Selective Search: Efficient and Effective Search of Large Textual Collections

ANAGHA KULKARNI[1], San Francisco State University
JAMIE CALLAN, Carnegie Mellon University

The traditional search solution for large collections divides the collection into subsets (*shards*), and processes the query against all shards in parallel (*exhaustive search*). The search cost and the computational requirements of this approach are often prohibitively high for organizations with few computational resources. This article investigates and extends an alternative: *selective search*, an approach that partitions the dataset based on document similarity to obtain *topic-based shards*, and searches only a few shards that are estimated to contain relevant documents for the query. We propose shard creation techniques that are scalable, efficient, self-reliant, and create topic-based shards with low variance in size, and high-density of relevant documents.

The experimental results demonstrate that selective search's effectiveness is on par with that of exhaustive search, and the corresponding search costs are substantially lower with the former. Also, the majority of the queries perform as well or better with selective search. An oracle experiment that uses optimal shard ranking for a query indicates that selective search can outperform exhaustive search's effectiveness. Comparison with a query optimization technique shows higher improvements in efficiency with selective search. The overall best efficiency is achieved when the two techniques are combined in an optimized selective search approach.

Categories and Subject Descriptors: H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval–Clustering, Search process, Selection process; H.3.4 [**INFORMATION STORAGE AND RETRIEVAL**]: Systems and Software–Distributed systems, Performance evaluation (efficiency and effectiveness)

General Terms: Design, Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Large-scale text search. Selective search. Partitioned search. Distributed information retrieval. Document collection organization. Resource selection.

ACM Reference Format:
Anagha Kulkarni and Jamie Callan, 2015. Selective Search: Efficient and Effective Search of Large Textual Collections *ACM Trans. Inf. Syst.* 1, 1, Article 1 (January 1), 34 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Traditionally, the prominent use-case of search solutions for large textual datasets has been commercial search engines. This has changed, however, with the growing

This research is sponsored in part by National Science Foundation grant IIS-1302206. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the author(s), and do not necessarily reflect those of the sponsor.
Author's addresses: A. Kulkarni, Computer Science Department, San Francisco State University, 1600 Holloway Ave, San Francisco, CA 94132; email: ak@sfsu.edu (Corresponding author)
J. Callan, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213. email: callan@cs.cmu.edu
[1]Part of this work was done when the first author was a doctoral student at Carnegie Mellon University.

DOI: http://dx.doi.org/10.1145/0000000.0000000

availability of large and information-rich collections. An increasing number of organizations and enterprises need search solutions that can process large volumes of data. The search approaches adopted by commercial search engines, however, cannot be prescribed as is to the new search applications where the operational requirements and constraints are often drastically different. For instance, commercial search engines can assume availability of large computing clusters when designing the search solution, but most small-scale organizations and research groups cannot. Search engines typically divided the large datasets into smaller subsets (*shards*) that are distributed across multiple computing nodes, and searched in parallel to provide rapid interactive search. We refer to this as the *exhaustive search* approach. The strict requirements about query response time and system throughput enforced by search engines are also not shared by many other search applications. Often batch query processing or more relaxed query response times are acceptable. Such observations motivate the work presented in this article, where the goal is to study the problem of searching large textual collections in low-resource environments.

Our previous work introduced *selective search*, an approach that can process large datasets using few computational resources [Kulkarni and Callan 2010a; 2010b; Kulkarni 2013]. This search technique first partitions the corpus, based on documents' similarity, into *topic-based* shards. During query evaluation only a few of the shards, that are estimated to contain relevant documents for the query, are searched. Lastly, the results from searched shards are merged to compile the final result list. Since the query is processed against only a few shards, as opposed to all shards, which is the case for exhaustive search, the computational requirements of selective search are much lower.

The first step of dividing the collection into topic-based shards is motivated by the *cluster hypothesis*, as per which, closely associated documents tend to be relevant to the same requests. This hypothesis suggests that if similar documents are grouped together then the relevant documents for a query also get concentrated into a few shards. Such an organization of the dataset creates shards that are semantically homogenous, and each shard can be seen as representing a distinct topic; thus the name topic-based. We develop *document allocation policies* that can divide large datasets into topic-based shards efficiently, and scale well with collection size. The proposed allocation policies are also widely applicable since they do not require knowledge of the query traffic, and do not use any external knowledge sources. Furthermore, one of the approaches is designed to create shards of nearly uniform size. This is important because such shards support better load-balancing, and provide low-variance in query run times. Most existing collection partitioning techniques lack some or all of these properties.

When a dataset is partitioned into topical shards, selective search proposes that the relevant shards for a query can be identified using *resource selection* or *resource ranking* algorithms, which have been studied extensively in the field of distributed information retrieval (DIR) (also, referred to as federated search) [Callan 2000; Shokouhi and Si 2011]. We employ a widely-used resource ranking algorithm, ReDDE [Si and Callan 2003], to operationalize the step of estimating the relevant shards for the query. Selective search evaluates the query against the top T shards in this ranking, and the returned results are merged to compile the final ranked list of documents.

We undertake a thorough empirical analysis of the selective search approach using some of the largest available document collections. The experimental results show that compared to other shard creation techniques, which have been used in prior work, the topic-based allocation policy consistently supports the best selective search effectiveness. The topic-based shards exhibit the highest concentration of relevant documents for a query, which validates the cluster hypothesis. The shards created with the size-

bounded topic-based policy exhibit more uniform size distribution, and also improve selective search efficiency.

As compared to the baseline approach, exhaustive search, the search costs for selective search are substantially lower, and much lower for the larger datasets. The corresponding search effectiveness of selective search, even when quantified using the recall-oriented MAP metric, is comparable to that of exhaustive search. Most of the previous work evaluates only at early ranks (P@{10,30}) because partial search approaches such as selective search were not expected to perform well for recall oriented tasks. The results in this article challenges this belief. Furthermore we show using an oracle experiment that selective search can significantly outperform exhaustive search for both, precision and recall-oriented tasks, given an optimal ranking of shards for the query.

Query optimization techniques[1] are widely used to improve the efficiency of a search system. We compare selective search with a well-known query optimization technique, term-bounded max-score (TBMS) [Strohman et al. 2005]. This analysis demonstrates that the improvements in search efficiency with selective search are at least twice that with query optimization. Also, selective search and query optimization, together provide the lowest search costs, indicating the complementary strengths of the two approaches.

In summary, this article extends the research on selective search by: 1. reconfirming previous claims using better evaluation framework (more accurate metrics and stronger baselines), 2. by proposing a new shard creation technique that partitions the corpus into nearly equal-sized topic-based shards, which offer the most cost-effective configurations for selective search, 3. by demonstrating that selective search of topic-based shards can significantly outperform exhaustive search's effectiveness if the optimal shard ranking can be estimated, and 4. by investigating selective search and a query optimization technique for their individual and combined ability to improve search efficiency.

The remainder of the paper is organized as follows. The next section situates our work in the context of related prior research. Sections 3 and 4 describe the selective search and its centerpiece of shard creation. Section 5 provides the details of the experimental framework. Section 6 presents the results and our interpretations of the results. Section 7 studies selective search performance when optimal ranking of shards is available. Section 8 reports the comparative analysis of selective search and query optimization technique, and our conclusions from this work are shared in Section 9.

2. RELATED WORK

Three lines of research provide the necessary context for our work: cluster-based retrieval, resource selection research from federated search, and work on search efficiency of large-scale search system.

2.1. Cluster-based retrieval

Search systems that employed clustering algorithms were more popular a few decades ago than they have been in the recent past. In an early work by Salton [1971], which predates the use of inverted indexes, document clustering was employed to improve retrieval efficiency of the search system. In contrast, Jardine and van Rijsbergen employed document clustering to improve retrieval effectiveness [Jardine and van Rijsbergen 1971]. More specifically, the collection was organized into a hierarchy of progressively more similar document clusters using the single-link clustering algorithm. During query evaluation the single most similar cluster from the hierarchy was chosen

[1]Also referred to as dynamic index pruning, and top-k retrieval.

for each query using a top-down tree traversal based approach. The empirical evaluation demonstrated that the proposed technique improves over the traditional ad hoc document retrieval only when an oracle selects the relevant cluster for the query. Although, this work had a limited impact on the subsequent techniques proposed in this area, this paper remains an important landmark in cluster-based retrieval because it formally defined the *cluster hypothesis*.

Croft [1980] provided a more principled cluster selection technique by formulating the task in a probabilistic framework where the conditional probability of the cluster being relevant to the query was estimated using Bayes' rule as follows.

$$P(C_i|Q) = \frac{P(Q|C_i)P(C_i)}{P(Q)} \propto P(Q|C_i)P(C_i) \tag{1}$$

Here the prior probability, $P(C_i)$ is simply the size of the cluster, and $P(Q|C_i)$ is approximated by the product of individual query term generation probabilities as: $P(Q|C_i) = \prod_{q_j \epsilon Q} P(q_j|C_i)$. This cluster ranking model is similar to the widely used *query likelihood model* for document retrieval. Empirical evaluation demonstrated improvement in precision but not in recall as compared to the baseline, cosine similarity based document retrieval from the complete collection. Subsequent work in cluster-based retrieval that attempted to reproduce the trends observed by Croft, and Jardine and van Rijsbergen, using better evaluation setups, were unable to demonstrate consistent and useful gains in search effectiveness [Griffiths et al. 1986; Voorhees 1985; Willett 1988]. In fact, Voorhees concluded after an extensive investigation that even the combination of the costliest clustering algorithm (complete linkage) and a costly top-down cluster selection approach could only provide marginal improvement over the baseline document retrieval approach [Voorhees 1985]. A later work by El-Hamdouchi and Willett also came to similar conclusions [El-Hamdouchi and Willett 1989].

There are several potential causes for the inconsistent improvements observed with cluster-based retrieval methods, such as, ineffective document representation (only document titles or abstracts were used), sub-optimal cluster selection algorithms, datasets with short documents, and small datasets. Overall, none of the studies could justify the additional cost of creating elaborate collection hierarchies by demonstrating consistent and substantial improvements in search effectiveness. Also, the scalability of the hierarchical clustering approaches was becoming a serious limitation as larger document collections were starting to become available. As a result, this line of work that focused on clustering collections into smaller subsets went dormant.

The work by Xu and Croft is related to cluster-based retrieval approaches because it uses the cluster hypothesis [Xu and Croft 1999]. However, it did not continue the tradition of creating document taxonomies, neither did it expect to improve search effectiveness over complete search. Xu and Croft employed an efficient collection partitioning technique (linear time complexity in collection size), and a more effective cluster selection algorithm. Specifically, a two-pass K-means clustering algorithm and a Kullback-Liebler divergence based distance metric were employed to partition the complete collection into clusters of similar documents. The efficiency of the partitioning algorithm allowed for experimentation with collections containing over a half a million documents while using a full-text retrieval approach. For each query the relevant clusters were selected by computing the distance between the query's unigram language model and each cluster's unigram language model using Kullback-Liebler divergence. Empirical evaluation compared the accuracy of the documents retrieved from the top 10 clusters to those retrieved from the complete collection. The effectiveness of the proposed approach was found to be on par with the baseline. When

compared to a second baseline system that organized the documents based on their sources, the proposed approach provided substantially more accurate retrieval.

More recently Puppin et al. employed a co-clustering algorithm to partition a large document collection using query log data [Puppin et al. 2006]. The query log covered a period of time when exhaustive search was used for each query. These training queries and the documents that they retrieved were co-clustered to generate a set of *query clusters* and a set of corresponding *document clusters*. A collection of 6 million documents was partitioned into 17 clusters. Documents that were not retrieved by any of the training queries could not be clustered using the proposed technique (52% of the collection). Searching the top few clusters was found to be more accurate than searching the top few baseline partitions created using random allocation.

2.2. Resource selection

The field of federated search studies the problem of servicing search requests from multiple (often autonomous) collections of documents (*resources*) [Callan 2000; Shokouhi and Si 2011]. The ability to identify a small set of resources that are relevant to the query is central to the success of any federated search system. As such, this research problem has been investigated extensively, and the proposed algorithms can be classified into one of three families: vocabulary-based, sample-based, and feature-based algorithms. In our work we adapt one of the widely used sample-based algorithms, ReDDE [Si and Callan 2003], for the task of ranking the shards based on their estimated relevance to the query. Below we review the ReDDE algorithm, and summarize the other categories of resource ranking algorithms.

The *sample-based algorithms* use a subset of documents sampled from each resource to define a ranking function. Often, the sampled documents are stored in an inverted index, referred to as, the *central sample index* (CSI), which is the basis for many of the sample-based approaches [Si and Callan 2003; Shokouhi 2007; Thomas and Shokouhi 2009; Kulkarni et al. 2012]. The ReDDE algorithm runs the query against the CSI, and the top n retrieved documents are assumed to be relevant [Si and Callan 2003]. If n_R is the number of documents in n that are mapped to shard R then a score s_R for each R is computed as: $\theta_R = n_R * w_R$, where the shard weight w_R is the ratio of size of the shard ($|R|$) and the size of its sample ($|S_R|$). The shard scores θ_R are then normalized to obtain a valid probability distribution which is used to rank the shards.

A vocabulary-based approach learns a representative model for each resource based on its summary statistics. These models are then used to estimate the resources' relevance to the query [Gravano et al. 1994; Gravano and García-Molina 1995; Gravano et al. 1999; Callan et al. 1995; Callan 2000]. The feature-based approaches combine various sources of information, such as, query category, click-through data, relevance estimates from vocabulary-based and sample-based approaches, to define features. A classification model is learned for each resource based on these features, which is then used to rank the resources [Arguello et al. 2009].

2.3. Large-scale search

A host of strategies are typically employed by search systems to meet various operational requirements, such as, query response time, throughput, resource utilization, hardware costs, and infrastructure costs [Barroso et al. 2003; Chowdhury and Pass 2003; Risvik et al. 2003; Cacheda et al. 2007; Moffat et al. 2007; Baeza-Yates et al. 2009; Macdonald et al. 2012; Freire et al. 2013]. Index partitioning is commonly used to enable distributed query processing which in turn improves query response time. Another popular technique is that of index replication where data redundancy is employed to achieve the needed throughput and fault-tolerance.

Tiering is another common strategy where the collection is divided based on documents' quality or geographical location, with the goal of evaluating most queries against only a small subset of the collection. A *document quality* based tiering approach separates the high quality documents into the first tier, and the remaining documents are delegated to the lower tier(s) [Barroso et al. 2003; Baeza-Yates et al. 2009]. The document quality is defined as a function of various document properties, such as, its spam score, click probability, and number of inlinks. In this architecture most queries are served by the first tier which contains a relatively small subset of the collection and the lower tiers act as fall-back options which are searched only if the higher tier fails to generate sufficient search results.

For Web search, it has been observed that search intents tend to be geographically localized. A tiering technique that exploits this property can be used to drive down the search cost. *Geographical tiering* divides the document collection into tiers based on the geographical source of the documents [Cambazoglu et al. 2009]. Cambazoglu et al. maintain a goal of constructing search results in a geographically tiered search system that are identical to those generated by complete search. Each incoming query is processed against the local tier by default and the highest possible document score that could be computed for the query at each of the remote tiers is estimated using statistics obtained from offline queries. The highest document score estimates are compared to the score of the kth document retrieved from the local tier to predict if the remote tier would contribute document(s) to the top k ranks. The query is forwarded to all the remote tier which are predicted to contribute one or more documents. In addition to reducing the search cost by converging to a smaller search space this technique also has the other advantages of reducing the network traffic and query response time.

It is important to note that the tiers created by any of the above strategies are still quite large. In fact, because of their size each tier is typically further divided into multiple smaller shards that are evaluated in parallel for each query. For search environments with limited computing resources the cost of searching even just the primary tier can be high.

In addition to the above techniques caching strategies are routinely employed at several levels of the search system to reduce the disk and network activities required for query evaluation. The posting lists for frequently accessed terms, for instance, are cached in main memory [Baeza-Yates et al. 2007; Saraiva et al. 2001]. Caching techniques that store partially computed results may also be employed to further improve system's efficiency [Baeza-Yates et al. 2007; Fagni et al. 2006; Lempel and Moran 2003; Gan and Suel 2009]. Query results are also cached in order to leverage one of the common trends on Web search: *query popularity spikes* [Puppin et al. 2010]. When a query can be served directly from the cache, search cost savings are substantial. Like for exhaustive search, caching techniques can be applied for selective search as well. A few interesting future research problems along this line are discussed in Section 9.

Index pruning techniques study the problem of improving search efficiency by manipulating the inverted index of the document collection. *Static index pruning* techniques explore the space of *lossy compression* techniques where certain portions of the inverted index that would have minimal adverse effect on the search effectiveness are permanently excluded [Carmel et al. 2001; de Moura et al. 2005; Büttcher and Clarke 2006; Nguyen 2009]. *Dynamic index pruning* are lossless query optimization techniques that aim at reducing the amount of computations needed for a query, on average [Turtle and Flood 1995; Brown 1995; Strohman et al. 2005; Tonellotto et al. 2013]. Of the various techniques reported above, dynamic index pruning is most closely related to our work. Thus we review a state-of-the-art dynamic index pruning technique, *term bounded max-score* (TBMS), below, and later provide a comparative analysis of TBMS and our approach in Section 8.

The *term bounded max-score* (TBMS) algorithm [Strohman et al. 2005] combines the strengths of two prior works: *max-score* algorithm [Turtle and Flood 1995] and *topdocs lists* [Brown 1995]. As used by Strohman, topdocs are partial posting lists that are sorted on document weight which is the ratio of term frequency and document length. Query evaluation starts by adding all the entries from topdocs of each query term to the candidate document set, D. The documents in the set D are scored and ranked to provide the first version of the top k results for the query. The smallest document score from set D is used as the threshold for the max-score algorithm. Specifically, when a document outside of the candidate set is considered for evaluation, first an upper bound on the document's score is estimated by adding together the highest possible score contributions of each query term, and then the resulting document score is compared to the threshold. If the document score is not higher than the threshold then the document is guaranteed to be not in the top k ranks, and thus is not evaluated. TBMS uses this methodology to skip over posting lists entries and also query terms to reduce the number of computations needed for a query.

3. SELECTIVE SEARCH

Selective search is a recently proposed approach for large-scale search in low-resource environments [Kulkarni and Callan 2010a; 2010b; Kulkarni 2013]. This search approach consists of two main phases: shard creation, and shard ranking and retrieval. During shard creation the collection is organized into smaller partitions (shards) using a *document allocation policy*. This is an offline step that is seldom repeated for a dataset. The shard ranking and retrieval phase performs query evaluation, and thus is the online step that is repeated for every query. For query evaluation, a small subset of shards that are likely to contain relevant documents for the query is identified and searched. The returned result lists are merged to compile the final ranked results.

In order for this search strategy to work effectively and efficiently, the following two requirements need to be met. First, the created shards must exhibit a skewed distribution of relevant documents, that is, the majority of relevant documents for a query must be present in a single (or few) shard(s). This property must hold for every query (or most queries) that the search system receives. The second requirement is that given such a partitioning of the documents, it must be possible to identify the relevant shards for the query. Selective search proposes that when these requirements are met large document collections can be searched efficiently using few computational resources, and the corresponding search effectiveness would be comparable to that of exhaustive search.

For the shard creation step we experiment with four *document allocation policies*. In addition to the relevance skew property, we develop these policies to be sensitive to other crucial properties, such as, scalability, and ability to parallelize the process of collection partitioning. The details about these allocation policies are given in the following sections. For the second requirement, selective search leverages the prior work on *resource selection* from the distributed IR field (Section 2.2). The resource ranking algorithms are designed to score the resources (text databases) based on their estimated relevance to the query [Shokouhi and Si 2011]. Selective search employs a widely-used resource ranking algorithm to score the shards for a query, and then processes the query against the top T shards from the ranking. The document scores in each of the T result lists are comparable and thus merging the lists is straightforward. This is so because the global statistics, such as, the idf (inverse document frequency over the complete collection), that are used for score normalization, are shared with all shards in advance at partitioning time.

4. SHARD CREATION

A document collection can be divided into a set of shards using many different document-to-shard allocation policies. One of our requirements for the document allocation policy is that it be generally applicable. We thus restrict ourselves to only those *document allocation policies* that do not rely on external resources such as query-logs or categorization schemes, which were both used in previous work [Larkey et al. 2000; Puppin et al. 2006]. Although such resources can provide valuable additional information that can be used in the allocation decision, they are not always available.

We also assume that the document allocation policy must operate on large document collections and might have access to limited computational resources. Efficiency of the approach is thus an important consideration. Algorithms that can be parallelized, that can scale linearly in collection size are preferred. Finally, the shards created by the document allocation policy should support competitive selective search effectiveness. When the relevant documents for a query are concentrated in a few shards the search can be restricted to those shards without compromising search effectiveness. Thus a successful document allocation policy would organize the collection such that only a few shards contain the majority of the relevant document for a particular query. We study three type of document allocation policies that satisfy the above requirements to varying extents: random, source-based and topic-based, which are described next. We only study one-to-one allocation policies that assign a document to only one shard. Thus all the three policies described below partition the dataset into disjoint shards.

4.1. Random document allocation

If the collection has to be partitioned into K shards, then the random allocation policy assigns each document to one of the K shards at random with equal probability. This policy has the advantages of being the most efficient, scalable, and easy to implement. Also, its applicability is high since it can be used to partition any document collection. It naturally lends itself to parallelization, and appending a new set of documents to existing shards is also straightforward. It is thus not a surprise that commercial Web search engines such as Google [Barroso et al. 2003] and Bing[2] employ this allocation policy for sharding.

On the downside, random allocation policy spreads the relevant documents across multiple shards instead of concentrating them into a few shards. Thus it is likely that shards created using a random policy cannot support effective selective search. Nonetheless it is a contender because of its advantages, its use in prior research [Puppin et al. 2006], and its use by large-scale commercial search systems.

4.2. Source-based document allocation

The source-based allocation policy organizes the collection into shards using the *source* of the document, such as, the company department that generated the document, or the Web host. The motivation for this policy is summarized by the *cluster hypothesis*, as per which, closely associated documents tend to be relevant to the same information need [van Rijsbergen 1979]. Thus the premise is that the documents from the same source are closely associated to each other, and thus source-based partitioning would concentrate the relevant documents into few shards. For instance, it would be reasonable to assume that majority of the web-pages under the *espn.com* source are related to sports and to each other. There are of course abundant counter examples. For example, Wikipedia documents share the Web host but are not closely associated in most cases.

[2]Personal communication.

ALGORITHM 1: Sample-based K-means (SB K-means)

Input: Document collection: C, Sample size: |S|, Number of shards: K.
Output: Topical shards.

```
// Sample |S| documents from collection C
S ← Sample(C, |S|)

// LEARN PHASE: Cluster the sampled documents to obtain K cluster-centroids.
CENT_K ← K-Means(S, K)

// PARTITION PHASE: Divide the collection into K shards using the centroids.
R_K ← Partition(C, CENT_K)

return R_K
```

The source-based allocation policy was widely used by prior research in distributed IR with the intent of replicating real world distributed search environments consisting of independent document sources [Callan et al. 1995; Callan et al. 1999; French et al. 1999; Voorhees et al. 1995; Xu and Croft 1999]. Often document URLs provided the *source* information. Each unique top-level hostname was assigned a separate shard and the documents were partitioned accordingly. However, for many recent collections this strategy would lead to a very large number of small partitions. In this work we employ the following approach instead. We first sort the document URLs, which arranges documents from the same website consecutively, and then groups of $|M|/K$ consecutive documents are assigned to each shard where $|M|$ is the total number of documents in the collection and K is the total number of shards to be created. However, this is not a strict policy. Splitting of websites across shards is avoided wherever possible.

4.3. Topic-based document allocation

The topic-based allocation policy also appeals to the *cluster hypothesis* in order to group closely associated, and thus relevant documents, into few shards [van Rijsbergen 1979]. More specifically, documents that are *lexically similar* are considered to be closely associated by this allocation policy. Such an organization of the dataset creates shards that are semantically homogenous, and each shard can be seen as representing a unique *topic*. The two implementations that we have developed for topic-based allocation policy are described next.

4.3.1. Sample-based K-means (SB K-means). We employ a variant of the time-tested K-means clustering algorithm [Lloyd 2006] to partition the documents based on their similarity. K-means is a simple version of the *Expectation-Maximization* algorithm [Dempster et al. 1977] that starts by partitioning the dataset into K clusters using a set of K seed centroids. Following this the algorithm iteratively alternates between the Maximization step where the centroid estimates are updated using the current dataset partitions, and the Expectation step where the dataset is repartitioned using the revised centroids.

Although the computational complexity of the K-means algorithm is linear in the number of documents, applying this algorithm to very large collections can still be prohibitively expensive in resource-constrained environments. To solve this scalability problem we develop a sample-based K-means (SB K-means) approach that applies the standard K-means algorithm to only a small subset of documents. Algorithm 1 provides the high-level steps of the SB K-means algorithm, which starts by sampling a small set of documents from the collection. The sampled documents are then parti-

tioned into K clusters using standard K-means algorithms, in the *learn* phase. The K cluster-centroids are used in the *partition* phase to divide the complete collection into K topical shards. The specific parameterizations used in these steps is reported next.

Simple random sampling (SRS) is employed to compile the subset of documents (S) that is used by the learn phase. The SRS strategy chooses documents at random from the complete collection (without replacement). The size of the sample $|S|$ is determined based on operational requirements. For example, $|S|$ can be chosen such that the learn phase of the algorithm fits in the memory. The effects of this sample size selection strategy are analyzed in Section 4.3.3. We use a *vocabulary-size based rejection sampling (VRS)* technique to sample K documents that serve as the seed centroids for the K-means algorithm in the learn phase. The VRS technique first chooses a candidate document from the sampled documents, and then accepts it if its vocabulary-size is above certain threshold τ, else it is rejected. We set τ to be the average document vocabulary-size for the sample. This strategy is motivated by the observation that small vocabulary documents, such as, the *page-not-found* error pages, cannot serve as stable anchors for the clustering process[3]. A fixed number of iteration (specifically, 5) are conducted of the K-means algorithm to limit the run-time of the learn step.

The K-means algorithm requires a similarity or a distance metric for document-to-centroid assignments during the expectation step. We employ a symmetric version of negative Kullback-Liebler divergence that computes the similarity between a document D and a centroid C^i as follows.

$$sim(C^i, D) = \sum_{w \in C^i \cap D} p_{c^i}(w)\ log \frac{p_d(w)}{\lambda\ p_B(w)} + \sum_{w \in C^i \cap D} p_d(w)\ log \frac{p_{c^i}(w)}{\lambda\ p_B(w)} \tag{2}$$

$p_c^i(w)$ and $p_d(w)$ are the unigram language models of the cluster-centroid C^i and the document D, respectively. $p_B(w)$ is the probability of the term w in the background model which is the arithmetic mean of the K centroid models. λ is the smoothing parameter. Using the maximum likelihood estimation (MLE), the cluster-centroid language model is,

$$p_c^i(w) = \frac{c(w, C^i)}{\sum_{w'} c(w', C^i)} \tag{3}$$

where $c(w, C^i)$ is the occurrence count of w in C^i. Following Zhai and Lafferty [2004], we estimate $p_d(w)$ using MLE with Jelinek-Mercer smoothing which gives

$$p_d(w) = (1 - \lambda)\ \frac{c(w, D)}{\sum_{w'} c(w', D)} + \lambda\ p_B(w) \tag{4}$$

As such, the presence of $p_B(w)$ in the denominator plays an important role of incorporating the inverse collection frequency of the term into the metric which Zhai and Lafferty [2004] found to behave similar to the traditional inverse document frequency (IDF) statistic[4]. The above similarity metric is also used in the partition phase to infer the document-to-shard allocation for the remaining documents in the collection. A document is assigned to the shard with which it exhibits highest similarity, and any ties are broken randomly.

[3]A seed centroid selection strategy that is based on more reliable measures of document quality (such as, PageRank) than document vocabulary-size might support better clustering solutions.

[4]Please refer to [Ogilvie and Callan 2001] for the transformation of negative KL-divergence with smoothing into a similarity metric as used in Equation 2.

ALGORITHM 2: Size-bounded Sample-based K-means (SB^2 K-means)

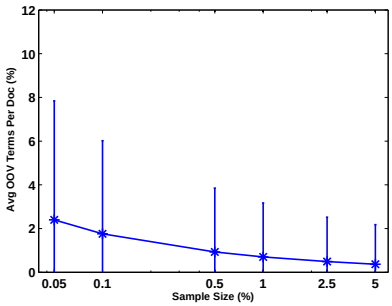
Input: Document collection: C, Sample size: |S|, Number of shards: K.
Output: Topical shards.

```
// Sample |S| documents from collection C
S ← Sample(C, |S|)

// LEARN PHASE: Cluster the sampled documents to obtain K centroids and K
   sample-shards.
{CENT_K, R^s_K} ← K-Means(S, K)

// SPLIT PHASE: Divide the large sample-shards.
CENT_{K_split} ← Split(R^s_K)

// PARTITION PHASE: Divide the collection into K_split shards using the centroids.
R_{K_split} ← Partition(C, CENT_{K_split})

// MERGE PHASE: Merge small shards with other shards.
R_{K_merge} ← Merge(R_{K_split})

return R_{K_merge}
```

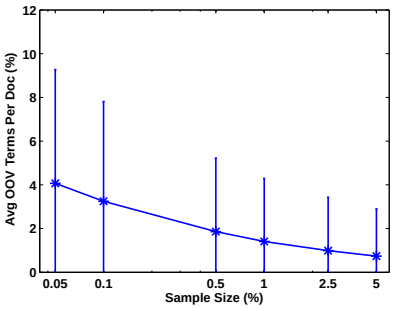
The computational complexity of traditional K-means algorithm is $O(K|M||V_d|I) + O(K|V_M|I)$, where $|M|$ is number of documents in the collection, $|V_d|$ is the average document vocabulary size, I is the number of K-means iterations, and $|V_M|$ is the vocabulary size of the complete collection. The two components in the complexity formulation corresponds to the Maximization and Expectation steps, respectively. In contrast, the complexity of the SB K-means algorithm is lower at: $O(K|S||V_d|I) + O(K|V_S|I) + O(K|M||V_d|)$, where $|V_S|$ is the vocabulary size of the sample. The third component models the complexity of the non-iterative partition step. Since $|S| \ll |M|$ and $V_S \ll V_M$, the first two components of the SB K-means complexity are substantially smaller than those of traditional K-means. The partition step naturally lends itself to parallelization because the allocation decision for a document is independent of the inference for any other document. Thus the partition phase can be distributed across all the available computing nodes.

4.3.2. Size-bounded Sample-based K-means (SB^2 K-means). Distributed search systems prefer to divide the collection into equal sized partitions which allows for better load balancing and also provides low variance in query run times. The SB K-means approach, however, is not guaranteed to create equal sized shards. The inherent topical composition of the collection determines the size distribution of its shards. For one of the datasets used in this work, the largest topical shard created by SB K-means is twenty-four times bigger than the average shard size, and at the same time many other shards are nearly empty. Overall, there is a large variance in size of the shards created by SB K-means. In order to address this limitation we propose the *size-bounded sample-based K-means* algorithm (SB^2 K-means), which is an extension of the above approach.

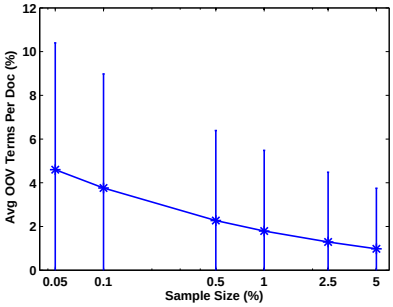
The *size-bounded sample-based K-means* algorithm consists of four phases: *learn*, *split*, *partition*, and *merge* (Algorithm 2). Like in SB K-means, the learn phase clusters the sampled documents into K sample-shards. The split phase employs an iterative process which identifies *large* sample-shards, and divides them using the standard



(a) GOV2 Dataset. Average Document length: 949 terms.



(b) ClueWeb09-B Dataset. Average Document length: 918 terms.



(c) ClueWeb09-English Dataset. Average Document length: 757 terms.

Fig. 1: Sample size vs. percentage of OOV terms per document, on average. (Error bars indicate one standard deviation of the mean. X-axis in log scale.)

K-means algorithm (details below). The resulting K_{split} cluster-centroids are used by the partition step to divide the complete collection into K_{split} shards. The merge phase, also an iterative process, identifies *small* shards, and combines them with other shards to provide the final set of K_{merge} shards.

For the split step we define large sample-shards as those that are over 110% of the targeted size, which is $|S|/K$. The iterations of the split step continue until there are no more large sample-shards or until a maximum iteration count is reached. We allowed at most five split iterations. The final step of the algorithm, merge, defines small shards as those with fewer documents than 90% of the targeted shard size. The shards that are not large (size greater than 110% of the targeted size) are identified as the potential *sink* shards. Notice that the source and the sink sets are not mutually exclusive. A merging iteration starts with the largest sink shard and ends when the smallest sink shard has been considered. Each sink shard absorbs the largest possible small shard as long as the resulting shard is not a large shard. At most five repetitions of the merge process were allowed. Due to the iterative nature of this merge strategy more than two original shards can get merged.

4.3.3. Sample-size and OOV terms. Both the topic-based allocation policies use a sample (S), instead of the entire collection, to define the topical shards in the collection, in order to reduce the computational cost of the partitioning process. However, using a

Table I: Datasets.

Dataset	Size (GB)	Number of Documents	Number of Words (billion)	Vocabulary Size (million)	Average Document Length
GOV2	400	25,205,179	24	39	949
ClueWeb09-B	1500	50,220,423	46	96	918
ClueWeb09-English	15000	503,903,810	381	1,226	757

Table II: Query sets.

Dataset	Number of Queries	Average Query Length	Average Number of Relevant Documents per Query	Number of Relevance Levels
GOV2	150	3.1	181 ($\pm$ 149)	2
ClueWeb09-B	200	2.1	66 ($\pm$ 45)	4
ClueWeb09-English	200	2.1	107 ($\pm$ 69)	4

subset of the corpus also introduces the issue of out-of-vocabulary (OOV) terms during inference. The documents outside of the sample are bound to contain terms that were not observed in S and thus are absent from the learned cluster-centroids. In this situation inference must proceed using the seen terms and ignore the OOV terms. However, the inference quality can potentially degrade because of the discounting of the OOV terms. As a result, it may be important to select a sample size that leads to a small percentage of OOV terms per document. Note that inference occurs at the document level and thus as long as the percentage of OOV terms per document is small – even if the overall percentage of words that are out-of-vocabulary is high – the inference quality for each document would not be affected severely.

We know from Heaps' law [Heaps 1978] that when examining a corpus, the rate at which vocabulary is discovered tapers off as the examination continues. Based on this we hypothesize that using a relatively small sample might be sufficient to obtain an acceptably low percentage of OOV terms per document, on average. We verify this hypothesis empirically. Figure 1 plots the sample size versus the average percentage of OOV terms per document for the three datasets used in this work. A sample size as small as 0.05% of the collection is sufficient to discover more than 95% of the document vocabulary, on average. Also note that the drop in the average values is sub-linear in sample size. This is in accordance with Heaps' law. Thus after a certain point there is little benefit in increasing the sample size since additional documents do not reduce the percentage of OOV terms significantly.

The average document lengths for GOV2, ClueWeb09-B, and ClueWeb09-English are progressively smaller at 949, 918, and 757 terms, respectively. The average document length directly influences the rate of vocabulary discovery, and consequentially the average OOV terms per document. This is evidenced by the consistent trend in the plots where the average OOV terms per document for GOV2 is lowest, and that of ClueWeb09-English is highest among the three datasets.

5. EXPERIMENTAL METHODOLOGY

This section describes the methodology we adopted for the experimental evaluation of the proposed search approach. First we define the datasets, the corresponding query sets, and the evaluation metrics used for the analysis.

5.1. Data

For the empirical analysis three large datasets were used in this work: GOV2, the CategoryB portion of ClueWeb09, and the English subset of ClueWeb09. These are summarized in Table I. The GOV2 corpus [Clarke et al. 2004] consists of 25 million documents from US government domains, such as .gov and .us, and also from government related websites, such as, www.ncgov.com and www.youroklahoma.com[5]. 150 topics from 2004 and 2005 TREC Terabyte track were used for evaluation with this dataset and the statistics for these queries and their corresponding relevance judgments are provided in the Table II. ClueWeb09 is a newer dataset that consists of 1 billion web pages crawled in January and February 2009. The ClueWeb09-B dataset consists of the first 50 million English pages, and the ClueWeb09-English consists of all the English pages in the dataset (over 500 million). For evaluation with both, ClueWeb09-B and ClueWeb09-English datasets, the 200 topics from the TREC Web tracks 2009 through 2012 were used.

5.2. Evaluation Metrics

A set of standard IR metrics: P@{10,30,100}, NDCG@100, and MAP, was used to quantify search effectiveness at various recall levels, and to model different grades of relevance. Search efficiency is modeled using two cost metrics, $C_{Total}(q)$ and $C_{Latency}(q)$, that are inspired by the metrics employed by Kulkarni et al. [2012], and Aly et al. [2013]. C_{Total} provides an upper bound on the number of documents evaluated for the query q, and $C_{Latency}(q)$ quantifies the longest execution path for the query, assuming a distributed query processing framework. The formulation for the two search cost metrics is as follows.

$$C_{Total}(q) = \sum_{i=1}^{T} |D^q_{S[i]}| + |D^q_{CSI}| \tag{5}$$

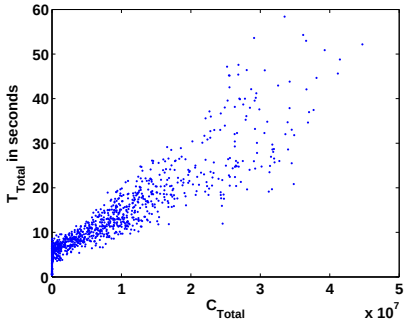
$$C_{Latency}(q) = \max_{1 \le i \le T} |D^q_{S[i]}| + |D^q_{CSI}| \tag{6}$$

where $|D^q_{S[i]}|$ is the number of documents that contain at least one of the query terms, in the ith shard. In other words, the size of the candidate document set for q in $S[i]$. The second component in the above equations models the cost of the shard ranking process. We employ a sample-based shard ranking algorithm, ReDDE, that uses the central sample index (CSI) to infer shard ranking for each query (Section 2). The cost of executing ReDDE is quantified using $|D^q_{CSI}|$ which is the number of documents in CSI that contain at least one of the query terms. The cost values reported in the result tables are averages over all evaluation queries.

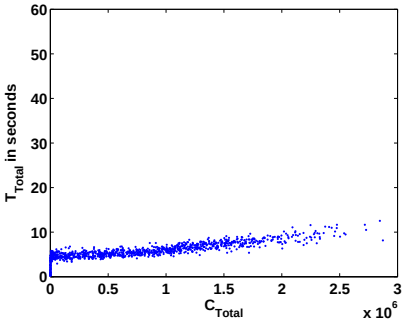
The choice of these cost metrics was driven by a couple of factors. First factor was practical constraints. For the investigations that are undertaken in this work, each of the three large datasets have to be partitioned into shards in four different ways. It is non-trivial to be able to accommodate multiple copies of these datasets on our modest and shared computing platform. Instead, the above metrics allow us to keep a single copy of each dataset and perform rapid experimentation with several different configurations using simulated retrieval. Secondly, a recent prior work has demonstrated that the number of documents evaluated for a query strongly correlates with the more direct measures of query evaluation cost, such as, query response time [Macdonald et al. 2012]. Below we reconfirm this finding using the ClueWeb09-B dataset.

The scatter plot in Figure 2a compares the C_{Total} cost metric with the equivalent time-based metric T_{Total} which is simply the response time for each query when it is

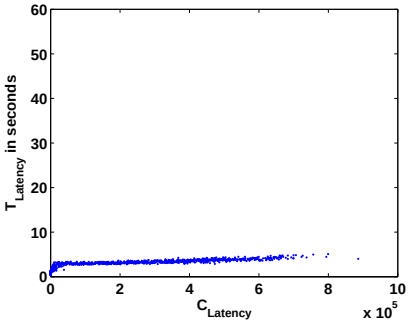
[5]http://ir.dcs.gla.ac.uk/test_collections/GOV2-summary.htm



(a) C_{Total} versus T_{Total} for Complete Monolith Inverted Index.



(b) C_{Total} versus T_{Total} for Topic-Based Partitioned Inverted Indexes.



(c) $C_{Latency}$ versus $T_{Latency}$ for Topic-Based Partitioned Inverted Indexes.

Fig. 2: Validation of Cost Metrics. Dataset: ClueWeb09-B.

run against the complete inverted index for the dataset. For this analysis 1000 queries were sampled from the TREC Million Query Track query logs, and run against a single inverted index for the ClueWeb09-B dataset on a 8-core Intel Xeon 2.60GHz with 16GB RAM. As is evidenced by the scatter plot, the number of documents evaluated for a query is strongly correlated with the query evaluation time. Specifically, the Spearman's ρ correlation is 0.95 for this data. Plots 2b and 2c analyze these two variables in the context of the distributed selective search setup. Specifically, the ClueWeb09-B dataset was partitioned into topic-based shards, which were transformed into inverted indexes. Each query was run against the CSI to identify the relevant shards for the query, and the top five shards were searched in parallel for each query on the same hardware configuration as above. The T_{Total} is the total of elapsed time at CSI and each of the five shards searched for the query, and the $T_{Latency}$ is the sum of elapsed time at CSI and the longest running shard search for the query. Both these metrics show strong correlation with their equivalent cost metrics, C_{Total} and $C_{Latency}$, respectively. The Spearman's ρ correlation between C_{Total} and T_{Total} is 0.88, and between $C_{Latency}$ and $T_{Latency}$ is 0.91.

5.3. Experimental Setup

For selective search experiments each dataset was partitioned into K shards using one of the document allocation policies. The value for K was chosen such that the av-

erage shard size would be half a million documents. Thus GOV2 was partitioned into 50 shards, ClueWeb09-B into 100, and ClueWeb09-English into 1000. Recall that the number of shards created by SB^2 K-means allocation policy is not strictly deterministic. The SB^2 K-means algorithm was seeded with 50, 100, and 1000 as the K values for GOV2, ClueWeb09-B, and ClueWeb09-English, respectively. The final number of shards that SB^2 K-means partitioned GOV2, ClueWeb09-B, and ClueWeb09-English into are 50, 92, and 807, respectively.

For the topic-based policies we also need to select the size of the sample that would be used by the *learn* phase of the algorithms to estimate the cluster-centroids (Section 4.3.1 and 4.3.2). We leverage the observations from Section 4.3.3, that analyzed the effects of sample size on the percentage of OOV terms per document, to select a sample size such that the dataset partitioning process is efficient. For GOV2 and ClueWeb09-B datasets we sample 1% (250K and 500K documents), and for the largest dataset, ClueWeb09-English, we sample 0.1% (500K) of the collection documents.

From each dataset a 0.5% sample of the collection documents was compiled using simple-random-sampling to construct a central sample index (CSI) which is needed for the resource ranking algorithm, ReDDE. The effect of the CSI size on resource ranking has been studied in the distributed IR research [Shokouhi and Si 2011]. Although larger CSI sizes support more effective resource ranking, the corresponding costs are higher. An investigation of the effects of CSI size in the context of selective search is studied in Section 6.5. We chose the CSI size of 0.5% because it provided a good balance between ranking effectiveness and cost in our experiments. For each query, the top T shards from ReDDE's ranking were searched, and the T result lists were merged to compile the final document ranking. We experimented with the range of T values to study search performance under different levels of resource constraints. For every T value the corresponding exhaustive search baseline was created by randomly partitioning the collection into T shards, and searching all T shards in parallel.

6. RESULTS AND DISCUSSION

The results for the experiments that compare exhaustive search with selective search are given in Tables III through V for the three datasets. Some of the prominent trends that span across the three datasets are as follows. The document allocation policy used to partition the collection strongly influences the performance of the selective search approach. The retrieval effectiveness of selective search with random shards is substantially lower than that of exhaustive search, especially so at deeper ranks (P@100, MAP). Selective search with source-based shards demonstrates better ability to compete with exhaustive search, however, even this setup struggles to perform well at deeper ranks, especially for the larger datasets.

The topic-based allocation policies (SB K-means and SB^2 K-means) support the most competitive selective search performance. Although, costlier than random and source-based policies, the topic-based policies converge to exhaustive search effectiveness fastest. When compared to each other, the SB K-means approach sometimes achieves exhaustive search's effectiveness sooner than the SB^2 K-means, however, the corresponding search costs are higher for SB K-means. Shard ranking algorithms are known to exhibit bias for larger shards. Since SB K-means shards are expected to have high variance in their sizes it could be that larger shards are ranked highly which increases the costs in this configuration. We investigate the distribution of sizes for topic-based shards in detail in Section 6.1. The best balance between search effectiveness and efficiency is provided by selective search with SB^2 K-means shards. Because of its higher performance the rest of the analysis focuses on selective search with SB^2 K-means shards.

Table III: Results for exhaustive search, and selective search with different document allocation policies. Dataset: GOV2. (T: Number of shards searched per query. K: Number of shards the dataset was partitioned into. The underlined values are not statistically different from the corresponding exhaustive search values, using paired T-test ($p < 0.01$). The values in brackets report improvement in search efficiency with selective search.)

Search Method	T/K	Search Effectiveness					Search Cost	
		P@10	P@30	P@100	NDCG@100	MAP	C_{Total} (million)	$C_{Latency}$ (million)
Exhaustive	1/1	0.53	0.48	0.38	0.42	0.29	2.89	2.89
Sel-Random	1/50	0.21	0.10	0.03	0.06	0.01	0.07 (98%)	0.07 (98%)
Sel-Source	1/50	0.34	0.25	0.15	0.18	0.06	0.11 (96%)	0.11 (96%)
Sel-SB K-means	1/50	0.39	0.32	0.23	0.25	0.13	0.19 (93%)	0.19 (93%)
Sel-SB^2 K-means	1/50	0.38	0.31	0.23	0.25	0.13	0.13 (96%)	0.13 (96%)
Exhaustive	5/5	0.53	0.48	0.38	0.42	0.29	2.89	0.58
Sel-Random	5/50	0.38	0.27	0.13	0.17	0.04	0.36 (88%)	0.07 (88%)
Sel-Source	5/50	0.48	0.41	0.29	0.32	0.16	0.50 (83%)	0.15 (74%)
Sel-SB K-means	5/50	0.53	0.47	0.36	0.40	0.25	0.84 (71%)	0.30 (48%)
Sel-SB^2 K-means	5/50	0.52	0.46	0.35	0.39	0.24	0.57 (80%)	0.19 (67%)
Exhaustive	10/10	0.53	0.48	0.38	0.42	0.29	2.89	0.29
Sel-Random	10/50	0.44	0.33	0.19	0.23	0.07	0.72 (75%)	0.07 (76%)
Sel-Source	10/50	0.52	0.45	0.33	0.37	0.21	0.94 (67%)	0.17 (41%)
Sel-SB K-means	10/50	0.53	0.48	0.37	0.42	0.27	1.38 (52%)	0.32 (-10%)
Sel-SB^2 K-means	10/50	0.52	0.47	0.37	0.41	0.27	1.01 (65%)	0.20 (31%)
Exhaustive	15/15	0.53	0.48	0.38	0.42	0.29	2.89	0.19
Sel-Random	15/50	0.46	0.37	0.24	0.28	0.09	1.07 (63%)	0.07 (63%)
Sel-Source	15/50	0.53	0.47	0.35	0.39	0.24	1.36 (53%)	0.19 (0%)
Sel-SB K-means	15/50	0.53	0.48	0.37	0.42	0.28	1.68 (42%)	0.31 (-63%)
Sel-SB^2 K-means	15/50	0.52	0.48	0.37	0.42	0.28	1.39 (52%)	0.20 (-5%)
Exhaustive	20/20	0.53	0.48	0.38	0.42	0.29	2.89	0.14
Sel-Random	20/50	0.49	0.40	0.28	0.31	0.12	1.44 (50%)	0.07 (50%)
Sel-Source	20/50	0.54	0.48	0.36	0.41	0.26	1.68 (42%)	0.20 (-43%)
Sel-SB K-means	20/50	0.53	0.48	0.38	0.42	0.28	1.87 (35%)	0.30 (-114%)
Sel-SB^2 K-means	20/50	0.53	0.48	0.37	0.42	0.28	1.72 (40%)	0.21 (-50%)

When studying selective search's performance for individual evaluation metrics we see that very few top shards need to be searched (5 for GOV2, and 1 for ClueWeb09 datasets) to be competitive at early ranks, as is illustrated by the trends for P@10 and P@30 metrics. To optimize the performance at deeper ranks (P@100 and NDCG@100), selective search has to search more shards (10 for GOV2, 5 for ClueWeb09-B, and merely 1 for ClueWeb09-English). The MAP metric, which is the most recall-oriented metric of those studied in this work, needs the relatively largest search budget for all the three datasets (20 for GOV2, 10 for ClueWeb09-B, and 5 for ClueWeb09-English).

As compared to the total search cost ($Cost_{Total}$) of exhaustive search, the selective search costs are substantially lower for all the datasets. For GOV2 the largest improvement in search efficiency is 80% at $T = 5$ for P@10 metric, and the smallest improvement is 40% at $T = 20$ for MAP. The improvements are much higher for the larger datasets. 97% for ClueWeb09-B at $T = 1$ for P@10 metric, and 77% at $T = 10$ for MAP. For ClueWeb09-English the largest improvement is 99% at $T = 1$ for P@10, and the smallest improvement is 96% at $T = 5$ for MAP. Even with the additional cost of

Table IV: Results for exhaustive search and selective search with different document allocation policies. Dataset: ClueWeb09-B. (T: Number of shards searched per query. K: Number of shards the dataset was partitioned into. The underlined values are not statistically different from the corresponding exhaustive search values, using paired T-test ($p < 0.01$). The values in brackets report improvement in search efficiency with selective search.)

Search Method	T/K	Search Effectiveness					Search Cost	
		P@10	P@30	P@100	NDCG@ 100	MAP	C_{Total} (million)	$C_{Latency}$ (million)
Exhaustive	1/1	0.23	0.21	0.16	0.25	0.16	6.25	6.25
Sel-Random	1/100	0.05	0.02	0.01	0.02	0.01	0.09 (99%)	0.09 (99%)
Sel-Source	1/100	0.11	0.06	0.02	0.05	0.02	0.10 (98%)	0.10 (98%)
Sel-SB K-means	1/100	0.22	0.18	0.12	0.19	0.11	0.24 (96%)	0.24 (96%)
Sel-SB^2 K-means	1/92	0.22	0.19	0.13	0.19	0.11	0.19 (97%)	0.19 (97%)
Exhaustive	5/5	0.23	0.21	0.16	0.25	0.16	6.25	1.25
Sel-Random	5/100	0.12	0.06	0.02	0.06	0.01	0.47 (92%)	0.09 (93%)
Sel-Source	5/100	0.19	0.12	0.06	0.11	0.04	0.50 (92%)	0.11 (91%)
Sel-SB K-means	5/100	0.24	0.21	0.16	0.24	0.15	0.94 (85%)	0.32 (74%)
Sel-SB^2 K-means	5/92	0.23	0.21	0.16	0.24	0.15	0.79 (87%)	0.23 (82%)
Exhaustive	10/10	0.23	0.21	0.16	0.25	0.16	6.25	0.63
Sel-Random	10/100	0.15	0.10	0.04	0.09	0.03	0.95 (85%)	0.10 (84%)
Sel-Source	10/100	0.22	0.16	0.08	0.14	0.06	1.00 (84%)	0.12 (81%)
Sel-SB K-means	10/100	0.24	0.21	0.16	0.24	0.15	1.63 (74%)	0.34 (46%)
Sel-SB^2 K-means	10/92	0.23	0.21	0.16	0.24	0.15	1.42 (77%)	0.24 (62%)
Exhaustive	15/15	0.23	0.21	0.16	0.25	0.16	6.25	0.42
Sel-Random	15/100	0.17	0.11	0.06	0.11	0.03	1.42 (77%)	0.10 (76%)
Sel-Source	15/100	0.22	0.17	0.09	0.15	0.07	1.48 (76%)	0.12 (71%)
Sel-SB K-means	15/100	0.24	0.21	0.16	0.24	0.15	2.25 (64%)	0.36 (14%)
Sel-SB^2 K-means	15/92	0.23	0.21	0.16	0.25	0.15	2.00 (68%)	0.24 (43%)
Exhaustive	20/20	0.23	0.21	0.16	0.25	0.16	6.25	0.31
Sel-Random	20/100	0.18	0.13	0.07	0.13	0.04	1.90 (70%)	0.10 (68%)
Sel-Source	20/100	0.23	0.18	0.11	0.17	0.08	2.00 (78%)	0.13 (58%)
Sel-SB K-means	20/100	0.23	0.21	0.16	0.24	0.15	2.83 (55%)	0.36 (-16%)
Sel-SB^2 K-means	20/92	0.23	0.21	0.16	0.25	0.16	2.55 (59%)	0.24 (23%)

ranking shards, the selective search approach needs less search effort than exhaustive search.

In case of exhaustive search, the cost in terms of latency decreases nearly linearly as more number of shards (T) are used to process the query in parallel. In contrast, for selective search the latency slowly increases as more shards are searched. There are two reasons for this trend. First, is that the cost of searching the CSI is independent of the number of shards searched, and thus remains constant even if the query processing is distributed across more shards. Second, as more shards are processed the probability of encountering a slow shard (one with large number of candidate documents) increases. Even with these challenges, the latency cost for selective search is substantially lower than that of exhaustive search for the ClueWeb09 datasets. For the smaller dataset, GOV2, selective search supports better retrieval latency than exhaustive search for all metrics except MAP. This is not surprising since selective search processes 40% of the shards to optimize for MAP, and also has the overhead of the shard ranking step.

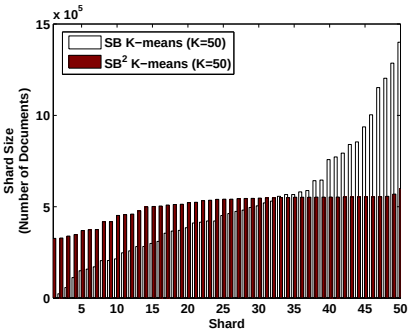
For the largest dataset, ClueWeb09-English, selective search is more effective than exhaustive search for many of the metrics, and some of the improvements with SB K-

Table V: Results for exhaustive search and selective search with different document allocation policies. Dataset: ClueWeb09-English. (T: Number of shards searched per query. K: Number of shards the dataset was partitioned into. The underlined values are not statistically different from the corresponding exhaustive search values, using paired T-test ($p < 0.01$). The values in brackets report improvement in search efficiency with selective search.)

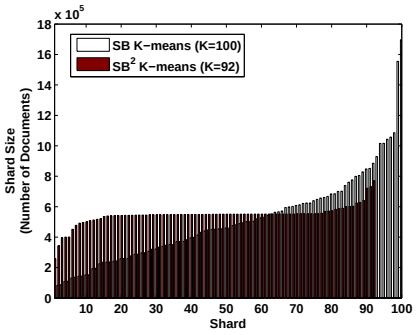
Search Method	T/K	Search Effectiveness					Search Cost	
		P@10	P@30	P@100	NDCG@100	MAP	C_{Total} (million)	$C_{Latency}$ (million)
Exhaustive	1/1	0.10	0.11	0.10	0.11	0.06	57.38	57.38
Sel-Random	1/1000	0.01	0.01	0.00	0.00	0.00	0.34 (99%)	0.34 (99%)
Sel-Source	1/1000	0.03	0.02	0.01	0.01	0.00	0.35 (99%)	0.35 (99%)
Sel-SB K-means	1/1000	0.13	0.12	0.08	0.10	0.05	0.76 (99%)	0.76 (99%)
Sel-SB^2 K-means	1/807	0.11	0.11	0.08	0.09	0.04	0.58 (99%)	0.58 (99%)
Exhaustive	5/5	0.10	0.11	0.10	0.11	0.06	57.38	11.48
Sel-Random	5/1000	0.04	0.02	0.01	0.01	0.00	1.72 (97%)	0.34 (97%)
Sel-Source	5/1000	0.07	0.04	0.02	0.03	0.01	1.73 (97%)	0.35 (97%)
Sel-SB K-means	5/1000	0.12	0.12	0.10	0.11	0.06	3.19 (94%)	1.10 (90%)
Sel-SB^2 K-means	5/807	0.12	0.12	0.10	0.11	0.06	2.53 (96%)	0.74 (94%)
Exhaustive	10/10	0.10	0.11	0.10	0.11	0.06	57.38	5.74
Sel-Random	10/1000	0.06	0.03	0.01	0.02	0.00	3.46 (94%)	0.35 (94%)
Sel-Source	10/1000	0.09	0.05	0.02	0.04	0.01	3.48 (94%)	0.36 (94%)
Sel-SB K-means	10/1000	△0.12	△0.12	0.10	0.12	0.06	5.73 (90%)	1.25 (78%)
Sel-SB^2 K-means	10/807	0.11	0.12	0.10	0.11	0.06	4.81 (92%)	0.81 (86%)
Exhaustive	15/15	0.10	0.11	0.10	0.11	0.06	57.38	3.83
Sel-Random	15/1000	0.06	0.03	0.01	0.02	0.01	5.19 (91%)	0.35 (91%)
Sel-Source	15/1000	0.09	0.06	0.03	0.04	0.01	5.22 (91%)	0.36 (91%)
Sel-SB K-means	15/1000	△0.12	△0.12	0.10	0.12	0.06	8.12 (86%)	1.43 (63%)
Sel-SB^2 K-means	15/807	0.11	0.11	0.10	0.11	0.06	6.96 (88%)	0.88 (77%)
Exhaustive	20/20	0.10	0.11	0.10	0.11	0.06	57.38	2.87
Sel-Random	20/1000	0.07	0.04	0.02	0.03	0.01	6.92 (88%)	0.35 (88%)
Sel-Source	20/1000	0.10	0.06	0.03	0.05	0.01	6.99 (88%)	0.37 (87%)
Sel-SB K-means	20/807	△0.12	△0.12	0.10	0.12	0.06	10.48 (82%)	1.48 (48%)
Sel-SB^2 K-means	20/807	0.11	0.12	0.10	0.12	0.06	8.97 (84%)	0.93 (68%)

means shards are statistically significant. This result suggests that topical partitioning of the collection is able to reduce the false positives from the search space, which might be especially helpful for a noisy and heterogeneous datasets like ClueWeb09-English. Such improvements in search effectiveness are not observed for the other two datasets that have less noise, are well-curated, and homogenous. Recall that GOV2 documents were obtained from government related websites (homogenous), and the ClueWeb09-B dataset is 12% Wikipedia documents, most of which are well-curated, and lower in spam.

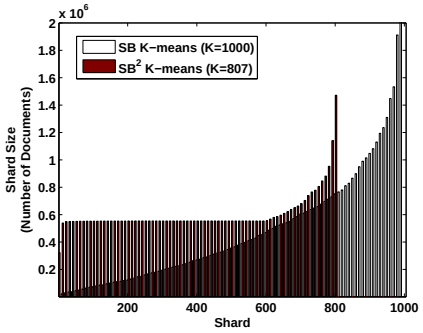
Overall, these results indicate that the selective search approach with topic-based shards provides a cost-effective alternative to the exhaustive search approach. Depending upon the type of search task of interest (precision-oriented or recall-oriented), the improvement in search efficiency can vary. However, even the smallest reduction in total search cost offered by selective search is substantial. For smaller datasets, selective search is well-suited for batch query processing environments where query response time is not crucial. For larger datasets, however, selective search is the best choice for both, batch and interactive query processing. In general, selective search is



(a) Dataset: GOV2.



(b) Dataset: ClueWeb09-B.



(c) Dataset: ClueWeb09-English.

Fig. 3: Shard size distribution for topic-based shards created with SB K-means and SB^2 K-means algorithms.

especially well-suited for large and heterogeneous datasets, but can also provide an efficient and effective search solution for small and homogenous datasets.

6.1. Topic-based Policies: Shard Sizes

This section investigates the efficacy of the topic-based algorithms at creating shards with uniform size distribution. Figure 3 compares the sizes of the shards created by SB K-means and SB^2 K-means with the expected shard sizes for the three datasets. Recall, that for each dataset the total number of shards (K) was chosen such that the average shard size would be half a million documents (Section 5.3). Including a margin of 10%, the targeted size range is [0.45M, 0.55M] documents. About 75% of the shards created with SB^2 K-means are in the targeted size range for all three datasets. In contrast, only about 15% of the SB K-means shards are in this range.

The plots illustrate that the shards created by SB^2 K-means have less variance in their sizes than those created by SB K-means. This is important for two reasons: computational load-balancing, and query evaluation latency. When the shard sizes are comparable, the utilization of computational nodes deployed for query evaluation is balanced. Secondly, the longest execution path for a query, which determines the search cost latency, has tighter bounds with uniform shard sizes than with skewed shard sizes. This is evidenced by the search cost latency results in Tables III, IV, and V,

where the SK^2 K-means shards provide consistently better cost latency than the SB K-means shards.

Another important trend in these plots is the positive correlation between the collection size and the fraction of large shards created by SB^2 K-means – more large shards are created for larger datasets. This suggests a useful future improvement to the SB^2 K-means. Instead of using a fixed number of iterations for the *split* phase of SB^2 K-means algorithm, a strategy that dynamically decides the number of iterations based on the collection size or the fraction of remaining large shards, could be more effective at reducing large shards.

6.2. Robustness Analysis

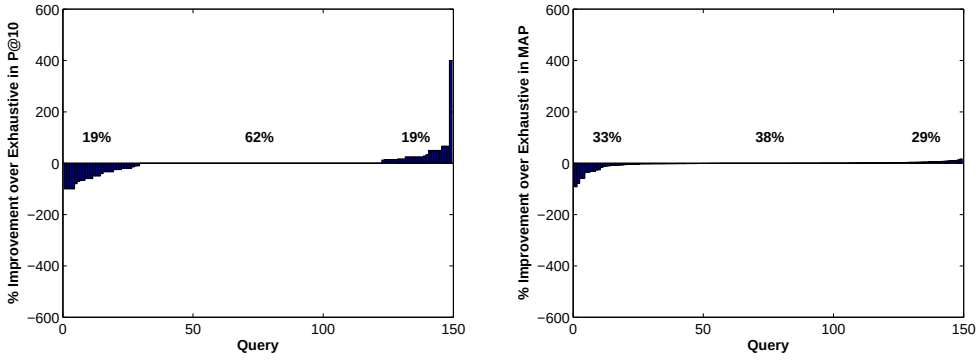
This section investigates the stability of selective search's effectiveness at the individual query level. Figure 4 reports the improvement in P@10 and MAP with selective search for each evaluation query. A two-fold cross validation framework was used to choose the number of top shards searched (T) for each dataset. The T value tuned on one half of the query-set was used for the remaining queries (test-set). The tuning process chose the smallest T value for which the metric value (P@10 or MAP), averaged over all queries in the fold, for selective search was on par with that of exhaustive search. This framework searched the top 5.5 shards on average for GOV2, and the top 2 shards for both the ClueWeb09 datasets when tuning for P@10.

The plots in the left column of Figure 4, the P@10 plots, illustrate that for all three datasets, at least 80% of the queries are as effective or better with selective search than exhaustive search. About 60% of the queries have the exact same effectiveness with both the search approaches, and 20% of the queries improve with selective search. The magnitude of improvement is larger for the bigger datasets, underscoring the benefits of using selective search for large, and heterogeneous datasets. About 20% of the queries degrade with selective search. These errors are caused by one of three reasons: partitioning error (relevant documents spread across many shards), shard ranking error, or failure to retrieve relevant documents at top ranks. Each of these error types motivates a worthwhile future research direction.

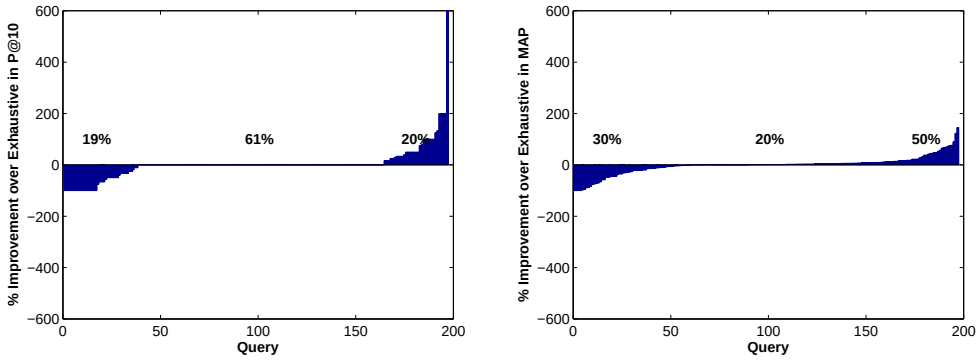
The two-fold cross-validation setup when tuning for MAP metric, chose to search the top 16.5 shards on average for GOV2, top 10 for ClueWeb09-B, and top 9 for ClueWeb09-English. Due to its recall-oriented nature, the MAP metric is harder to optimize for which is reflected in the fact that more shards need to be searched. Also, the formulation of the MAP metric makes it more sensitive to small changes in the ranking of relevant documents than the P@n metrics. As a result, reproducing the exact values as that with exhaustive search is harder for MAP. Still, at least 60% of the queries perform better or just as well with selective search, as is illustrated by the MAP plots (right column) in Figure 4. For the ClueWeb09 datasets half of the queries improve with selective search.

6.3. Distribution of Relevant Documents

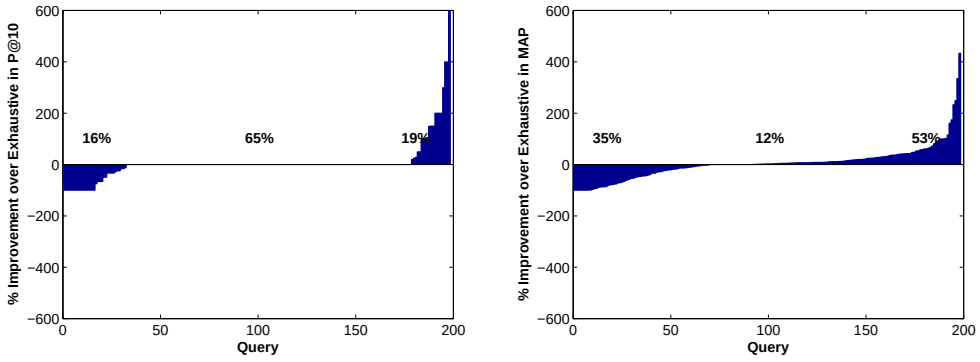
In this section we investigate the main cause for the stark difference in selective search performance with the three types of shards: random, source-based, and topic-based (SB^2 K-means). Selective search's ability to balance effectiveness and efficiency is dependent on the distribution of the relevant documents across shards. If the relevant documents for a query are spread across many shards then more shards need to be searched to be effective, which increases the cost. In contrast, if the relevant documents are concentrated into a few shards then a small search budget is sufficient. Figure 5 illustrates the spread of the relevant documents for the three types of shards, and for the three datasets. For every query, the fraction of its relevant documents in



(a) GOV2

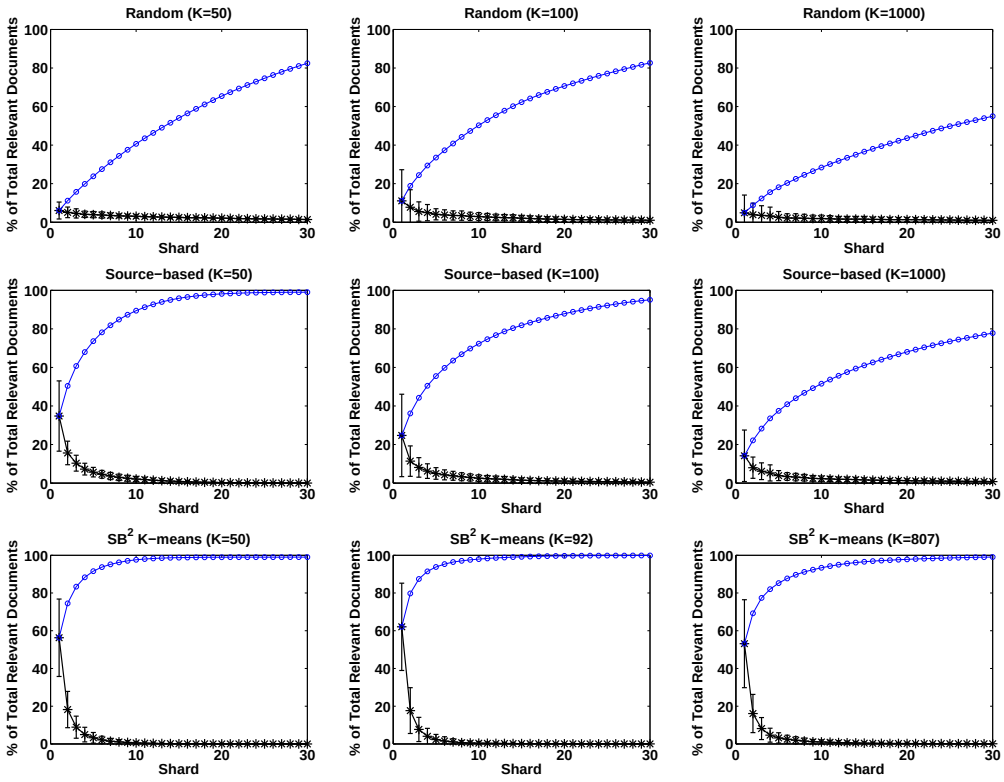


(b) ClueWeb09-B



(c) ClueWeb09-English

Fig. 4: Comparison of selective search and exhaustive search at individual query level, in terms of P@10 (left column) and MAP (right column).



(a) GOV2 (b) ClueWeb09-B (c) ClueWeb09-English

Fig. 5: Distribution of relevant documents across shards.

each shard was computed, and the shards were sorted on these relevance scores (descending order). This is referred to as the relevance-based ranking (RBR) of shards. The average of the relevance scores over all the queries is reported for the top 30 shards in the plots.

The relevant documents are spread almost uniformly across the random shards. The source-based shards have better concentration of relevant documents than random. However, topic-based shards exhibit the most skewed distribution of relevant documents. These trends are consistent across the three datasets. The topical shard with the largest percentage of relevant documents contains almost twice as many relevant documents as the topmost source-based shard, and more than thrice as many relevant documents as the topmost random shard. At least half of the relevant documents for a query are situated in a single topical shard. The top 3 topical shards together contain more than 80% of the relevant documents. These observations motivate the analysis reported in Section 7, where we evaluate selective search with relevance-based shard ranking, instead of ReDDE, for estimating the ranking of shards for a query.

We believe that selective search provides competitive effectiveness, especially at deeper ranks (P@100, NDCG@100, and MAP), because of the ability of topical shards to generate such skewed distributions of relevant documents. This is one of the unique strengths of selective search. Most of the previous work in distributed IR and feder-

ated search has either rarely evaluated with recall-oriented metrics, or has struggled to demonstrate competitive performance at deeper ranks.

These plots also suggest a few interesting future directions. The large error bars indicate high variability in the allocation policy's ability to concentrate relevant documents for different queries. Secondly, although concentrating 60% of the relevant documents is a good start, a more skewed distribution with a higher percentage of relevant documents into fewer shard would be better. Thus an improved allocation policy that provides more consistent performance across queries, and that can provide higher concentration of relevant documents would be worth pursuing. For instance, a soft clustering technique that can allocate a document to more than one shard based on the topical composition of the document would be worth studying.

6.4. Overlap-based Evaluation and Effects of Query Length

The goal of the analysis presented in this section is two-fold. First, we test selective search performance using larger query sets than those used in the prior sections. Second, we investigate the effect of query length on selective search performance. The number of terms in the query directly influences the evaluation effort needed for the query. Existing query optimization approaches are known to be sensitive to query length. For instance, some approaches report higher efficiency gains for longer queries (7 words) than shorter queries (2.5 words) [Broder et al. 2003; Smeaton and van Rijsbergen 1981; Wong and Lee 1993]. Our goal is to study how sensitive selective search is to query length.

For this analysis we employ *overlap*-based evaluation methodology where the top n documents retrieved by exhaustive search are assumed to be relevant to the query. Selective search's effectiveness for the query is then computed relative to these pseudo-relevant documents. One might also say that the overlap evaluation assumes that the goal of selective search is to *replicate* (but not improve upon) exhaustive search. In our experiments we assume that the top 10 documents retrieved by exhaustive search for a query are relevant, and thus evaluate selective search's effectiveness using P@10 metric.

The overlap-based evaluation does not require human relevance judgments and thus allows experimentation with much larger query sets than TREC query sets. We experiment with the GOV2 and ClueWeb09-B datasets for this analysis. A set of 1195 queries was sampled from the AOL query log which was used with the GOV2 dataset. This set consists of queries of varying length, 1 word through 10 words. The sizes for some of the query length based bins are given in Table VI, for the remaining bins the sizes are: 4 words 178 queries, 6 words 103 queries, 8 words 83 queries, 9 words 41 queries, and 10 words 12 queries. For the ClueWeb09-B dataset 1002 queries were sampled from the query set used for the TREC 2009 Million Query Track[6]. This set has a fewer range of query lengths, 1 word through 7 words. Apart from the bin sizes reported in Table VII, this set consists of 200 queries of 4 words, 38 queries of 6 words, and 5 queries of 7 words.

Selective search was performed with each of the three shard types, random, source-based, and topic-based (SB^2 K-means). The top 5 shards were searched for each query. A comparable setup was employed for exhaustive search, where each dataset was partitioned into 5 random shards, and all 5 shards were searched for each query. The results for the two datasets are presented in Tables VI and VII. In order to study the impact of query length on search performance, the queries were binned based on their length, and the results for some of the query length categories are also reported in Tables VI and VII. The results for the bins with fewer queries than 100 are not reported.

[6]http://ir.cis.udel.edu/million/

Table VI: Results for overlap-based evaluation of selective search, and impact of query length. and Dataset: GOV2. (Number of top shards searched (T): 5.)

Query Length	No. Of Queries	Search Method	P@10	C_{Total}	$C_{Latency}$
All	1195	Exhaustive	1.00	7.68	1.54
		Random	0.11	0.99 (87%)	0.20 (87%)
		Source-based	0.48	1.23 (84%)	0.32 (79%)
		Topic-based	0.75	1.28 (83%)	0.34 (78%)
1	129	Exhaustive	1.00	0.28	0.06
		Random	0.08	0.06 (79%)	0.01 (83%)
		Source-based	0.41	0.11 (61%)	0.04 (33%)
		Topic-based	0.61	0.13 (54%)	0.05 (17%)
2	191	Exhaustive	1.00	3.28	0.66
		Random	0.11	0.42 (87%)	0.08 (88%)
		Source-based	0.54	0.56 (83%)	0.17 (74%)
		Topic-based	0.70	0.60 (82%)	0.19 (71%)
3	185	Exhaustive	1.00	7.37	1.47
		Random	0.11	0.92 (88%)	0.18 (88%)
		Source-based	0.66	1.14 (85%)	0.30 (80%)
		Topic-based	0.77	1.20 (84%)	0.33 (78%)
5	136	Exhaustive	1.00	9.57	1.91
		Random	0.11	1.23 (87%)	0.25 (87%)
		Source-based	0.64	1.51 (84%)	0.38 (80%)
		Topic-based	0.77	1.61 (83%)	0.41 (79%)
7	137	Exhaustive	1.00	10.48	2.10
		Random	0.12	1.34 (87%)	0.27 (87%)
		Source-based	0.61	1.66 (84%)	0.41 (80%)
		Topic-based	0.77	1.71 (84%)	0.44 (79%)

It is important to not read too much into the search effectiveness results when using the overlap-based evaluation since in reality it is rarely the case that all the top n documents retrieved by exhaustive search are relevant. However, these results are reliable indicators of the general trends. For instance, the main conclusions from the prior sections are re-confirmed by these results: the topic-based shards provide the best balance between effectiveness and efficiency, and selective search of topic-based shards is substantially more efficiency than exhaustive search.

One of the new insights from these results is that selective search provides fairly consistent improvement in efficiency for queries with 2 or more terms. However, for single term queries the efficiency gains can be lower, especially for smaller datasets. Overall, this analysis suggests that selective search's performance is not highly sensitive to query length, and it also reconfirms the efficiency improvements offered by selective search using a larger query set.

6.5. Effect of CSI Size

For sample-based resource ranking algorithms such as ReDDE, the size of the centralized sample index (CSI), is an important parameter. An overly small CSI is prone to under-sampling errors (one or more query terms are not present in the CSI) which lead to shard ranking errors, thus degrading search effectiveness. On the other hand a large CSI can increase the cost of the shard ranking process which degrades search efficiency. In this section we experiment with a range of CSI sizes to study the effect of this parameter on selective search performance. Although prior work in federated search has investigated the influence of this parameter, the impact on search efficiency was

Table VII: Results for overlap-based evaluation of selective search, and impact of query length. Dataset: ClueWeb09-B. (Number of top shards searched (T): 5.)

Query Length	No. Of Queries	Search Method	P@10	C_{Total}	$C_{Latency}$
All	1002	Exhaustive	1.00	9.63	1.93
		Random	0.05	0.76 (92%)	0.15 (92%)
		Source-based	0.15	0.80 (92%)	0.17 (91%)
		Topic-based	0.74	1.13 (88%)	0.30 (84%)
1	159	Exhaustive	1.00	0.74	0.15
		Random	0.05	0.08 (89%)	0.02 (87%)
		Source-based	0.13	0.08 (89%)	0.02 (87%)
		Topic-based	0.60	0.16 (78%)	0.05 (67%)
2	200	Exhaustive	1.00	3.85	0.77
		Random	0.05	0.29 (92%)	0.06 (92%)
		Source-based	0.13	0.30 (92%)	0.07 (91%)
		Topic-based	0.71	0.50 (87%)	0.16 (79%)
3	200	Exhaustive	1.00	9.89	1.98
		Random	0.06	0.74 (93%)	0.15 (92%)
		Source-based	0.15	0.77 (92%)	0.17 (91%)
		Topic-based	0.77	1.12 (89%)	0.31 (84%)
5	200	Exhaustive	1.00	16.49	3.30
		Random	0.06	1.24 (92%)	0.25 (92%)
		Source-based	0.16	1.30 (92%)	0.29 (91%)
		Topic-based	0.82	1.77 (89%)	0.45 (86%)

rarely studied [Callan 2000; Shokouhi and Si 2011]. Also, we experiment with much smaller CSI sizes than those used and recommended in prior work. Our motivation comes from the observation that each topic-based shard has a *core vocabulary* that consists of terms that are central to the topic and thus occur frequently in the shard's documents. We thus conjecture that even a small sample from the shard can uncover the core vocabulary of the shard, and thus support effective shard ranking.

Table VIII reports selective search effectiveness and efficiency when employing CSI of different sizes. These results demonstrate that as the CSI size progressively reduces, the selective search effectiveness degrades. However, the rate of degradation is slow, especially for the larger datasets and at early ranks (P@10, P@30 and P@100). The reduction in CSI size provides a steady but sub-linear reduction in search cost. This is expected because smaller CSI offers cost savings only in the shard ranking phase of the selective search pipeline, and not in the shard searching phase.

Since ClueWeb09-A is partitioned into 807 topical shards, each shard is about 0.12% of the collection. In the above experiments where the top 5 shards are searched, about 0.6% of the collection is processed in the shard search phase. As a result, the total search cost is dominated by the cost of searching CSI for the settings where CSI size is greater than 0.6%. In contrast, for GOV2 dataset where the average shard size is 2% of the collection, the shard search phase processes 10% of the collection, and the total search cost is dominated by shard searching. As a result, the savings in search costs with smaller CSI sizes is less pronounced for GOV2 than ClueWeb09-A. Overall, we chose the CSI size of 0.5% for our experiments because although 8 times smaller than the 4% sample, it supports selective search that is just as effective but at least twice as efficient, for all three datasets.

Table VIII: Effect of CSI size on selective search effectiveness and efficiency. Number of top shards searched (T): 5. $\bigtriangledown$ denotes significantly worse effectiveness than with 4% CSI, using paired T-test ($p < 0.01$).

(a) GOV2

CSI size (%)	Search Effectiveness					Search Cost	
	P@10	P@30	P@100	NDCG@100	MAP	C_{Total} (million)	$C_{Latency}$ (million)
4.00	0.53	0.47	0.36	0.40	0.25	1.07	0.28
2.00	0.52	0.47	0.36	0.40	0.25	0.79	0.22
1.00	0.52	0.47	0.36	0.40	0.25	0.65	0.20
0.50	0.52	0.46	0.35	0.39	0.24	0.57	0.19
0.25	$\bigtriangledown$0.49	$\bigtriangledown$0.45	$\bigtriangledown$0.34	$\bigtriangledown$0.38	$\bigtriangledown$0.24	0.53	0.18
0.10	$\bigtriangledown$0.49	$\bigtriangledown$0.43	$\bigtriangledown$0.33	$\bigtriangledown$0.37	$\bigtriangledown$0.21	0.51	0.18
0.05	$\bigtriangledown$0.49	$\bigtriangledown$0.43	$\bigtriangledown$0.32	$\bigtriangledown$0.36	$\bigtriangledown$0.21	0.47	0.17

(b) ClueWeb09-B

CSI size (%)	Search Effectiveness					Search Cost	
	P@10	P@30	P@100	NDCG@100	MAP	C_{Total} (million)	$C_{Latency}$ (million)
4.00	0.23	0.21	0.16	0.24	0.15	12.58	2.74
2.00	0.23	0.21	0.16	0.24	0.15	6.84	1.58
1.00	0.23	0.21	0.15	0.24	0.15	4.00	1.03
0.50	0.23	0.21	0.16	0.24	0.15	2.53	0.74
0.25	0.22	0.21	0.15	0.23	0.14	1.78	0.58
0.10	0.21	0.19	0.15	0.22	$\bigtriangledown$0.13	1.34	0.51
0.05	0.21	0.18	$\bigtriangledown$0.13	$\bigtriangledown$0.20	$\bigtriangledown$0.12	1.15	0.49

(c) ClueWeb09-English

CSI size (%)	Search Effectiveness					Search Cost	
	P@10	P@30	P@100	NDCG@100	MAP	C_{Total} (million)	$C_{Latency}$ (million)
4.00	0.23	0.21	0.16	0.24	0.15	12.58	2.74
2.00	0.23	0.21	0.16	0.24	0.15	6.84	1.58
1.00	0.23	0.21	0.15	0.24	0.15	4.00	1.03
0.50	0.23	0.21	0.16	0.24	0.15	2.53	0.74
0.25	0.22	0.21	0.15	0.23	0.14	1.78	0.58
0.10	0.21	0.19	0.15	0.22	$\bigtriangledown$0.13	1.34	0.51
0.05	0.21	0.18	$\bigtriangledown$0.13	$\bigtriangledown$0.20	$\bigtriangledown$0.12	1.15	0.49

7. SELECTIVE SEARCH WITH RELEVANCE-BASED RANKING

Table IX reports the results for a selective search experiment that employed relevance-based ranking of shards (RBR) instead of ReDDE. Relevance-based ranking sorts the topical shards based on the fraction of relevant documents for the query present in each shard. Since this ranking approach uses the query relevance judgments, it is an oracle experiment that provides an upper-bound on selective search performance.

In these results searching the single best topical shard outperforms exhaustive search significantly for most of the metrics. Exhaustive search has access to 100% of the relevant documents while selective search has access only to 60% of the relevant documents on average when only the top shard is searched (Section 6.3). Yet, the

Table IX: Results for selective search with relevance-based shard ranking. T: Number of shards searched per query. K: Number of shards the dataset was partitioned into. The underlined values are not statistically different from the corresponding exhaustive search values, using paired T-test ($p < 0.01$). $\triangle$ indicates statistically significant higher value than the corresponding exhaustive search value.

(a) GOV2

Search Method	T/K	Search Effectiveness					Search Cost	
		P@10	P@30	P@100	NDCG@100	MAP	C_{Total}	$C_{Latency}$
Exhaustive	1/1	0.53	0.48	0.38	0.42	0.29	2.89	2.89
Exhaustive	3/3	0.53	0.48	0.38	0.42	0.29	2.89	0.96
Exhaustive	5/5	0.53	0.48	0.38	0.42	0.29	2.89	0.58
Sel & RBR	1/50	$\triangle$0.61	$\triangle$0.52	0.38	$\triangle$0.43	0.24	0.13 (96%)	0.13 (96%)
Sel & RBR	3/50	$\triangle$0.58	$\triangle$0.52	$\triangle$0.40	$\triangle$0.45	0.30	0.35 (88%)	0.16 (83%)
Sel & RBR	5/50	0.57	$\triangle$0.52	$\triangle$0.40	$\triangle$0.45	$\triangle$0.31	0.54 (81%)	0.17 (71%)
Sel & ReDDE	1/50	0.38	0.31	0.23	0.25	0.13	0.13 (96%)	0.13 (96%)
Sel & ReDDE	3/50	0.50	0.44	0.33	0.37	0.22	0.37 (87%)	0.17 (82%)
Sel & ReDDE	5/50	0.52	0.46	0.35	0.39	0.24	0.57 (80%)	0.19 (67%)

(b) ClueWeb09-B

Search Method	T/K	Search Effectiveness					Search Cost	
		P@10	P@30	P@100	NDCG@100	MAP	C_{Total}	$C_{Latency}$
Exhaustive	1/1	0.23	0.21	0.16	0.25	0.16	6.25	6.25
Exhaustive	3/3	0.23	0.21	0.16	0.25	0.16	6.25	2.08
Exhaustive	5/5	0.23	0.21	0.16	0.25	0.16	6.25	1.25
Sel & RBR	1/92	$\triangle$0.32	$\triangle$0.29	0.18	$\triangle$0.32	$\triangle$0.19	0.18 (97%)	0.18 (97%)
Sel & RBR	3/92	$\triangle$0.31	$\triangle$0.29	$\triangle$0.18	$\triangle$0.34	$\triangle$0.22	0.47 (93%)	0.21 (90%)
Sel & RBR	5/92	$\triangle$0.30	$\triangle$0.27	$\triangle$0.20	$\triangle$0.34	$\triangle$0.23	0.70 (89%)	0.21 (83%)
Sel & ReDDE	1/92	0.22	0.19	0.13	0.19	0.11	0.19 (97%)	0.19 (97%)
Sel & ReDDE	3/92	0.23	0.20	0.15	0.22	0.13	0.51 (92%)	0.22 (89%)
Sel & ReDDE	5/92	0.23	0.21	0.16	0.24	0.15	0.79 (87%)	0.23 (82%)

(c) ClueWeb09-English

Search Method	T/K	Search Effectiveness					Search Cost	
		P@10	P@30	P@100	NDCG@100	MAP	C_{Total}	$C_{Latency}$
Exhaustive	1/1	0.10	0.11	0.10	0.11	0.06	57.38	57.38
Exhaustive	5/5	0.10	0.11	0.10	0.11	0.06	57.38	19.12
Exhaustive	5/5	0.10	0.11	0.10	0.11	0.06	57.38	11.48
Sel & RBR	1/807	$\triangle$0.23	$\triangle$0.22	$\triangle$0.15	$\triangle$0.20	$\triangle$0.10	0.55 (99%)	0.55 (99%)
Sel & RBR	3/807	$\triangle$0.18	$\triangle$0.18	$\triangle$0.15	$\triangle$0.19	$\triangle$0.11	1.50 (97%)	0.64 (97%)
Sel & RBR	5/807	$\triangle$0.17	$\triangle$0.18	$\triangle$0.14	$\triangle$0.19	$\triangle$0.11	2.28 (96%)	0.69 (94%)
Sel & ReDDE	1/807	0.11	0.11	0.08	0.09	0.04	0.58 (99%)	0.58 (99%)
Sel & ReDDE	3/807	0.12	0.12	0.10	0.11	0.05	1.57 (97%)	0.66 (97%)
Sel & ReDDE	5/807	0.12	0.12	0.10	0.11	0.06	2.53 (96%)	0.74 (94%)

Table X: Exhaustive search and selective search with and without query optimization. Improvements over exhaustive search are reported in round brackets. Improvements with query optimization are reported in square brackets.

(a) Number of top shards searched (T): 5

		Optimization=Off		Optimization=On	
		C_{Total}	$C_{Latency}$	C_{Total}	$C_{Latency}$
GOV2	Exhaustive	2.89	0.58	1.73 [40%]	0.35 [40%]
	Sel-SB^2 K-means	0.57 (80%)	0.19 (67%)	0.39 (78%) [32%]	0.14 (60%) [26%]
ClueWeb09-B	Exhaustive	6.25	1.25	4.84 [23%]	0.97 [22%]
	Sel-SB^2 K-means	0.79 (87%)	0.23 (82%)	0.64 (87%) [19%]	0.19 (80%) [17%]
ClueWeb09-English	Exhaustive	57.38	11.48	39.54 [31%]	7.91 [31%]
	Sel-SB^2 K-means	2.53 (96%)	0.74 (94%)	1.88 (95%) [26%]	0.56 (93%) [24%]

(b) Number of top shards searched (T): 10

		Optimization=Off		Optimization=On	
		C_{Total}	$C_{Latency}$	C_{Total}	$C_{Latency}$
GOV2	Exhaustive	2.89	0.29	1.73 [40%]	0.17 [41%]
	Sel-SB^2 K-means	1.01 (65%)	0.20 (31%)	0.66 (62%) [35%]	0.14 (18%) [30%]
ClueWeb09-B	Exhaustive	6.25	0.63	4.84 [23%]	0.48 [24%]
	Sel-SB^2 K-means	1.42 (77%)	0.24 (62%)	1.14 (76%) [20%]	0.20 (58%) [17%]
ClueWeb09-English	Exhaustive	57.38	5.74	39.54 [31%]	4.00 [30%]
	Sel-SB^2 K-means	4.81 (92%)	0.81 (86%)	3.53 (91%) [27%]	0.62 (85%) [23%]

latter does substantially better than the former at early ranks and deep ranks, both. This suggests that the *purity* of the search space used by the retrieval algorithm has a significant impact on the search effectiveness. On an average, the density of relevant documents in the search space of exhaustive search is $7{\cdot}10^{-6}$, $1{\cdot}10^{-6}$, and $2{\cdot}10^{-7}$ for Gov2, ClueWeb09-B, ClueWeb09-English, respectively. Whereas the relevance density of the topmost single shard is $4{\cdot}10^{-4}$, $1{\cdot}10^{-4}$, and $2{\cdot}10^{-4}$ for Gov2, ClueWeb09-B, ClueWeb09-English, respectively. As the proportion of false positive documents increases it gets harder to separate relevant from the non-relevant documents. We see further evidence in support of this hypothesis as more shards are searched: The precision at early ranks for selective search degrades. The trend is similar at deeper ranks as well, however, the pace of the degradation is slower, indicating lower sensitivity to the noise.

For reference, Table IX also reports selective search results with ReDDE. These results underscore the importance of the efficacy of the shard ranking algorithm employed by selective search. Selective search's performance with relevance-based ranking is substantially better than with ReDDE consistently, indicating that there is ample room for improving the effectiveness of the shard ranking algorithms. A worthwhile future research direction.

8. EFFECTS OF QUERY OPTIMIZATION

Dynamic index pruning approaches, commonly referred to as query optimization techniques, are routinely employed by search systems to reduce query processing costs. Term-bounded Max Score (TBMS) is one such technique (described in Section 2) that manipulates the inverted index and the query evaluation process to minimize the number of documents evaluated for the query. In this section we compare selective search with TBMS, and also investigate the collective effect of the two on search efficiency.

The experiments presented in this section study four search configurations: exhaustive search with and without TBMS, and selective search with and without TBMS. The results are reported in Table X. When analyzing the effect of query optimization on exhaustive search (values in square brackets) we see consistent and substantial improvements in efficiency for all three datasets. The improvements are higher for GOV2 (about 40%) than for ClueWeb09 datasets (about 20% or 30%). We believe this is mainly due to the longer GOV2 queries (3.1 terms) than the ClueWeb09 queries (2.1 terms). The efficacy of most query optimization techniques is known to be dependent on the length of the query [Wong and Lee 1993; Strohman et al. 2005].

The efficiency improvements with selective search (reported in round brackets) are larger than those with query optimization, for all three datasets. For instance, the C_{Total} and $C_{Latency}$ values are 87% and 83% lower with selective search, but only 23% and 22% lower with TBMS, for ClueWeb09-B. This trend holds also for the configuration with larger T value (T=10), with one exception. The $C_{Latency}$ for the smallest dataset, GOV2, is 0.20 with selective search, which is higher than that with query optimization (0.17). For the bigger datasets, however, selective search is consistently more efficient.

These results also illustrate the complementary nature of query optimization and selective search. When query optimization is applied to selective search we obtain the lowest search costs. In case of GOV2 dataset, for example, the C_{Total} with just selective search (0.57), and the C_{Total} with just query optimization (1.73), which are both higher than in the combined configuration (0.39). This trend holds for all the results. The improvements over exhaustive search with selective search are fairly stable across the two configurations of with and without TBMS. For example, C_{Total} and $C_{Latency}$ for selective search are 96% and 94% lower than exhaustive search in the non-optimized configuration, and 95% and 93% lower in the optimized configuration, for ClueWeb09-English.

9. CONCLUSIONS

This paper investigates and extends selective search, a recently proposed approach for processing large textual datasets with few computational resources. This search technique first partitions the dataset into smaller subsets (shards), and only a few of the shards, that are estimated to contain relevant documents for the query, are searched during query evaluation. As part of this research, we developed scalable, efficient and self-reliant shard creation approaches that can partition any large textual collection rapidly. An empirical analysis that compares several shard creation techniques, demonstrates that one of the proposed approaches, SB^2 K-means, supports the best selective search performance. This algorithm partitions the corpus based on i) document similarity, and ii) shard sizes. The resulting shards are topically homogeneous and exhibit lower variance in size. We show that one of the main reasons for the success of topic-based partitioning strategies is their ability to concentrate relevant documents for a query into few shards, which is necessary to support effective and efficient selective search.

Prior work demonstrated that selective search substantially improves over exhaustive search's efficiency. Our experiments, conducted using three large datasets, reconfirm this finding using a stronger baseline: a distributed exhaustive search setup that parallelizes query processing. We also improve the experimental framework by using efficiency metrics that better model the search costs than the ones used in prior work, and that also facilitate comparison of query latency. The results demonstrate that for precision-oriented tasks selective search can support 80–99% reduction in total search cost, and 67–99% reduction in latency, while assuming access to as few computational nodes as 1 or 5. For recall-oriented tasks the reductions are 77–96% in total search cost,

and 62–94% in latency for the large datasets. For the smallest dataset GOV2, selective search lowers the search cost by 40%, but degrades the latency when optimizing for MAP. Typically though recall-oriented tasks implement batch query processing where the improvements in total search cost are more important than latency.

A query-level robustness analysis for the P@10 metric shows that at least 80% of the queries perform as well or better with selective search. Even for a more sensitive metric like MAP at least 65% of the queries do better or as well with selective search. Through an oracle experiment that uses the optimal ranking of shards for a query, we demonstrate the potential of selective search to improve over exhaustive search's effectiveness. In these experiments selective search of the single best shard for a query, outperforms exhaustive search consistently on most of the effectiveness metrics. This is a surprising finding, especially since the single best shard contains only 60% of the relevant documents for a query, on an average, while exhaustive search has access to all the relevant documents. This provides an interesting insight that the purity of the search space used by the retrieval algorithm can be more important than the coverage of the relevant documents.

This research also compares selective search with term-bounded max-score (TBMS) algorithm, a widely used query optimization technique for improving search efficiency. These experiments illustrate that the improvements in efficiency with selective search are substantially higher than those with query optimization. The best configuration, however, is the one that applies query optimization to selective search, which provides the lowest search costs until now for all the datasets. This confirms the complementary strengths of the two approaches which was postulated in previous selective search work.

Overall, the investigations conducted in this paper demonstrate that selective search is a reliable alternative to exhaustive search, and is especially helpful in resource-constrained environments, where balancing search efficiency and effectiveness is crucial. This research also illustrates selective search's ability to synergize with other related techniques to further improve search performance. Several interesting future lines of research are suggested by the findings in this work. Developing shard creation techniques that increase the skew in the distribution of relevant documents across shards is one such venue. Experimenting with alternatives to the K-means clustering algorithm to partition the collection into topic-based shards could also be fruitful. The popular topic-modeling technique, latent Dirichlet allocation (LDA), and the efficient nearest-neighbor search technique, Locality-sensitive hashing (LSH), are a few such promising alternatives. Also, partitioning techniques that can exploit external knowledge sources, such as, document categories or classification hierarchies, when such data is available, could produce better topic-based shards.

Currently geographically distributed document collections are prominent only at Web-scale search. However, increasing number of medium and small-scale organizations also have to tackle the problem of searching geographically distributed collections as their operations expand globally. In such search environments the selection of shards to be searched for a query needs a joint modeling of shard relevance and geographical proximity, similar to that in geographically tiered collections [Cambazoglu et al. 2009]. The topic-based organization of the dataset can make it hard to balance the computational load on the system, especially in search environments where the popularity of topics fluctuates wildly. Load balancing techniques that are specifically designed for selective search of topic-based shards is an important future investigation.

Caching techniques have been successful at improving the efficiency of traditional search systems (review of prior research in Section 2). We believe that caching techniques and selective search have complementary strengths because they exploit dif-

ferent properties of the search process in order to improve efficiency. The central idea behind caching techniques is that of reusing prior work, while for selective search it is reducing the search space through cluster hypothesis. In addition to caching of frequently accessed posting lists, and query results, the ranking of shards estimated for frequent queries could also be cached in case of selective search. The topic-based organization used by selective search might also influence the benefits derived from caching techniques. As such, exploring the research problem of designing caching strategies for several stages of the selective search pipeline is also a promising future direction. Another interesting direction for this work is to develop shard ranking algorithms that models shard purity along with relevance.

REFERENCES

Jaime Arguello, Jamie Callan, and Fernando Diaz. 2009. Classification-based resource selection. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 1277–1286.

Ricardo Baeza-Yates, Aristides Gionis, Flavio Junqueira, Vanessa Murdock, Vassilis Plachouras, and Fabrizio Silvestri. 2007. The Impact of Caching on Search Engines. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 183–190.

Ricardo Baeza-Yates, Vanessa Murdock, and Claudia Hauff. 2009. Efficiency trade-offs in two-tier Web search systems. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 163–170.

Luiz Andró Barroso, Jeffrey Dean, and Urs Hölzle. 2003. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro.* 23, 2 (2003), 22–28.

Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 426–434.

Eric W. Brown. 1995. Fast evaluation of structured queries for information retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 30–38.

Stefan Büttcher and Charles L. A. Clarke. 2006. A document-centric approach to static index pruning in text retrieval systems. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 182–189.

Fidel Cacheda, Victor Carneiro, Vassilis Plachouras, and Iadh Ounis. 2007. Performance analysis of distributed information retrieval architectures using an improved network simulation model. *Information Processing and Management.* 43 (2007), 204–224.

Jamie Callan. 2000. Distributed Information Retrieval. In *Advances in Information Retrieval*, W. Bruce Croft (Ed.). 127–150.

Jamie Callan, Margaret Connell, and Aiqun Du. 1999. Automatic Discovery of Language Models for Text Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 479–490.

James P. Callan, Zhihong Lu, and W. Bruce Croft. 1995. Searching distributed collections with inference networks. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 21–28.

B. Barla Cambazoglu, Vassilis Plachouras, and Ricardo Baeza-Yates. 2009. Quantifying performance and quality gains in distributed Web search engines. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 411–418.

David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoölle S. Maarek, and Aya Soffer. 2001. Static Index Pruning for Information Retrieval Systems. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 43–50.

Abdur Chowdhury and Greg Pass. 2003. Operational requirements for scalable search systems. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 435–442.

Charles Clarke, Nick Craswell, and Ian Soboroff. 2004. Overview of the TREC 2004 Terabyte track. In *Proceedings of the 2004 Text Retrieval Conference*.

Edleno S. de Moura, Célia F. dos Santos, Daniel R. Fernandes, Altigran S. Silva, Pavel Calado, and Mario A. Nascimento. 2005. Improving Web search efficiency via a locality based static pruning method. In *Proceedings of the 14th International Conference on World Wide Web*. 235–244.

Arthur P. Dempster, N M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B.* 39, 1 (1977), 1–38.

A. El-Hamdouchi and P. Willett. 1989. Comparison of hierarchic agglomerative clustering methods for document retrieval. *The Computer Journal*. 32, 3 (1989), 220–227.

Tiziano Fagni, Raffaele Perego, Fabrizio Silvestri, and Salvatore Orlando. 2006. Boosting the Performance of Web Search Engines: Caching and Prefetching Query Results by Exploiting Historical Usage Data. *ACM Transactions on Information Systems*. 24, 1 (Jan. 2006), 51–78.

Ana Freire, Craig Macdonald, Nicola Tonellotto, Iadh Ounis, and Fidel Cacheda. 2013. Hybrid Query Scheduling for a Replicated Search Engine.. In *Proceedings of the European Conference on Information Retrieval*. 435–446.

James C. French, Allison L. Powell, Jamie Callan, Charles L. Viles, Travis Emmitt, Kevin J. Prey, and Yun Mou. 1999. Comparing the performance of database selection algorithms. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 238–245.

Qingqing Gan and Torsten Suel. 2009. Improved Techniques for Result Caching in Web Search Engines. In *Proceedings of the 18th International Conference on World Wide Web*. 431–440.

Luis Gravano and Héctor García-Molina. 1995. Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. In *Proceedings of the Conference on Very Large Data Bases*. 78–89.

Luis Gravano, Héctor García-Molina, and Anthony Tomasic. 1994. The effectiveness of GIOSS for the text database discovery problem. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. 126–137.

Luis Gravano, Héctor García-Molina, and Anthony Tomasic. 1999. GlOSS: text-source discovery over the Internet. *ACM Transactions on Database Systems*. 24 (June 1999), 229–264. Issue 2.

Alan Griffiths, H.Claire Luckhurst, and Peter Willett. 1986. Using inter-document similarity information in document retrieval systems. *Journal of the American Society for Information Science*. 37, 1 (1986), 3–11.

J. Heaps. 1978. *Information Retrieval – Computational and Theoretical Aspects*. Academic Press Inc.

N. Jardine and Cornelis Joost van Rijsbergen. 1971. The use of hierarchical clustering in information retrieval. *Information Storage and Retrieval*. 7 (1971), 217–240.

Anagha Kulkarni. 2013. *Efficient and Effective Large-scale Search*. Ph.D. Dissertation. Carnegie Mellon University.

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

Anagha Kulkarni and Jamie Callan. July 2010b. Topic-based Index Partitions for Efficient and Effective Selective Search. In *SIGIR 2010 Workshop on Large-Scale Distributed Information Retrieval*. 19–24.

Anagha Kulkarni, Almer Tigelaar, Djoerd Hiemstra, and Jamie Callan. 2012. Shard Ranking and Cutoff Estimation for Topically Partitioned Collections. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 555–564.

Leah S. Larkey, Margaret E. Connell, and Jamie Callan. 2000. Collection selection and results merging with topically organized U.S. patents and TREC data. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 282–289.

Ronny Lempel and Shlomo Moran. 2003. Predictive Caching and Prefetching of Query Results in Search Engines. In *Proceedings of the 12th International Conference on World Wide Web*. 19–28.

S. Lloyd. 2006. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*. 28, 2 (Sept. 2006), 129–137.

Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2012. Learning to predict response times for online scheduling. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 621–630.

A. Moffat, W. Webber, J. Zobel, and R. Baeza-Yates. 2007. A Pipelined Architecture for Distributed Text Query Evaluation. *Information Retrieval*. 10, 3 (2007), 205–231.

Linh Thai Nguyen. 2009. Static Index Pruning for Information Retrieval Systems: A Posting-Based Approach. In *SIGIR 2009 Workshop on Large-Scale Distributed Information Retrieval*. 25–32.

Paul Ogilvie and Jamie Callan. 2001. Experiments Using the Lemur Toolkit. In *Proceedings of the 2001 Text Retrieval Conference*.

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

Diego Puppin, Fabrizio Silvestri, Raffaele Perego, and Ricardo Baeza-Yates. 2010. Tuning the capacity of search engines: Load-driven routing and incremental caching to reduce and balance the load. *ACM Transactions on Information Systems*. 28, 2 (2010), 5:1–5:36.

Knut Magne Risvik, Yngve Aasheim, and Mathias Lidal. 2003. Multi-Tier Architecture for Web Search Engines. In *Proceedings of the First Latin American Web Congress*. 132–143.

Paricia Correia Saraiva, Edleno Silva de Moura, Novio Ziviani, Wagner Meira, Rodrigo Fonseca, and Berthier Riberio-Neto. 2001. Rank-preserving Two-level Caching for Scalable Search Engines. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 51–58.

Milad Shokouhi. 2007. Central-Rank-Based Collection Selection in Uncooperative Distributed Information Retrieval. In *Proceedings of the 29th European Conference on Information Retrieval*. 160–172.

Milad Shokouhi and Luo Si. 2011. Federated Search. *Foundations and Trends in Information Retrieval.* 5, 1 (2011), 1–102.

Luo Si and Jamie Callan. 2003. Relevant document distribution estimation method for resource selection. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 298–305.

Alan F. Smeaton and Cornelis Joost van Rijsbergen. 1981. The nearest neighbour problem in information retrieval: an algorithm using upperbounds. In *Proceedings of the 4th Annual International ACM SIGIR Conference on Information Storage and Retrieval*. 83–87.

Trevor Strohman, Howard Turtle, and W. Bruce Croft. 2005. Optimization strategies for complex queries. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 219–225.

Paul Thomas and Milad Shokouhi. 2009. SUSHI: Scoring scaled samples for server selection. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 419–426.

Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. 2013. Efficient and effective retrieval using selective pruning. In *Proceeding of the International Conference on Web Search and Data Mining*. 63–72.

Howard Turtle and James Flood. 1995. Query evaluation: strategies and optimizations. *Information Processing and Management.* 31, 6 (1995), 831–850.

Cornelis Joost van Rijsbergen. 1979. *Information Retrieval*. Butterworths.

Ellen M. Voorhees. 1985. The cluster hypothesis revisited. In *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 188–196.

Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. 1995. Learning collection fusion strategies. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 172–179.

Peter Willett. 1988. Recent trends in hierarchic document clustering: a critical review. *Information Processing and Management.* 24, 5 (1988), 577–597.

Wai Yee Peter Wong and Dik Lun Lee. 1993. Implementations of Partial Document Ranking Using Inverted Files. *Information Processing and Management.* 29, 5 (1993), 647–669.

Jinxi Xu and W. Bruce Croft. 1999. Cluster-based language models for distributed retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 254–261.