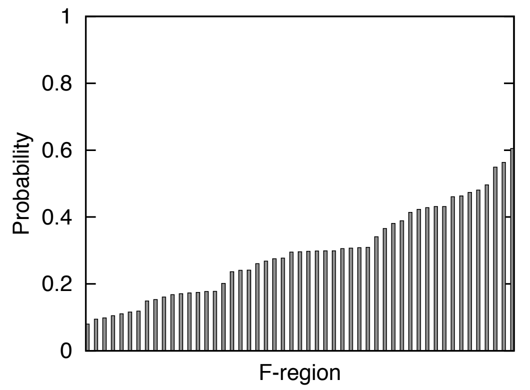
4. MACHINE LEARNING APPROACHES

In this section, we introduce machine learning approaches to learn a function $g : \mathcal{D} \rightarrow 2^{\mathcal{S}}$ that assigns every document to a subset $\mathcal{S}$ of available sites. Section 4.1 introduces the formal problem setting and shows the appropriateness of binary classification methods. Section 4.2 discusses the representation of documents.

4.1 Formal Problem Setting

Let x_i denote the feature representation of the i-th document and $y_i \in \{+1, -1\}$ be the ground-truth of a document for site s, where

[1]RFC 3986 specifies that the query component of an URI is the part between the ? and the end of the URI or the character #.



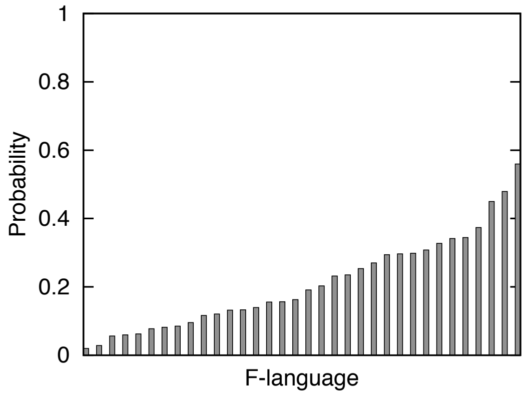


Figure 2: Probability that a document is requested by at least one query vs. the feature value: `F-region` (top) and `F-language` (bottom).

$y_i = +1$ if x_i appears in the top-k result set of queries that come from site s. That is, y_i indicates whether to store document x_i at site s ($y_i = +1$) or not ($y_i = -1$).[2]

Translating all documents and target values in this way gives us a training set of n pairs $\mathcal{D}_s = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ for site s. Recall that for a site s, the decision of storing a document is independent of possibly existing replicas of that document at sites $s' \neq s$. That is, for each site $s \in \mathcal{S}$, the formal problem setting reduces to a binary classification problem as follows.

For each site s, we aim at learning a linear model of the form

$$f_s(x) = \langle w_s, x \rangle + b_s,$$

where w_s is called the weight vector and b_s the threshold. At application time, new documents are stored at site s if $\operatorname{sign}[f_s(x)] = +1$ and rejected if $\operatorname{sign}[f_s(x)] = -1$.

For each site, optimal parameters are supposed to minimize latency *and* storage overhead. Latency can be identified with the number of *false negatives*, that is, documents that should be stored at the site but have been rejected:

$$\Omega_{\text{latency}}(f_s, \mathcal{D}_s) = \sum_{(x,y)\in\mathcal{D}_s} I\left[y = +1 \wedge \operatorname{sign}[f_s(x)] = -1\right],$$

where $I[\cdot]$ is an indicator function yielding $I[z] = 1$ if z is *true* and $I[z] = 0$ if z is *false*. Analogously, storage overhead can be related to the number of *false positives*, that is, documents that are unnecessarily stored because they are never requested:

$$\Omega_{\text{storage}}(f_s, \mathcal{D}_s) = \sum_{(x,y)\in\mathcal{D}_s} I\left[y = -1 \wedge \operatorname{sign}[f_s(x)] = +1\right].$$

Since both, latency and storage capacity, are equally important, a natural approach to finding f_s is to minimize the unweighted sum which is precisely the classification error:

$$\begin{aligned} f_s^* &= \operatorname*{argmin}_f \quad \Omega_{\text{latency}}(f, \mathcal{D}_s) + \Omega_{\text{storage}}(f, \mathcal{D}_s) \\ &= \operatorname*{argmin}_f \sum_{(x,y)\in\mathcal{D}_s} I\left[y \neq \operatorname{sign}[f(x)]\right]. \end{aligned}$$

That is, we can apply off-the-shelf machine learning techniques minimizing the error rate, such as Decision Trees [23], Logistic Regression [19], or Support Vector Machines [15], to learn classifiers for the sites $s \in \mathcal{S}$. In this work, we focus on Support Vector Machines that minimize an upper bound on the error rate while maximizing the separating margin between the two classes to preserve a well-posed, convex optimization problem with unique solution.

Once optimal functions $f_1^*, \ldots, f_{|\mathcal{S}|}^*$ are computed, the final assignment g is easily assembled by stacking the classifiers:

$$g(x) = (f_1^*(x), \ldots, f_{|\mathcal{S}|}^*(x)).$$

[2]To avoid cluttering the notation unnecessarily, we omit the additional subscript s for $y_{s,i}$ since the binarized ground-truth highly depends on the actual site s.

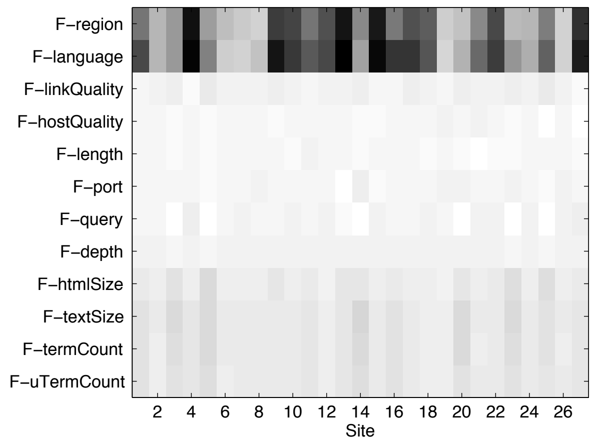


Figure 3: Visualization of correlation coefficients between features and sites. Darker colors indicate stronger correlations.

4.2 Document Representation

So far, x_i denoted the feature representation of the i-th document. Our study relies on only two categorical attributes with 238 (`F-region`) and 83 (`F-language`) realizations. Usually, categorical attributes are binary encoded such that ones indicate the presence and zeros the absence of a certain realization. For example, an attribute a having three possible realizations $a \in \{r_1, r_2, r_3\}$ would be translated into $(1,0,0)$, $(0,1,0)$, or $(0,0,1)$, respectively, depending on whether the actual realization is r_1, r_2, or r_3. Applying this principle to our attributes yields $238{+}83 = 321$ dimensional feature vectors for each document. These vectors can be efficiently stored in sparse representation as only two elements will be non-zero.

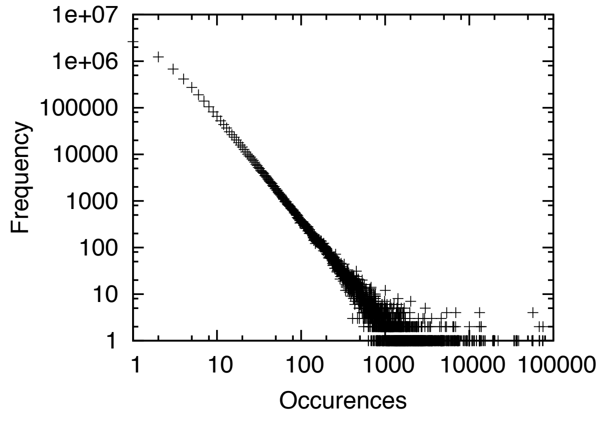


Figure 4: Document frequency distribution.

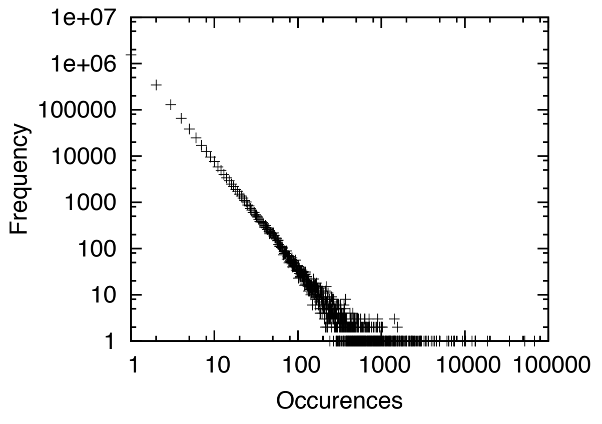


Figure 5: Query frequency distribution.

Given the binarized representation, the learning algorithm adapts to the present combinations of regions and languages. However, depending on the actual realizations, the two features might not be expressive enough to allow for reliable results. For instance, scenarios like X-OR, which cannot be learned with linear models, might occur. To allow for a richer set of features, we therefore incorporate features indicating the *absence* of certain regions or languages for a document. These so-called *negative features* have been proven beneficial in many areas including robotics [16] and natural language processing [6].

Translated to our example, a document with the attribute $a = r_2$ would have the representation $(0, 1, 0, 1, 0, 1)$ where the last three elements indicate the absence of r_1 and r_3. We devise a second representation of the documents including negative features leading to 642 dimensional feature vectors with exactly 321 non-negative elements per document.

5. EVALUATION

5.1 Experimental Setup

5.1.1 Documents

We use a set of documents crawled from the Web as our document collection (about 25 million documents). This collection excludes documents that contain text in languages such as Chinese, Japanese, and Korean (we use the UTF code range of characters to detect such documents). The reason behind excluding these documents is that there is no tokenization algorithm available to us for these languages, for which space is not a natural word delimiter.

The obtained collection is split into two sets randomly as training and test sets, each containing about 80% and 20% of the documents, respectively. A power-law distribution is observed in the frequency of document views (Fig. 4), i.e., many documents are rarely viewed while few documents are viewed many times.

5.1.2 Queries

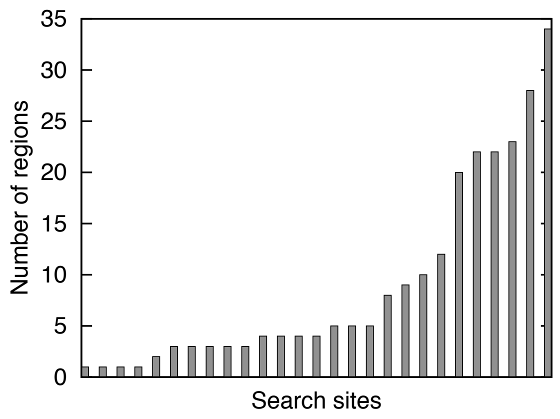
To create the ground-truth class labels, we use a training query set of 5.9 million queries, randomly sampled from the query logs (the same day) of a commercial search engine. We also sample a test query set of 5.7 million queries from a different day of query logs to assign class labels to the test collection. For each query, we record the site the query originates from (for our query log, there are 27 possible sites). During query sampling, we exclude successive page requests and consider only the queries requesting the first page, i.e., the top 10 results. All queries are passed through standard cleansing procedures such as stop-word elimination and case-folding. We also unique query terms and sort them alphabetically. Fig. 5 shows that our sample query log follows a clear power-law distribution in terms of query occurrence frequency.

We use the Terrier search engine to create two separate inverted indexes, one for the training collection and the other for the test collection. Using the same search engine, we evaluate every test and train query on their respective indexes and obtain the top 10 results. Every document retrieved in top 10 results of a query is assigned a label corresponding to the region of the query. As some documents are requested by different queries from different regions, they are assigned multiple labels. In top k computations, we use a BM25 variant. We avoid the use of techniques that incorporate more information into the scoring function, such as a link analysis metric or region boosting. This is because some features (e.g., `F-region`, `F-linkQuality`, and `F-hostQuality`) show correlation with these scoring techniques. We want to test features that are as independent as possible from the scoring function.

5.1.3 Baselines

We employ two baselines, based on region and language attributes. The first baseline, referred to as `REG`, assigns documents to sites according to their region. That is, we assume that all documents in a region are stored in the geographically closest data center. We identify the closest site by measuring the great-circle distances between the document's region and the regions in which a data center is located. In the second baseline, referred to as `LAN`, we use the predicted language of the document for assignment. In this approach, we use the information provided on Wikipedia about the usage frequency of languages in different regions. We first try to assign the document to the region with the highest number of native speakers. If there is no data center in that region, we iterate in order of decreasing frequency until we find a region with a data center. If none is found, we assign the document to the geographically closest data center.

Fig. 6 shows how many different regions and languages are assigned to a site. According to Fig. 6 (top), a few search sites are assigned to many regions while the bulk is assigned to only a few. The distribution is even more biased for the language baseline (Fig. 6, bottom). Note that these baselines are completely blind to document views and accesses by users of different search sites in reality.



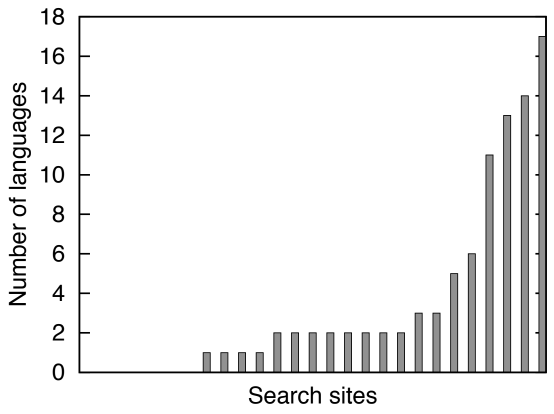


Figure 6: The number of different regions (top) and languages (bottom) assigned to each search site by the baseline techniques.

Table 1: Accuracy values for individual search sites

		Accuracy			
Class	Size	REG	LAN	SVM	SVM-n
18	21674	99.86	99.92	57.35	57.35
25	23011	99.87	99.92	57.58	57.58
6	24972	99.82	99.91	57.83	57.83
5	46863	99.51	99.83	32.04	60.21
20	51719	97.52	98.94	58.09	58.09
7	54814	99.74	99.80	60.08	60.08
11	56087	99.74	99.78	32.21	59.14
1	56210	98.99	99.80	59.82	0.20
23	60826	99.59	99.49	38.91	60.43
17	94259	99.61	99.36	29.23	62.05
9	98347	99.62	99.68	59.19	0.36
21	144831	99.41	99.44	0.55	64.23
16	184054	98.32	99.19	33.51	65.46
26	738741	98.45	98.67	16.12	2.86
10	874524	94.77	93.95	15.94	81.06
0	1016069	95.93	95.27	20.56	4.91
19	1034849	94.31	94.94	32.71	4.96
8	1191769	93.63	95.36	5.89	5.89
14	1205321	95.54	95.92	5.69	5.69
2	1676430	91.09	89.91	9.68	9.68
22	2319567	88.14	86.88	13.05	13.15
13	2397392	89.08	87.73	12.30	12.30
3	2822912	92.54	93.28	14.10	14.10
15	2929832	86.39	84.59	15.40	15.40
4	3635510	73.05	76.56	23.43	23.43
12	4115087	86.40	87.67	22.71	22.71
24	7084813	66.74	76.75	44.29	44.29
Avg. accuracy		95.29	95.734	30.68	36.41

5.2 Results

5.2.1 *Accuracy*

In the first part of our evaluation, we focus on the predictive accuracies of Support Vector Machines (SVMs) using the two feature representations described in Section 4.2. We refer to using only positive features simply as `SVM` and denote the incorporation of negative features as `SVM-n`.

For each site and feature representation, we use 5-fold cross-validation [33] for selecting the optimal value of the SVM trade-off parameter $C \in [10^{-4}, 10^{4}]$. Using the optimal C, the model is retrained on all training examples and evaluated on the independent test set. We account for skewed class distributions by re-weighting the impact of examples according to their class ratios throughout all experiments.

A problem in measuring the accuracy for a site s is differently labeled instances of the same document. Since all these instances have the same feature representation, they are classified into the same category. However, irrespectively of whether the prediction is correct or not, some of these instances contribute positively to the accuracy and some negatively. We intend to exclude the possibility that negatively labeled instances overrule the impact of the positively labeled ones since they refer to other sites and induce a bias on the accuracies. A simple remedy is to re-label the ground-truth of all instances of a document as positive if at least one of them is assigned to the site s. However, note that this approach significantly alters the distribution of documents for the sites. We thus decided to remove all negatively labeled instances of a document from the test set of site s if at least one of these instances is positively labeled. If all instances of a document are positively/negatively labeled, then all instances are kept.

Table 1 shows the accuracies for all 27 sites, ordered with by their size, i.e., the number of instances in the class. The two baselines perform excellent in terms of accuracy and significantly outperform the machine learning approaches. For the latter, the size of the sites is crucial for predictive performance. While small sites are covered reasonably well the SVMs perform poorly for large ones. The reason is the great variety of documents at large sites. That is, documents of any region and language are retrieved at large sites and the presence of all realizations of the attributes hardens the learning process. Hence, the optimal solution for large sites is to simply assign all documents to the site, which explains the poor accuracies. By contrast, the baselines assign every document by design to only a single site. Thus, if according to the ground-truth, a document should be stored on x sites, the number of true negatives is at least $tn = 27 - x - 1$, which constitutes the main portion of the accuracies in our setting. We will see in the next section that accuracy is not always a good indicator for replication and forwarding rates. Before, let us note that, compared to `SVM`, the negative features in `SVM-n` help slightly to capture small- and medium-sized sites better. For large sites, the performance is identical.

5.2.2 *Replication and Forwarding Rates*

When designing multi-site search engines, there are two important concerns with respect to resource utilization and user satisfaction. Having more documents in a site implies more storage for

Table 2: Replication and query forwarding rates for individual sites

		Replication rate				Forwarding rate			
Class	Size	REG	LAN	SVM	SVM-n	REG	LAN	SVM	SVM-n
18	21674	0.001	0.000	0.000	0.000	0.768	0.771	0.771	0.771
25	23011	0.001	0.000	0.000	0.000	0.772	0.777	0.777	0.777
6	24972	0.001	0.000	0.053	0.000	0.742	0.761	0.714	0.761
5	46863	0.005	0.000	0.714	0.000	0.746	0.779	0.410	0.779
20	51719	0.032	0.024	0.019	0.000	0.323	0.281	0.287	0.487
7	54814	0.001	0.000	0.064	0.000	0.749	0.756	0.676	0.756
11	56087	0.004	0.003	0.714	0.000	0.633	0.622	0.354	0.787
1	56210	0.014	0.000	0.049	1.000	0.734	0.765	0.572	0.000
23	60826	0.004	0.005	0.638	0.000	0.350	0.343	0.228	0.366
17	94259	0.005	0.013	0.783	0.000	0.650	0.622	0.267	0.765
9	98347	0.005	0.004	0.511	1.000	0.622	0.608	0.501	0.000
21	144831	0.009	0.000	1.000	0.000	0.613	0.762	0.000	0.762
16	184054	0.023	0.014	0.706	0.000	0.546	0.548	0.280	0.756
26	738741	0.008	0.010	0.887	1.000	0.632	0.592	0.063	0.000
10	874524	0.053	0.083	0.888	0.000	0.764	0.637	0.163	0.823
0	1016069	0.013	0.000	0.864	0.995	0.737	0.788	0.160	0.004
19	1034849	0.015	0.005	0.828	1.000	0.858	0.863	0.178	0.000
8	1191769	0.075	0.060	1.000	1.000	0.650	0.568	0.000	0.000
14	1205321	0.042	0.045	1.000	1.000	0.686	0.626	0.000	0.000
2	1676430	0.020	0.008	1.000	1.000	0.838	0.872	0.000	0.000
22	2319567	0.007	0.003	1.000	1.000	0.860	0.881	0.000	0.000
13	2397392	0.019	0.001	1.000	1.000	0.769	0.795	0.000	0.000
3	2822912	0.032	0.031	1.000	1.000	0.561	0.486	0.000	0.000
15	2929832	0.007	0.000	1.000	1.000	0.784	0.812	0.000	0.000
4	3635510	0.470	0.000	1.000	1.000	0.732	0.878	0.000	0.000
12	4115087	0.055	0.063	1.000	1.000	0.688	0.585	0.000	0.000
24	7084813	0.079	0.627	1.000	1.000	0.823	0.377	0.000	0.000
Avg. replication factor		1.000	1.000	18.688	14.996	–	–	–	–
Avg. forwarding rate		–	–	–	–	0.741	0.651	0.027	0.041

those documents and processing capacity to index them. Reducing the number of documents manipulated in each site is consequently a critical goal. Reducing the amount of compute and storage resources for each site, however, is not the only concerning aspect. Using fewer hardware resources does not guarantee that the user perception of latency is good. To achieve this goal, sites ideally index all documents in the result set of queries it receives from local users without indexing the whole collection served by the engine.

In this section, we report on the actual replication and forwarding rates of the derived classifiers. The replication rate on a particular site is defined as the fraction of the full index that is replicated on that site, i.e., the rate is given by the number of postings that appear in the site's local index. Therefore, we accurately measure the storage requirement for local indexes. The replication factors we report in this section indicate the number of sites a document is replicated on average.

The query forwarding rate of a site is defined as the fraction of queries that cannot be locally processed by the site and hence needs to be forwarded to at least one other site. Therefore, we can have an estimate of what fraction of queries are expensive, i.e., have high response times. When computing the query forwarding rates, we assume the existence of an oracle that knows the placement of all document replicas and hence gives the forwarding decisions with perfect accuracy. That is, we assume that a query is processed locally only if all documents that will appear in the its top k list are indexed by the local index; otherwise, it is forwarded from the local site to one or more sites. The average forwarding rates displayed at the bottom of the table are computed over all queries submitted to all sites. We note that this average is obtained by micro-averaging, i.e., it is an average over all queries issued to the search engine.

Table 2 shows the results. Due to the nature of the baselines, their replication factor is trivially 1. Each document is stored at exactly one site, determined by either its region or language attribute. As a consequence, the forwarding rate is very high: across all queries, on average about 74% of the queries using the region and 65% using the language are forwarded.

The results change substantially when using `SVM` and `SVM-n`. For the machine learning approaches, the average degree of replication of a document goes up to almost 19 and 15, respectively, implying that the overall storage cost also increases by the same number. This is due to storing all documents for large sites as seen in the previous section. Interestingly, the fraction of forwarded queries is reduced by the higher degree of replication. For the SVM, the average forwarding rate is only 0.027, which can be considered marginal. Introducing negative features in `SVM-n` results in a slightly larger forwarding rate of 0.041 but reduces the average replication to only 15 sites. Therefore, there is a clear benefit in using `SVM` and `SVM-n`.

Irrespective of the method, the results for small sites leave room for improvement. Language and region baselines do not assign documents or only very few documents to such sites, while the machine learning approaches assign a larger fraction of documents to the sites and still requires a relatively large fraction of forwarded queries. `SVM-n` even results in an all-or-nothing strategy by either

Table 3: Replication factor for missing test instances

		Replication rate			
Class	Size	REG	LAN	SVM	SVM-n
18	21674	0.000	0.000	0.000	0.000
25	23011	0.000	0.000	0.000	0.000
6	24972	0.000	0.028	0.000	0.000
5	46863	0.005	0.006	0.000	0.000
20	51719	0.147	0.000	0.141	0.000
7	54814	0.000	0.000	0.000	0.000
11	56087	0.006	0.000	0.005	0.000
1	56210	0.023	0.000	0.000	1.000
23	60826	0.029	0.093	0.039	0.000
17	94259	0.005	0.003	0.012	0.000
9	98347	0.003	0.052	0.003	1.000
21	144831	0.008	0.005	0.000	0.000
16	184054	0.024	0.021	0.017	0.000
26	738741	0.014	0.005	0.015	1.000
10	874524	0.029	0.047	0.052	0.000
0	1016069	0.005	0.000	0.000	1.000
19	1034849	0.014	0.017	0.005	1.000
8	1191769	0.116	0.012	0.093	1.000
14	1205321	0.040	0.000	0.047	1.000
2	1676430	0.008	0.005	0.028	1.000
22	2319567	0.003	0.141	0.002	1.000
13	2397392	0.012	0.000	0.005	1.000
3	2822912	0.008	0.002	0.006	1.000
15	2929832	0.001	0.039	0.000	1.000
4	3635510	0.438	0.508	0.000	1.000
12	4115087	0.020	0.000	0.021	1.000
24	7084813	0.042	0.015	0.508	1.000
	Average	1.000	1.000	18.446	15.000

storing all documents at a site or none. While this results in an additional storage overhead for small sites, storing all documents appears reasonable for large sites as no forwarding is needed. Our results encourage the use of SVMs which substantially improve the replication rate for only a marginal increase in forwarding compared to a traditional replication scenario.

An important issue is that some queries return no results after evaluation over the entire index. For such queries, we assume that our oracle does not forward the query to non-local sites as it is known that the query will not be answered by any site. Therefore, in certain sites, even though the replication amount on the site is zero, the fraction of queries forwarded by that site may remain below 1. As an example, consider site 18, where no documents are replicated (see Table 2). The query forwarding rate remains at 0.771 since about 22.9% of queries issued to site 18 are not forwarded as they are known to not return any results.

5.2.3 Missing Documents

Our analysis so far focused on storing documents that are returned as results for a set of queries. However, not all documents are retrieved as part of a top-k result set [26]. Some documents for instance are even never returned. The common strategy is to store these documents rather than deleting them for potential future queries. Table 3 shows the replication rates for documents, that are not part in any of our result sets. The numbers are almost identical to the ones in Table 2. The distribution of the missing instances is therefore uniform with respect to region and language and they do not change the overall picture.

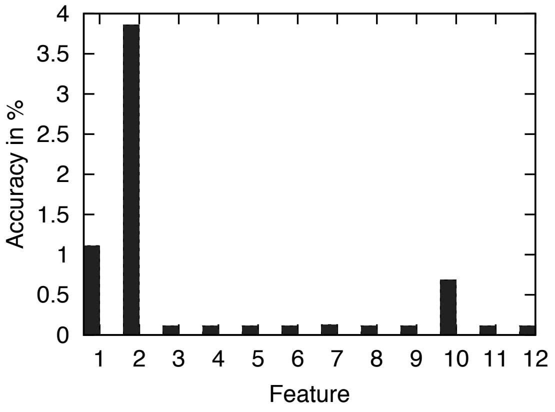


Figure 7: Contribution of features to the overall performance. The poor average performance is due to the noise that is introduced by additional and uncorrelated features.

5.2.4 Feature Analysis

In this section, we analyze the contribution of all feature to the overall prediction for the support vector machine classifiers. For this, we trained Support Vector Machines using all 12 features for the 27 sites. In the test phase, we strategically discarded all but one feature from the learned models and averaged the performances over the sites.

Fig. 7 shows the results. As already shown earlier, the features `F-region` and `F-language` possess the strongest correlation with the target values and contribute highest to the average accuracies. Nevertheless, the models are trained on all features and the performance is not as good as if trained on the respective single feature alone. This poor performance is due to the noise introduced by the uncorrelated features, which harms the learning process, and the figure justifies our initial choice of features.

6. RELATED WORK

We are not the first to propose techniques for building multi-site search engines. Baeza-Yates et al. and Cambazoglu et al. have investigated the feasibility of multi-site search engines [1, 11, 13] and specific problems on these architectures such as query forwarding [1, 13], index freshness [27], and result cache freshness [10]. Unlike traditional centralized, single-site search architectures [4, 25], the work on multi-site engines focuses on distributing the search process over multiple, cooperative, and geographically distant search sites. Our work is complementary to theirs. We specifically target the document assignment problem in multi-site search architectures.

It is important to observe that our setting shares more similarities with tiering [3] than with federated search [8] or metasearch [22]. In our setting, a site processes any local query first, and only if necessary other sites process such a query. Federated search systems typically first rank sites [9, 32] and then send queries to top-ranked sites. Our architecture is different from proposed P2P search architectures. Different from P2P search [5, 34], we do not assume churn or any adversarial behavior and the number of sites is on the order of tens instead of tens to hundreds of thousands as with peers in P2P. Even though we have not explored such possibilities, our techniques might be applicable to other forms of parallel and distributed architectures such as both federated IR architectures and P2P search architectures.

In the literature, in addition to statistical techniques [35, 36, 37], there is much research work on applying machine learning techniques to document classification [17, 24, 30]. A nice survey of related work of the latter type is provided in [29]. Our work differs from these approaches not only in the number of processed documents but also in the choice of features. In contrast to common feature representations such as tf-idf requiring expensive dictionaries, we focus on simple and inexpensive features that can be extracted on the fly. Furthermore, our approach makes use of state-of-the-art machine learning techniques such as Support Vector Machines, which are tailored to large-scale classification [7, 18, 31].

In [20], techniques are discussed for selecting partial document replicas using inference network. The emphasis in that work is on improving performance of IR systems without degrading retrieval accuracy. The same authors later investigated the impact of result caching on partial replica selection [21]. A naive document replication technique is employed recently in a geographically distributed search engine setting [1, 13]. This technique simply replicates the most popular 1% of documents on all data centers, without using any other features. The technique is not applicable to our work because its main objective is to reassign existing documents (with past access history) to sites, whereas our objective herein is to classify newly seen documents.

7. DISCUSSION

In a multi-site search engine setting, indexing and processing fewer documents per site leads to an important reduction of the amount of hardware used, thus reducing the total cost of equipment and ownership. At the same time, sites ideally are able to replicate all the documents that their local users seek to achieve low query processing latency. Reducing costs and increasing locality are competing forces that make the problem of deciding what documents to replicate in a given site non-trivial. A viable solution, however, must be able to strike a balance between these competing forces, presenting low rates of forwarded queries for low query processing latency and minimizing document replication.

With the goals of locality and cost reduction in mind, we can draw the following observations. *False positives* when classifying documents for a site imply that the site stores and indexes more documents. This metric consequently gives us an idea of how much more hardware resources we end up using unnecessarily. *False negatives*, on the other hand, imply that documents have been stored elsewhere, which lower the overall locality of the system thus leading to higher user latency.

In our work, we reported replication and forwarding rates, which are weighted variants of false positives and false negatives. That is, we obtain the replication rate by scaling the false positives with the corresponding unique terms and appropriate normalization. Vice versa, relating the false negatives to remotely issuing queries, we guarantee that the query forwarding rate is proportional to latency.

Our results show that our machine learning approach outperforms the baseline approaches, based on region or language. Although our approach requires higher storage capacities, it is characterized by small forwarding rates. As the latter improves user latency, our approach is able to provide a better user experience. Compared to identically replicated data centers, our solution saves about 45% of the storage capacity compared to replicating all documents on all sites while marginally increasing the forwarding rate.

8. CONCLUSION

We studied document assignment in multi-site search engines. We decompose the problem into several independent tasks that are solved with linear Support Vector Machines using positive and negative features obtained from the region and language of documents. Our results show a significant latency improvement when using machine learning: the fraction of forwarded queries drops from over 0.65 to 0.02. Although our findings show excellent latency values throughout the experiments, the results come at the cost of a high replication rate. Even though latency is important for user experience, there might be scenarios requiring an efficient storage with little or no overhead. Using negative features already enables a reduction of the replication factor: it drops from 19 to 15 when using negative features. Nevertheless, our approach is a step towards practical multi-site engines, and, to the best of our knowledge, the first work to approach document assignment in such architectures.

Acknowledgment

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] R. Baeza-Yates, C. Middleton, and C. Castillo. The geographical life of search. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 252–259, 2009.

[3] R. Baeza-Yates, V. Murdock, and C. Hauff. Efficiency trade-offs in two-tier web search systems. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 163–170, 2009.

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

[5] M. Bawa, G. S. Manku, and P. Raghavan. Sets: Search enhanced by topic segmentation. In *Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 306–313, 2003.

[6] B. Bohnet. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.

[7] L. Bottou and Y. LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

[8] J. Callan. Distributed information retrieval. In W. B. Croft, editor, *Advances in Information Retrieval. Recent Research from the Center for Intelligent Information Retrieval*, chapter 5, pages 127–150. Kluwer Academic Publishers, 2000.

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

[10] 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 *Proceedings of the 19th International Conference on World Wide Web*, pages 181–190, 2010.

[11] B. B. Cambazoglu, V. Plachouras, and R. Baeza-Yates. Quantifying performance and quality gains in distributed

web search engines. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 411–418, 2009.

[12] B. B. Cambazoglu, V. Plachouras, F. Junqueira, and L. Telloli. On the feasibility of geographically distributed web crawling. In *Proceedings of the 3rd International Conference on Scalable Information Systems*, 2008.

[13] 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.

[14] K. Church, A. Greenberg, and J. Hamilton. On delivering embarrassingly distributed cloud services. In *Proceedings of the 7th ACM Workshop on Hot Topics in Networks*, 2008.

[15] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[16] J. Hoffmann, M. Spranger, G. Daniel, J. Matthias, and H.-D. Burkhard. Further studies on the use of negative information in mobile robot localization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 62–67, 2006.

[17] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, 1998.

[18] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, 2006.

[19] C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008.

[20] Z. Lu and K. S. McKinley. Partial replica selection based on relevance for information retrieval. In *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 97–104, 1999.

[21] Z. Lu and K. S. McKinley. Partial collection replication versus caching for information retrieval systems. In *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 248–255, 2000.

[22] W. Meng, C. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys*, 34(1):48–89, 2002.

[23] S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2:345–389, 1998.

[24] H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 67–73, 1997.

[25] S. Orlando, R. Perego, and F. Silvestri. Design of a parallel and distributed web search engine. In *Proceedings of the Parallel Computing Conference*, pages 197–204, 2001.

[26] D. Puppin, F. Silvestri, R. Perego, and R. Baeza-Yates. Tuning the capacity of search engines: Load-driven routing and incremental caching to reduce and balance the load. *ACM Transactions on Information Systems*, 28:1–36, 2010.

[27] C. Sarigiannis, V. Plachouras, and R. Baeza-Yates. A study of the impact of index updates on distributed query processing for web search. In *Proceedings of the 31th European Conference on Information Retrieval*, pages 595–602, 2009.

[28] E. Schurman and J. Brutlag. Performance related changes and their user impact. In *Velocity: Web Performance and Operations Conference*, 2009.

[29] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[30] F. Sebastiani, A. Sperduti, and N. Valdambrini. An improved boosting algorithm and its application to automated text categorization. In *Proceedings of the 9th ACM International Conference on Information and Knowledge Management*, pages 78–85, 2000.

[31] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814, 2007.

[32] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–305, 2003.

[33] M. Stone. Cross-validation: A review. *Math. Operationsforsch. Statist. Ser. Statistics*, 9(1):127–129, 1978.

[34] C. Tang, Z. Xu, and M. Mahalingam. PeerSearch: Efficient information retrieval in peer-to-peer networks. In *Proceedings of HotNets-I, ACM SIGCOMM*, 2002.

[35] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1–2):69–90, 1999.

[36] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 42–49, 1999.

[37] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18(2/3):219–241, 2002.