Query-Biased Partitioning for Selective Search

Zhuyun Dai, Chenyan Xiong, Jamie Callan
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{zhuyund, cx, callan}@cs.cmu.edu

ABSTRACT

Selective search is a cluster-based distributed retrieval architecture that reduces computational costs by partitioning a corpus into *topical shards*, and *selectively* searching them. Prior research formed topical shards by clustering the corpus based on the documents' contents. This content-based partitioning strategy reveals common topics in a corpus. However, the topic distribution produced by clustering may not match the distribution of topics in search traffic, which may reduce the effectiveness of selective search.

This paper presents a query-biased partitioning strategy that aligns document partitions with topics from *query logs*. It focuses on two parts of the partitioning process: clustering initialization and document similarity calculation. A *query-driven* clustering initialization algorithm uses topics from query logs to form cluster seeds. A *query-biased* similarity metric favors terms that are important in query logs. Both methods boost retrieval effectiveness, reduce variance, and produce a more balanced distribution of shard sizes.

Keywords

Distributed Retrieval, Selective Search, Shard Partitioning

1. INTRODUCTION

Selective search [9, 11] is a cluster-based distributed retrieval architecture that aims to reduce computational costs. During indexing, it uses content-based similarity to partition the document corpus into *topical shards*. At query time, a *resource selection* algorithm [2, 12, 18, 19] selects (routes the query to) the (typically few) shards likely to contain most of the relevant docuents. The results from each selected shard are merged to produce the search results. Prior research shows that selective search reduces computational costs by more than 80% without reducing search accuracy [9].

Selective search is based on the hypothesis that similar documents are likely to be relevant to the same query; thus for any query, only a few shards must be searched. Resource selection algorithms use term frequency information to decide which shards probably contain documents relevant to a given query [2, 12, 18, 19]. Prior research defined documents to be similar if their *contents* are similar, and used LDA or k-means clustering to partition the corpus [9, 10, 11]. However, Dai et al. [6] showed that this strategy sometimes places relevant documents in shards that are not topically related to the queries that make them relevant (*unrepresenatitive shards*). When relevant documents are placed in unrepresentative shards, the resource selection algorithm is more likely to select the wrong shards, which reduces search accuracy. For example, most of the relevant documents for the TREC query `obama family tree` are in a shard about people names and genealogy but the highest ranked shards are about U.S. politics. It suggests that topics of the content-based partitions are not fully matched with user information needs.

This paper investigates how to group together documents that satisfy the same user intent. Query logs reflect topics frequently queried by users, and are used to align the partitioning with user intents. This paper proposes a a query-driven clustering initialization algorithm and a query-biased similarity metric. The query-driven initialization algorithm enables the clustering to start with topics distilled from the query log, by using topics from the query log to initialize the document clustering. The query-biased similarity metric makes sure that clustering stays focused on topics from the query log, by biasing towards important query log terms.

Experiments investigate the effectiveness of the proposed approaches on selective search tasks with two large datasets: Gov2 and the Category B portion of ClueWeb09 (CW09-B). We observe significant improvements on selective search's retrieval effectiveness and stability over the previous state-of-the-art with similar computational costs. The proposed methods are especially effective at Recall-oriented tasks, which are harder to improve because selective search only searches a small part of the corpus. These methods also produce more balanced shard sizes, which is important for load balancing and managing query latency. Experiments show the methods' robustness to the temporal mismatch between training and testing queries, and the robustness to a key parameter.

The rest of the paper is structured as follows. First, Section 2 positions the work with respect to prior work. The details about how to operationalize query-biased partitioning are given in Sections 3 and 4. Section 5 describes the experimental methodology. The experimental results are discussed in Section 6. Finally, Section 7 concludes.

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA

DOI: http://dx.doi.org/10.1145/2983323.2983706

2. RELATED WORK

Cluster-based retrieval improves efficiency by clustering the collection and selecting one or more clusters during query processing. Traditional cluster-based retrieval partitions a document corpus into many small clusters during indexing, for example, $O(\sqrt{\texttt{number of documents}})$ clusters. The corpus is stored in a single inverted index, with postings sorted by cluster id. During retrieval, only the portions of inverted lists for selected clusters are decompressed and processed. Most of each posting list is skipped, which substantially reduced search costs [1]

Another direction to improve efficiency is distributed information retrieval. It reduces query latency by dividing a large document corpus into small partitions (*index shards*) that are placed on different machines and searched in parallel. The most common strategy partitions the corpus randomly, which balances workloads.

Selective search combines ideas from traditional cluster-based and distributed information retrieval. It clusters the collection and store each cluster in a separate index. Given a query, a resource selection algorithm such as ReDDE [18], Rank-S [12] or Taily [2] selects which shards to search. Total i/o costs are lower than with traditional cluster-based retrieval because selective search uses fewer and larger shards so it is practical to store clusters in separate indexes. For example, Kulkarni and Callan split 50 million documents into 100 clusters of 500 thousand documents each [11]. Total computational costs are lower than with traditional distributed retrieval because just a few index shards are searched for any query [10, 2, 9, 11]. Simple strategies enable load to be balanced evenly across a cluster of machines [8].

The key to effective selective search is how the document corpus is partitioned into shards. The state-of-the-art is a sample-based K-means clustering method [10]. First it performs k-means clustering on a small sample of the collection. Then it projects each remaining document into the nearest cluster. The distance function is a symmetric version of negative Kullback-Liebler Divergence (KLD) that incorporates the inverse collection frequency of the term into the metric.

Prior research showed that sample-based k-means produces partitions that are effective for most queries, but for some queries relevant documents are in the 'wrong' shards [6]. For example, documents relevant to the query `obama family tree` get placed in a names and genealogy shard instead of the politics shard. In these cases, relevant documents are placed with documents that are similar in content rather than documents that answer the same query.

This paper investigates how to improve the sample-based K-means partitioning by providing guidance from the query log. Guidance about desired cluster properties can improve clustering results. Constrained clustering instructs the algorithm to keep specified instances in the same cluster (*must link*) or in different clusters (*must not link*) [20]. Metric learning learns a similarity function that minimizes the distances between data points that are known to be similar [21]. It is an open question whether these methods would improve partitioning for selective search.

Query logs have been used to guide partitioning for cluster-based architectures similar to selective search. Puppin et al. [17] treat queries from the log as indexing terms, and represent documents by the queries that retrieved them from a global index. If a document is not retrieved by any query from the log, it is assigned to a default cluster. Poblete and Baeza-Yates [16] also represent documents by the queries that retrieved them. Clusters are produced by co-clustering on a query-document matrix, and unretrieved documents are assigned to a default cluster. In evaluations on different datasets, these methods improved retrieval accuracy but assigned 52% of the documents to the default cluster [16, 17]. Poblete and Baeza-Yates argue that documents in the default cluster are low priority and can be assigned to a lower tier of the index. This observation may be true if the query log is comprehensive and query traffic does not change suddenly, however it was not evaluated experimentally.

Clustering initialization is another line of related research. Algorithms such as k-means are somewhat sensitive to initialization conditions. Many clustering initialization methods have been proposed [4]. Often they start with an analysis of the content being clustered. Most prior research on cluster-based and selective search ignored the effects of initial conditions on search accuracy.

3. QUERY-DRIVEN CLUSTERING INITIALIZATION

The first step to improved partitioning is `QInit`, a query-driven approach that initializes document clustering with topics discovered from a query log.

`QInit` mines user topics from a query log by clustering terms from the query log, with each word represented by its word embedding. Word embeddings are learned by neural network models that encode a word's context information in a continuous vector. They are very effective for measuring semantic similarity between words [3, 14]. By clustering the word embeddings of query log terms, semantically similar terms are grouped together to represent user search topics.

We use cosine similarity to measure the distance between terms' embedding, and average-link agglomerative bottom-up clustering to cluster terms. In the beginning, each word vector starts in its own cluster. Pairs of clusters that minimize the average distance are merged successively. The process stops when the number of clusters is reduced to a pre-specified number K. Unlike k-means, agglomerative clustering is deterministic, thus can reduce variance in the partitioning process. Its time complexity is cubic to the number of data points, but the number of frequent terms in the query log is small and thus can be clustered efficiently.

The discovered user topics (term clusters) are used to initialize the document partitioning algorithm. For each term cluster, we form a virtual document d_0 that contains all terms assigned to the term cluster. The weight of a term in the virtual document d_0 is defined using its importance in the query log, as shown below.

$$w_{t,d_0} = \log(tf_{t,Q} + 1) \times \log\left(\frac{|D|}{df_{t,D}} + 1\right) \qquad (1)$$

$tf_{t,Q}$ is the term frequency of term t in the query log, $|D|$ is the number of documents in the corpus, and $df_{t,D}$ is the number of documents containing term t.

The first component, $\log(tf_{t,Q} + 1)$, is the query-log TF part. It reflects the term's popularity in the query log, and promotes the importance of terms frequently used by users in search. The log function is used because term distribution in the query log is very skewed. A few terms with very high frequency in the query logs will dominate the clustering if $tf_{t,Q}$ is used directly.

Table 1: Examples of shards generated with `QInit`. The second column shows the terms in the initial cluster centroid with the highest weight defined in Equation 1. The third column shows queries that have relevant document in the resulting shard.

Dataset	ID	Top Terms in Initial Seed	Relevant Queries
CW09-B	1	wine, tea, coffee, smoking, alcohol, drink	starbucks, quit smoking
	2	animal, cock, bird, wild, egg, cat	dinosaurs, arizona game and fish, moths,...
Gov2	1	tax, revenue, loans, business, bank, taxation	reverse mortgages, timeshare resales, ...
	2	diabetes, autism, obesity, arthritis, hypertension, celiac	aspirin cancer prevention, embryonic stem cells, ...

Table 2: Notation.

t, d, c	term, document, cluster
D	the document corpus
$\vec{d}$	feature vector of document
$\vec{c}$	feature vector of cluster centroid
$tf_{t,d}$	term frequency of t in document d
$tf_{t,Q}$	term frequency of t in the query log
$df_{t,D}$	document frequency of term t in the corpus
$w_q(t)$	term t's bias weight learned from query log

The second component, $\log\left(\frac{|D|}{df_{t,D}}+1\right)$, is the inverted document frequency in the collection (collection IDF). It is used to demote terms that are too common in the corpus. For example, "com" is a very common term in the web environment. Although users often use "com" to form queries, few of them are actually searching for information about "com". The inverted document frequency component helps to select terms that represent user topics.

These K virtual documents are then used as seeds to initialize the partition process, for example, the k-mean clustering of documents [10].

Table 1 shows a few shards generated from K-means clustering initialized by `QInit`. For each selected shard, we show its seeds from the `QInit` algorithm, and queries that have relevant documents in that shard. For example, the shard initialized with 'wine, tea, coffee, ...' contains relevant documents of the query 'starbucks'. The seeds demonstrate that `QInit` can find semantically coherent topics from the query log. The similarity between a shard's seed and its relevant queries suggests that the clustering process is guided by the seeds and aligns document topics with query log topics.

4. QUERY-BIASED SIMILARITY METRIC

This section presents `QKLD`, a query-biased similarity metric for document partitioning. `QKLD` addresses the mismatch between corpus topics and user topics by biasing clustering towards important query log terms. It is built upon the previous state-of-art similarity metric in selective search [11], `KLD`, a symmetric negative Kullback-Leibler divergence based similarity function.

In `KLD`, each document d is represented by a bag-of-word vector $\vec{d}$. The t^{th} dimension of the vector corresponds to t^{th} term in the vocabulary. Each cluster c is represented by its centroid $\vec{c}$, the average of its documents' vectors. The similarity between a document $\vec{d}$ and a cluster $\vec{c}$ is:

$$sim_{KLD}(d,c) = \sum_{t \in d \bigcap c} s_{KLD}\left(\vec{d_t}, \vec{c_t}\right) \quad (2)$$

where each term t that appears both in the document and the cluster makes a similarity contribution $s_{KLD}(\vec{d_t}, \vec{c_t})$ to the total similarity.

This similarity function measures the similarity based on the content of documents. Clustering with a content-based similarity function reveals topics that are common in the document corpus, but may not match the distribution of topics in user's queries [6]. We address this mismatch by introducing a bias weight into the similarity metric:

$$sim_{QKLD}(\vec{d},\vec{c}) = \sum_{t \in d \bigcap c} (w_q(t)+b) \times s_{KLD}\left(\vec{d_t}, \vec{c_t}\right). \quad (3)$$

Equation 3 assigns a weight $w_q(t)+b$ to each term to reflect its importance in the query log. $w_q(t)$ is same as the weight used in the `QInit` initialization algorithm in Equation 1:

$$w_q(t) = \log(tf_{t,Q}+1) \times \log\left(\frac{|D|}{df_{t,D}}+1\right). \quad (4)$$

b is a positive smoothing parameter that controls the importance of the bias weights. It enables terms that do not appear in the query log to have non-zero weights. Thus, document content plays a role in clustering, and documents that do not contain query terms are handled naturally.

$w_q(t)$ biases `QKLD` clustering towards important query log terms. Thus, `QKLD` favors terms that are not only important in the document content, but also in the query log, with the two parts balanced by b.

For the s_{KLD} part, it follows `KLD`, and is defined as follows:

$$s_{KLD}(\vec{d_t}, \vec{c_t}) = p_c(t)\log\frac{p_d(t)}{\lambda p_B(t)} + p_d(t)\log\frac{p_c(t)}{\lambda p_B(t)} \quad (5)$$

where λ is a smoothing parameter. $p_c(t)$ is the language model of the cluster centroid c, which is the mean of its documents' vectors:

$$p_c(t) = \frac{1}{|c|}\sum_{\vec{d} \in c} \vec{d_t} \quad (6)$$

where $|c|$ is the number of documents in cluster c. $p_d(t)$ is the document's language model with Jelinek-Mercer smoothing:

$$p_d(t) = (1-\mu)\vec{d_t} + \mu p_B(t) \quad (7)$$

where μ is the smoothing parameter. $p_B(t)$ is the background model of the collection:

$$p_B(t) = \frac{1}{|D|}\sum_{\vec{d} \in D} \vec{d_t}. \quad (8)$$

It reflects global term statistics and behaves similarly to the traditional inverse document frequency (IDF) statistic [22].

Finally, the document's bag-of-word vector uses normalized term frequency:

$$\vec{d_t} = \frac{tf_{t,d}}{\sum_{t'} tf_{t',d}}. \quad (9)$$

Table 3: Datasets and query sets. WT = TREC Web Track, TB = TREC Terabyte Track.

Datasets	CW09-B	Gov2
Documents	50,220K	25,205K
Vocabulary	96M	39M
Total Queries	200	150
Query Sets	WT 09-12	TB 04-06
Average Query Length	2.2	3.1

Table 4: Queries in the first 2 months of the AOL query logs. AOL-All: all queries from the first 2 months; used for CW09-B experiments. AOL-Gov2: the subset of AOL-All queries with clicks on .gov and .us URLS; used for Gov2 experiments.

Dataset	AOL-All	AOL-Gov2
Queries	24,189,556	540,285
Queries after filtering	13,950,463	403,610
Terms (w/o numbers)	978,714	69,482
Terms after filtering	80,963	14,018

5. EXPERIMENTAL SETUP

Datasets: Experiments were conducted with two datasets that have different sizes and characteristics: ClueWeb09-B and Gov2. ClueWeb09-B (CW09-B) is the 50 million page 'category B' portion of the ClueWeb09 dataset[1]. Gov2 [5] is 25 million web pages from the US government web domains. The queries for CW09-B were from the TREC 2009-2012 Web Tracks topics: 4 sets of 50 queries. The queries for Gov2 were from TREC 2004-2006 Terrabyte Track topics: 3 sets of 50 queries. Summary statistics are given in Table 3.

Query Log: The proposed methods require a query log. Our experiments used the AOL query log[2]. For CW09-B the entire AOL query log (AOL-All) was used. For Gov2 we used the AOL-Gov2 subset: AOL queries with clicks on '.gov' and '.us' URLs. We also randomly sampled queries from the last 1 month of each query log to create supplemental testing sets (3.5K queries for AOL-ALL, 1.5K queries for AOL-Gov2). The supplementary testing sets were used in an experiment that measures the overlap in documents retrieved by exhaustive and selective search.

The query log was preprocessed to reduce noise. First, we removed successive duplicate queries from the same user because they belong to a single search activity, and URL queries because selective search is the wrong solution for *navigational* queries. Second, 418 Indri stopwords were removed. Finally, we removed terms with low term frequency in the query log or low document frequency in the corpus. The term frequency distribution in the query log has a very long tail: Over 91% of the terms appear less than 15 times in AOL-All, and 80% of the terms in AOL-Gov2 appear less than 5 times. A few hundred terms were removed due to low corpus document frequency ($df < 200$). Tail terms were filtered out because they are mainly misspellings and rare terms that would introduce noise into the partitioning. The preprocessing greatly reduced the vocabulary size, making it efficient to perform `QKLD` and `QInit`. Table 4 shows query log statistics before and after preprocessing.

Word Embeddings: `QInit` performs term-level clustering with word embeddings. We used word vectors of 100 dimensions trained on a spam-filtered subset of CW09-B with Mikolov's Continuous Bag-of-Words Model (CBOW) [14, 15].

Baseline and Proposed Methods: As the baseline we employed the partitioning technique proposed by Kulkarni and Callan [11], the previous state-of-the-art in shard partitioning. The baseline uses random seeding to initialize the sample-based k-means clustering, and `KLD` similarity function (Equation 2) to assign documents. We refer to the baseline as `KLD-Rand`. We compared three partitioning strategies to the baseline: `QKLD-Rand`, `KLD-QInit`, and `QKLD-QInit`. `QKLD-Rand` used the `QKLD` similarity metric (Equation 3) for document allocation, and random seeds for clustering initialization. `KLD-QInit` used the baseline similarity metric, but clustering was initialized with the new `QInit` algorithm described in Section 3. `QKLD-QInit` used the query-biased `QKLD` similarity function, and clustering was initialized with the query-driven `QInit` algorithm.

As suggested by Kulkarni [9], we set `KLD` parameters $\lambda = 0.1$ and $\mu = 0.1$ (Equations 5 and 7). The same values were used in `QKLD` for consistency. The `QKLD` smoothing parameter b (Equation 4) was chosen by a parameter sweep, and was set as $b = 1/16$ for CW09-B and $b = 1/8$ for Gov2. The sensitivity of `QKLD` to b is investigated in Section 6.3.2.

Following Kulkarni, all methods first cluster a 1% document sample, then project the remaining documents into the nearest cluster. A second level of clustering was performed to split big shards. The number of clusters K was set to produce shards of 500K documents on average for CW09-B, as specified by Kulkarni. She used similar cluster sizes for Gov2, however we found that smaller clusters were more effective. Our Gov2 experiments used clusters of about 170K documents.

Search Engine Indexes: Each document partition was indexed and searched by Indri[3], a widely-used open-source search engine. All parameter settings were Indri defaults, which have been shown to be effective for these corpora.

Resource Selection: Rank-S, a state-of-the-art sample-based resource selection algorithm, was used to select shards for each query. We used a 1% central sample index (CSI), as suggested by Kulkarni [9]. For consistency with previous work [2, 11, 12], we chose a parameter B that on average selected about 3% of CW09-B shards and 5% of Gov2 shards.

Queries: Unstructured bag-of-words queries were transformed into more effective structured queries by sequential dependence models (SDM) with parameters $(0.8, 0.1, 0.1)$ [13].

Evaluation Methods: Each approach was evaluated by its effect on search accuracy up to rank 1,000. Accuracy was measured by standard metrics: Precision at rank 10 (P@10), Normalized Discounted Cumulative Gain at rank 100 (NDCG@100), and Mean Average Precision at rank 1,000 (MAP). One experiment also measured search quality by the overlap in results returned by exhaustive search and selective search.

There are several random components in building a selective search system, such as random sampling and random seeding in document partitioning. Rank-S also requires ran-

[1]http://lemurproject.org/clueweb09/

[2]Some reviewers question whether it is appropriate to use the AOL query log; however it has been used by much prior research and is widely available, which makes our results reproducible. Experiments in Section 6.3.1 investigate the sensitivity of the proposed methods to the query log.

[3]http://lemurproject.org/indri/

dom sampling. In order to rule out random effects and evaluate system variance, we generated 10 independent system instances for each partitioning strategy. Statistical significance of model differences was judged by a query-level two-sided paired permutation test with $p < 0.05$.

6. EXPERIMENTAL RESULTS

This section describes experiments that compare the proposed QKLD-Rand and QKLD-QInit partitioning strategies to the baseline KLD-Rand. First we analyze the distribution of relevant documents across shards for each method to gain intuition about each method's effect on partitioning. Then we investigate each method's retrieval accuracy, robustness, and efficiency. Collectively, they provide a detailed analysis of how each method performs.

6.1 Clustering Analysis

The goal of query-biased partitioning is to better concentrate documents that are related to the same query. Concentrating the relevant documents for a query into few shards makes resource selection easier, and enables good Recall when only a few shards are searched. We begin by examining how well it achieves that goal.

We analyze the results of four partitioning strategies: The baseline KLD similarity metric with random k-means initialization (KLD-Rand); KLD similarity metric with Q-Init k-means intialization (KLD-QInit); and QKLD similarity metric with random k-means initialization (QKLD-Rand); QKLD similarity metric with QInit k-means initialization (QKLD-QInit).

Given query q, a relevance-based ranking s^q is formed by sorting shards by the number of relevant documents they contain. The percentage of relevant documents contained in the first t% of the shards is defined as:

$$coverage_t(q) = \frac{\sum_{i=1}^{floor(N*t\%)} R^q_{s^q_i}}{R^q} \quad (10)$$

where R^q is the total number of relevant documents for query q; $R^q_{s^q_i}$ is the number of relevant documents for query q in the shard at position i in s^q. The average coverage over the query set is defined as shown below.

$$coverage_t = \frac{\sum_{q=1}^{|Q|} coverage_t(q)}{|Q|}. \quad (11)$$

Table 5 reports the coverage for several common levels of search effort, defined by the percentage of shards searched. For both datasets and all partitioning methods, the coverage is close to 100% at $t = 10$. That is, relevant documents for most queries were concentrated in no more than 10% of the shards. Indeed, on average more than half of a query's relevant documents are concentrated in no more than 1% of shards. Such a skewed distribution allows selective search to search only a few shards, which reduces search costs.

QKLD was a more effective similarity metric than KLD across both datasets and both initialization methods. Weighting terms by frequency in documents *and* a query log is superior to weighting by frequency in documents alone. QKLD also does not produce the large 'default' cluster observed with other partitioning methods that use query logs [16, 17]; out of vocabulary problems cannot occur when the similarity metric considers document content.

Table 5: Relevant documents coverage of KLD-Rand, QKLD-Rand, KLD-QInit, QKLD-QInit. * indicates statistically significant difference with KLD-Rand. Bold numbers indicate highest coverage among the compared methods. CW09-B coverage is averaged over 200 queries from WT09-12, Gov2 coverage is averaged over 150 queries from TB04-06.

Dataset	Method	Percentage of Shards (t)			
		1%	3%	5%	10%
CW09-B	KLD-Rand	0.60	0.86	0.96	0.99
	KLD-QInit	0.60	0.86	0.95	0.99
	QKLD-Rand	0.65*	0.89	0.97	0.99
	QKLD-QInit	**0.67***	**0.90***	0.97*	**1.00**
Gov2	KLD-Rand	0.50	0.86	0.95	0.99
	KLD-QInit	0.50	0.86	0.95	0.99
	QKLD-Rand	0.54*	0.89*	0.95	0.99
	QKLD-QInit	**0.56***	**0.90***	**0.96**	0.99

Table 6: Published results for exhaustive search (Exh) and selective search (Sel). CW09-B Queries: 100 queries from 2009-2010. Gov2 Queries: 150 queries from 2004-2006.

Dataset	Source	P@10		MAP	
		Exh.	Sel.	Exh.	Sel.
CW09-B	[12]	0.27	0.30	0.18	0.16
	[2]	0.29	0.31	0.20	0.15
	Ours	0.29	0.31	0.19	0.16
Gov2	[12]	0.58	0.58	0.32	0.29
	[2]	0.58	0.55	0.34	0.24
	Ours	0.58	0.57	0.32	0.27

QInit cluster initialization was more effective than Rand for the QKLD similarity metric, but not for KLD. QKLD's query-biased similarity function is consistent with QInit seeds. The words in QInit seeds were likely to be given high weights by the QKLD similarity function. Working together, they kept the clustering focused on topics from the query log. On the other hand, KLD's similarity function only considers document content, which may differ substantially from query log content. The effects of the QInit seeds diminished quickly during k-means iterations, causing KLD-QInit clustering to converge to KLD-Rand behavior.

6.2 Search Effectiveness

Next we investigate the effects of QKLD-Rand and QKLD-QInit on retrieval effectiveness. As discussed in Section 6.1, KLD-QInit was no better than the baseline, so we omit it from the following experiments.

We first show that our KLD-Rand baseline is trustworthy by comparing our exhaustive search and baseline selective search effectiveness with previously published results. Table 6 reports the the search effectiveness scores from Kulkarni and Callan [11] and Aly et al. [2]. They partitioned collections with KLD-Rand, which is our baseline. Our baseline results are also reported. To be consistent with prior work [11, 2], CW09-B was evaluated with 100 queries from the TREC 2009 and 2010 Web Track. (Later experiments in this paper use 200 queries for CW09-B to improve reliability.) As shown in Table 6, our baseline is similar to published work and can be considered reliable.

Tables 7 and 8 report on the accuracy of selective search using partitions of Gov2 and CW09-B produced by KLD-Rand, QKLD-Rand, and QKLD-QInit, averaged across 10 sys-

Table 7: Retrieval accuracy of exhaustive search and three selective search methods on the CW09-B. Baseline: KLD-Rand. Mean and standard deviation are calculated over 10 independent system instances. 'All' is the performance over all 200 queries from 2009-2012. Percentage difference w.r.t. KLD-Rand are in round brackets. **k** indicates statistically significant difference with KLD-Rand. **q** indicates statistically significant difference with QKLD-Rand. Bold numbers indicate smallest standard deviation among the compared methods.

a) P@10

Query Set	Mean				Standard Deviation ($\times 10^{-3}$)		
	Exhaustive	KLD-Rand	QKLD-Rand	QKLD-QInit	KLD-Rand	QKLD-Rand	QKLD-QInit
2009	0.358	0.378	0.395(+4%)	0.397^{k}(+5%)	18.47	24.07	**14.64**
2010	0.212	0.243	0.261^{k}(+7%)	0.268^{k}(+10%)	16.96	13.76	**9.93**
2011	0.234	0.259	0.258(−0%)	0.270^{k}(+4%)	13.15	20.94	**12.77**
2012	0.208	0.220	0.223(+1%)	$0.232^{k,q}$(+5%)	20.28	18.05	**9.90**
All	0.253	0.275	0.284^{k}(+3%)	$0.290^{k,q}$(+5%)	7.50	9.74	**6.58**

b) NDCG@100

Query Set	Mean				Standard Deviation ($\times 10^{-3}$)		
	Exhaustive	KLD-Rand	QKLD-Rand	QKLD-QInit	KLD-Rand	QKLD-Rand	QKLD-QInit
2009	0.331	0.280	0.302^{k}(+8%)	0.304^{k}(+9%)	19.34	17.56	**10.80**
2010	0.252	0.246	0.266^{k}(+8%)	0.276^{k}(+12%)	13.09	7.05	**4.42**
2011	0.325	0.269	0.286^{k}(+6%)	$0.297^{k,q}$(+10%)	17.15	14.94	**9.81**
2012	0.235	0.219	0.240^{k}(+9%)	0.243^{k}(+10%)	13.58	12.18	**6.51**
All	0.286	0.254	0.273^{k}(+7%)	0.279^{k}(+10%)	9.92	5.39	**5.07**

c) MAP@1000

Query Set	Mean				Standard Deviation ($\times 10^{-3}$)		
	Exhaustive	KLD-Rand	QKLD-Rand	QKLD-QInit	KLD-Rand	QKLD-Rand	QKLD-QInit
2009	0.197	0.162	0.183^{k}(+13%)	0.187^{k}(+15%)	14.78	12.21	**10.88**
2010	0.176	0.151	0.172^{k}(+14%)	$0.184^{k,q}$(+21%)	12.02	**5.37**	6.85
2011	0.196	0.156	0.165(+6%)	$0.171^{k,q}$(+9%)	14.05	11.99	**8.02**
2012	0.174	0.149	0.167^{k}(+12%)	0.169^{k}(+13%)	9.54	7.39	**4.16**
All	0.186	0.155	0.172^{k}(+11%)	$0.178^{k,q}$(+15%)	8.77	**3.75**	5.22

Table 8: Retrieval performance of exhaustive search and three selective search methods on the Gov2. Baseline: KLD-Rand. Mean and standard deviation are calculated over 10 independent system instances. 'All' is the performance over all 150 queries from 2004-2006. Percentage difference w.r.t. KLD-Rand are in round brackets. **k** indicates statistically significant difference with KLD-Rand. **q** indicates statistically significant difference with QKLD-Rand. Bold numbers indicate smallest standard deviation among the compared methods.

a) P@10

Query Set	Mean				Standard Deviation ($\times 10^{-3}$)		
	Exhaustive	KLD-Rand	QKLD-Rand	QKLD-QInit	KLD-Rand	QKLD-Rand	QKLD-QInit
2004	0.561	0.570	0.570(+0%)	$0.589^{k,q}$(+3%)	12.91	**3.83**	5.73
2005	0.616	0.592	0.603^{k}(+2%)	0.612^{k}(+3%)	11.59	11.77	**8.78**
2006	0.566	0.554	0.572^{k}(+3%)	0.577^{k}(+4%)	10.38	10.84	**7.44**
All	0.581	0.572	0.582^{k}(+2%)	$0.593^{k,q}$(+4%)	7.10	6.61	**4.59**

b) NDCG@100

Query Set	Mean				Standard Deviation ($\times 10^{-3}$)		
	Exhaustive	KLD-Rand	QKLD-Rand	QKLD-QInit	KLD-Rand	QKLD-Rand	QKLD-QInit
2004	0.428	0.414	0.416(+0%)	$0.428^{k,q}$(+3%)	7.77	**1.73**	3.59
2005	0.472	0.434	0.442^{k}(+2%)	$0.454^{k,q}$(+5%)	8.39	10.71	**5.24**
2006	0.470	0.430	0.444^{k}(+3%)	0.447^{k}(+4%)	12.03	7.51	**5.71**
All	0.457	0.426	0.434^{k}(+2%)	$0.443^{k,q}$(+4%)	5.72	3.82	**2.65**

c) MAP@1000

Query Set	Mean				Standard Deviation ($\times 10^{-3}$)		
	Exhaustive	KLD-Rand	QKLD-Rand	QKLD-QInit	KLD-Rand	QKLD-Rand	QKLD-QInit
2004	0.284	0.256	0.256(+0%)	$0.263^{k,q}$(+3%)	6.79	3.99	**2.73**
2005	0.331	0.284	0.295^{k}(+4%)	$0.303^{k,q}$(+7%)	**3.80**	5.31	5.07
2006	0.330	0.274	0.286^{k}(+4%)	0.291^{k}(+6%)	9.24	6.78	**3.38**
All	0.315	0.272	0.279^{k}(+3%)	$0.286^{k,q}$(+5%)	3.90	**2.52**	2.56

Table 9: Relative gains of QKLD-Rand and QKLD-QInit over KLD-Rand at five points in the document ranking.

	CW09-B		Gov2	
NDCG @Rank	QKLD -Rand	QKLD -QInit	QKLD -Rand	QKLD -QInit
10	3.77%	5.70%	1.91%	4.22%
30	5.49%	6.69%	2.19%	4.97%
100	7.70%	10.03%	2.68%	4.42%
500	9.44%	12.37%	3.46%	5.58%
1000	9.87%	13.26%	4.29%	6.52%

tem instances for each dataset. 'All' represents evaluation over all of the queries as a single set.

Partitions created by the QKLD similarity metric enabled more accurate search than partitions created by KLD for both datasets in almost every experimental condition; the differences were statistically significant for most of the conditions. Partitions created by QKLD also produced more consistent NDCG@100 and MAP@1000 (indicated by lower standard deviation) across different system instances, but the effect was less consistent for P@10. The QKLD query-biased similarity metric produces partitions that make it easier for resource selection algorithms to select the right shards.

Initializing QKLD partitioning with QInit further improved the effectiveness of selective search. The MAP@1000 gains over the KLD-Rand baseline ranged from 10.8% to 22.9% for CW09-B, and 5.2% to 7.0% for Gov2; the differences were all statistically significant. Similar improvements were observed for P@10 and NDCG@100. QInit initialization also produced consistent improvements on all metrics (indicated by lower standard deviation) across different system instances.

Win/Loss Behavior: Each method's *risk* was evaluated by analyzing the number of queries that it improved (*wins*) or hurt (*losses*) compared to KLD-Rand, using the mean of MAP@1000 over 10 system instances. On CW09-B, 63% of the 200 queries were improved by QKLD-Rand, and 68% were improved by QKLD-QInit. On Gov2, the percentages were 60% for QKLD-Rand and 64% for QKLD-QInit. For both methods, more queries improve than deteriorate. QKLD-QInit improved more and hurt fewer queries than QKLD-Rand.

Effects on Recall and Precision. Typically selective search searches only a small part of a corpus, for example 3-5% of the shards. Thus, usually it is harder for a new method to improve Recall than Precision. Here we discuss the effects of QKLD-Rand and QKLD-QInit on different recall levels. Table 9 shows the relative gains of QKLD-Rand and QKLD-QInit over KLD-Rand on NDCG@rank at four recall levels. NDCG scores at all depth were improved by both methods; deeper depths achieved more gains.

In our experience, selective search rarely harms Precision. As shown in Table 7, the P@10 of the selective search baseline (KLD-Rand) was *better* than exhaustive search on CW09-B. Selective search can improve Precision by filtering out some of the false-positive documents – documents that are highly ranked by exhaustive search but not topically relevant to the query. The partitioning process separates some of the false-positives from the true-positives, which improves Precision. QKLD and QInit had higher Precision than the baseline, which shows their ability to separate false-positives that are topically irrelevant but similar in content.

Usually Recall is harder to improve because searching fewer shards misses documents that were assigned to other shards. Often, maintaining Recall requires searching more shards [11]. QKLD-Rand and QKLD-QInit increased NDCG1000 by a factor of 10 on CW09-B, and a factor of more than 5 on Gov2. Recall was improved because these partitioning strategies cover more relevant documents with the first few shards, as shown in Table 5.

6.3 Robustness

This section investigated the sensitivity of QKLD-Init and QKLD-QInit to experimental conditions. It investigated the effects of the temporal mismatch between the training and testing queries, and the effects of a key parameter in QKLD.

6.3.1 Query Log Influences

The experiments above trained systems using the first two months of the AOL query logs, and tested using TREC queries. However, there is temporal mismatch between the AOL queries used for training and the TREC queries used for testing. The AOL query log was released in 2006. The TREC Gov2 queries were created 2004-2006, and the TREC CW09-B queries were created 2009-2012. The next experiment investigated how temporal mismatch between training and test conditions affects the performance of each method.

Two temporal conditions were compared. In the *unaligned* condition, as in the experiments above, the AOL query log was used for training, and TREC queries were used for testing. In the *aligned* condition, different portions of the AOL query log were used for training and testing; training was done with queries from the first two months, and testing was done with queries from the third month.

Relevance judgments are not available for AOL queries, thus result set *overlap* was used to evaluate search quality. Overlap assumes that the goal is to replicate exhaustive search, but with less effort [7].

It is defined as the agreement on documents retrieved by two search methods at rank k:

$$overlap_k = \frac{|D_k^{exh} \cap D_k^{sel}|}{k} \tag{12}$$

where D_k^{exh} and D_k^{sel} are the top k documents retrieved by exhaustive and selective search. High overlap is desired.

The overlap at ranks 10, 100, 500 and 1000 were examined using 10 system instances for each method. Results are presented in Figure 1.

The important trends are consistent for the two conditions: QKLD-Rand had higher overlap between selective search and exhaustive search than the baseline, and QKLD-QInit further increased overlap and reduced variance.

We observed two differences in the performance on the two types of queries. First, TREC test queries produced *greater overlap* than AOL test queries for all methods, including the baseline. We believe that this is due to the higher quality of TREC test queries, which do not contain urls, misspellings, and other noise. Second, the *relative improvement* of QKLD methods over KLD-Rand was slightly larger for the aligned condition (13% aligned vs 10% unaligned for CW09-B; similar results for Gov2). This difference is not surprising due to the greater similarity of training and testing data under the aligned condition. However, the difference is small and probably not statistically significant.

We conclude that QKLD-Rand and QKLD-QInit are equally effective under the aligned and unaligned conditions, and

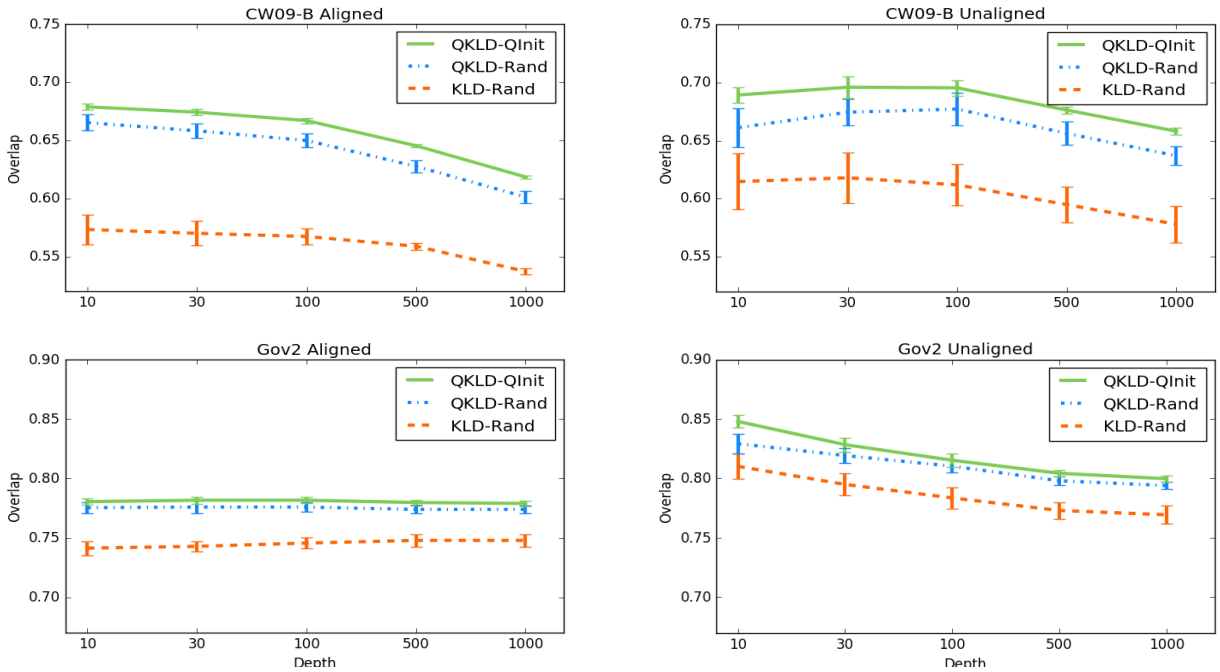


Figure 1: The overlap between exhaustive search and selective search results. The X axis represents the depth in the ranking. Lines show the average overlap of 10 selective search system instances with the exhaustive system. Error bars show the standard deviation over 10 instances. (Best viewed in color).

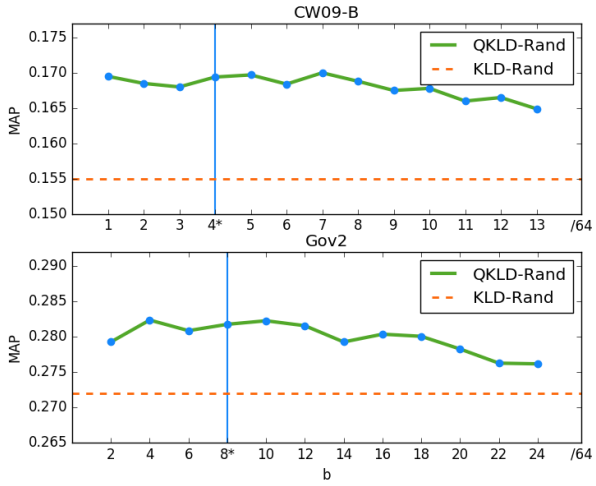


Figure 2: Sensitivity of search accuracy to parameter b in the query-biased similarity function (Equation 2). The vertical lines show the values of b used in all of the other experiments.

thus not sensitive to the temporal mismatch between the AOL query log and TREC queries. Using TREC queries and AOL queries to evaluate the quality of partitions created using AOL queries leads to the same conclusions.

6.3.2 *Sensitivity to Parameter b*

The key parameter in the `QKLD` similarity function (Equation 2) is b. We tested a range of values for b in order to understand its sensitivity to this parameter. Figure 2 presents MAP@1000 for `QKLD-Rand` shards generated with various values of b; X axis values are divided by 64 (e.g., $1/64, 2/64, \ldots$). MAP was fairly stable for low-to-medium ranges of b, with a large improvement compared to `KLD-Rand`. The bias from the query log becomes weaker when larger b is used. As a result, MAP starts to decrease when b is too large, converging to `KLD-Rand` behavior. These results demonstrate that the `QKLD` similarity function is not highly sensitive to b.

We chose the value of b used in the experiments by selecting the value in the middle of the interval where MAP@1000 was relatively stable. The chosen values of b are marked by vertical lines in Figure 2: $b = 4/64$ for CW09-B and $b = 8/64$ for Gov2. As shown in the graph, MAP@1000 was stable over a fairly wide range around the chosen b values. These values of b were used in all of our other experiments without additional tuning, to avoid overfitting.

6.4 Efficiency

Prior research shows that selective search is more efficient than distributed search using random partitions [2, 8, 9, 10, 11]. The next experiment investigates whether query-biased partitioning changes selective search efficiency.

We use an experimental methodology established by prior research, which distributed the Gov2 and CW09-B collections across two 8-core machines [8, 9, 11]. For exhaustive search, each machine core serves one index partition, thus the documents are distributed randomly across $2 \times 8 = 16$ partitions. For selective search, each machine core can serve multiple index partitions because just a few partitions are searched for any query; in our experiments, each core serves an average of 7 CW09-B shards or 12 Gov2 shards. Shards can be assigned to machines randomly or by a load balancing strategy [8]; the choice does not affect this experiment.

We use efficiency metrics proposed by Aly et al. [2]: C_{RES}

Table 10: Search costs of four distributed search methods. Search cost values are million documents evaluated per query.

	CW09-B		Gov2	
	C_{RES}	C_{LAT}	C_{RES}	C_{LAT}
`Exhaustive`	5.24	0.33	2.89	0.18
`KLD-Rand`	0.53	0.24	0.39	0.11
`QKLD-Rand`	0.52	0.24	0.39	0.11
`QKLD-QInit`	0.52	0.23	0.38	0.11

and C_{LAT}. C_{RES} calculates resource usage as the upper bound on the number of documents evaluated for query q:

$$C_{RES}(q) = |D^q_{CSI}| + \sum_{i=1}^{T} |D^q_{S_i}|. \quad (13)$$

C_{LAT} calculates query latency as the number of evaluated documents on the longest execution path for query q, assuming that shards are processed in parallel:

$$C_{LAT}(q) = |D^q_{CSI}| + \max_{i=1}^{T_q} |D^q_{S_i}|. \quad (14)$$

$|D^q_{CSI}|$ is the cost of the resource selection algorithm, calculated as the number of documents in its Central Sampled Index (CSI) that have at least one query term. $|D^q_{S_i}|$ is the number of documents in the i^{th} shard selected for query q that have at least one query term. T_q is the number of shards selected for query q. We report average values of C_{RES} and C_{LAT} over all queries on each dataset. Note that these metrics consider the number of *evaluated documents*. Another choice would be the number of query term postings processed. Both choices lead to the same conclusions.

Table 10 compares the search costs of `QKLD-Rand` and `QKLD-QInit` with the selective search baseline `KLD-Rand`. Exhaustive search results are shown for informational purposes, but they are irrelevant to this experiment.

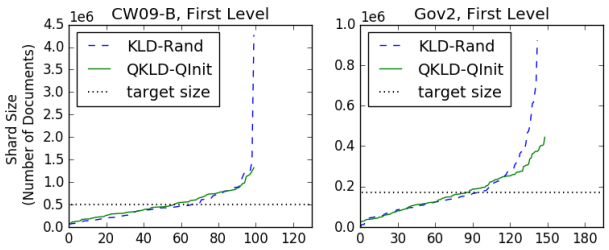
`QKLD-Rand` and `QKLD-QInit` partitioning produced query latency (C_{LAT}) and total search (C_{RES}) costs that are similar to the costs for `KLD-Rand` partitioning. Although search costs are slightly lower with `QKLD-QInit` partitioning, the differences are not statistically significant. We conclude that query-biased partitioning does not increase search costs.

Distribution of Shard Sizes: An even distribution of shard sizes makes it easier to balance computational loads across machines. `KLD-Rand` can produce an unbalanced distribution of shard sizes. For instance, most of the CW09-B Wikipedia pages are placed in one shard, whose size is about 8 times larger than the average shard size. Even if another round of clustering is used, the clustering algorithm may find it difficult to split large clusters [11].

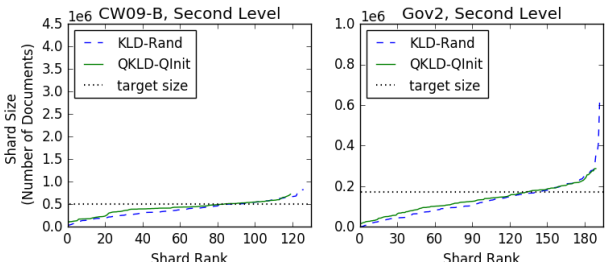
We investigated the effects of the query-biased partitioning on the distribution of shard sizes. Figure 3 compares the shard size distributions produced by `KLD-Rand` and `QKLD-QInit` during the first (Figure 3a) and second (Figure 3b) levels of k-means clustering.

After the first level of clustering on CW09-B, about 10% of the `KLD-Rand` shards were 2 times larger than the average-sized shard and needed to be split. These shards contained 25% of all the documents. Only about 3% of the `QKLD-QInit` shards needed to be split, containing only 7% of the collection. Similar results were observed on Gov2.

After two levels of clustering, `QKLD-QInit` shards were closer to the target size, with fewer too-small or too-big



(a) After the first level of clustering



(b) After the second level of clustering

Figure 3: Shard size distribution for shards created by `KLD-Rand` and `QKLD-QInit`. The X axis indicates the shards sorted by sizes. The Y axis represents shard size in number of documents. Averaged across 10 system instances.

shards. On CW09-B the standard deviation of `KLD-Rand` shard sizes is 1.8 million, and that of `QKLD-QInit` is 1.3 million. `QKLD-QInit` also had a 30% smaller standard deviation on Gov2 shards. Second round `QKLD-QInit` clustering costs were substantially lower than second round `KLD-Rand` clustering because fewer big clusters needed to be subdivided.

We conclude that query-biased clustering produces a more even distribution of shard sizes. `QKLD-QInit` is slightly better than `KLD-Rand` at avoiding clusters that are smaller than desired, but the main effect is that `QKLD-QInit` avoids producing most of the large clusters that `KLD-Rand` produces; and, the large clusters that `QLKD-QInit` does produce are easier to subdivide in a second round of clustering

7. CONCLUSIONS

This paper presents a query-biased method of partitioning a large document corpus for selective search. It introduces `QInit`, a query-driven clustering initialization method, to start the partitioning with topics that are important in search traffic. `QInit` uses word embeddings to discover topics in the query log and uses these topics as the seeds for document clustering. It also introduces a query-driven similarity metric, `QKLD`, that biases the clustering similarity metric toward terms that are important in a query log.

Query-biased similarity produced partitions that delivered significantly improved search accuracy compared to the previous state-of-the-art, even when clusters were seeded randomly. When it was combined with the query-biased cluster seeding, search accuracy was further improved. Accuracy was improved at low Recall (e.g., P@10), which is not surprising; selective search typically does well at low Recall. However, accuracy also improved at higher Recall (e.g., NDCG@100 and MAP@1000), which has been difficult for selective search architectures in the past.

Most selective search systems use `KLD-Rand` and other ele-

ments that introduce variance; different trials typically yield somewhat different results. `QKLD-QInit` reduced by half the variance of results obtained with different system instances, making the selective search architecture more predictable.

The main advantage of selective search over distributed search with random partitions is its efficiency. Query-biased partitioning retains that efficiency, and is also less likely to produce large index partitions that complicate load balancing and increase query latency.

Finally, query-biased clustering is not unduly sensitive to experimental conditions or tuning. Improvements were observed using training queries that were only loosely related to the test queries. Closer alignment between training and test queries provided a small gain in our experiments.

Query-biased partitioning improves the accuracy, stability, and reliability of a selective search architecture. We believe that the `QKLD` query-biased similarity metric can also be applied to resource selection, however that remains a topic for future research.

8. ACKNOWLEDGMENTS

This research was supported by National Science Foundation (NSF) grants IIS-1302206 and IIS-1422676, and a Google Research Award. Any opinions, findings, conclusions, and recommendations expressed in this paper are the authors' and do not necessarily reflect those of the sponsors.

9. REFERENCES

[1] I. S. Altingövde, E. Demir, F. Can, and Ö. Ulusoy. Incremental cluster-based retrieval using compressed cluster-skipping inverted files. *ACM Transactions on Information Systems*, 26(3), 2008.

[2] R. Aly, T. Demeester, and D. Hiemstra. Taily: Shard selection using the tail of score distributions. In *Proceedings of SIGIR*, 2013.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[4] M. E. Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

[5] C. L. Clarke, N. Craswell, and I. Soboroff. Overview of the trec 2004 terabyte track. In *TREC*, volume 4, page 74, 2004.

[6] Z. Dai, Y. Kim, and J. Callan. How random decisions affect selective distributed search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 771–774. ACM, 2015.

[7] J. C. French, A. L. Powell, F. C. Gey, and N. Perelman. Exploiting a controlled vocabulary to improve collection selection and retrieval effectiveness. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management*, pages 199–206, 2001.

[8] Y. Kim, J. Callan, S. Culpepper, and A. Moffat. Load-balancing in distributed selective search. In *Proceedings of the 39th International ACM SIGIR conference on on Research and Development in Information Retrieval*, pages ??–?? ACM, 2016.

[9] A. Kulkarni. *Efficient and Effective Large-scale Search*. PhD thesis, Carnegie Mellon University, 2013.

[10] A. Kulkarni and J. Callan. Document allocation policies for selective searching of distributed indexes. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM '10)*, pages 449–458. ACM, 2010.

[11] A. Kulkarni and J. Callan. Selective search: Efficient and effective search of large textual collections. *ACM Transactions on Information Systems (TOIS)*, 33(4):17, 2015.

[12] A. Kulkarni, A. Tigelaar, J. Callan, and D. Hiemstra. Shard ranking and cutoff estimation for topically partitioned collections. In *Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM'12)*, 2012.

[13] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *Proceedings of the 28th International ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM, 2005.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[16] B. Poblete and R. Baeza-Yates. Query-sets: using implicit feedback and query patterns to organize web documents. In *Proceedings of the 17th International Conference on World Wide Web*, pages 41–50. ACM, 2008.

[17] D. Puppin, F. Silvestri, and D. Laforenza. Query-driven document partitioning and collection selection. In *Proceedings of the 1st international conference on Scalable information systems*, page 34. ACM, 2006.

[18] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 298–305. ACM, 2003.

[19] P. Thomas and M. Shokouhi. Sushi: scoring scaled samples for server selection. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 419–426. ACM, 2009.

[20] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, volume 1, pages 577–584, 2001.

[21] E. P. Xing, M. I. Jordan, S. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

[22] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.