Dynamic Shard Cutoff Prediction for Selective Search

Hafeezul Rahman Mohammad*
Carnegie Mellon University
hmohamma@cs.cmu.edu

Keyang Xu*
Petuum Inc.
xky0714@gmail.com

Jamie Callan
Carnegie Mellon University
callan@cs.cmu.edu

J. Shane Culpepper
RMIT University
shane.culpepper@rmit.edu.au

ABSTRACT

Selective search architectures use resource selection algorithms such as Rank-S or Taily to rank index shards and determine how many to search for a given query. Most prior research evaluated solutions by their ability to improve efficiency without significantly reducing early-precision metrics such as P@5 and NDCG@10.

This paper recasts selective search as an early stage of a multi-stage retrieval architecture, which makes recall-oriented metrics more appropriate. A new algorithm is presented that predicts the number of shards that must be searched for a given query in order to meet recall-oriented goals. Decoupling shard ranking from deciding how many shards to search clarifies efficiency vs. effectiveness trade-offs, and enables them to be optimized independently. Experiments on two corpora demonstrate the value of this approach.

ACM Reference Format:
Hafeezul Rahman Mohammad, Keyang Xu, Jamie Callan, and J. Shane Culpepper. 2018. Dynamic Shard Cutoff Prediction for Selective Search. In *Proceedings of SIGIR '18*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210005

1 INTRODUCTION

Selective search is a distributed search architecture that avoids searching the entire corpus for each query. When the index is built, it is divided into small, topically-oriented shards. During retrieval, first shards (or *resources*) are ranked by their likelihood of returning documents relevant to the query, and then only the most query-relevant shards are searched. The accuracy and efficiency of the selective search architecture depends upon the number of shards that are searched (the *cutoff*). Searching too few shards harms accuracy, while searching too many shards harms efficiency.

While a large body of work now exists around this technology [16, 19–21, 23–25], many interesting problems remain. In this paper we focus on two related issues. First, distributed search is increasingly viewed as an early-stage retrieval process where the task is to efficiently collect as many *possibly relevant* documents *before* applying more expensive learning-to-rank algorithms [5, 28, 37, 44]. As such, optimizing for early precision metrics such as ERR [8] and NDCG@10 [17] during selective search may not be desirable. Second, the ideal number of shards to search can depend heavily on the resource selection algorithm, the desired search type (recall-driven or early-precision-driven), and the specific query.

In spite of the importance of selecting the right number of shards with respect to targeted evaluation, this aspect of selective search has not been studied extensively by prior research. Some approaches treat the cutoff as a parameter to be tuned for a query set; that is, the same value is used for every query [2, 4, 13, 23, 40, 41]. Other approaches treat it as part of the resource selection problem, where the result is a shard ranking and a cutoff. For example, SUSHI [45] selects shards that are expected to have documents in the top-*n* of the final ranking, Taily [1] sets the cutoff based on an estimate of the minimum number of relevant documents in each shard, and Rank-S [25] sets the cutoff using a rank-based decay function of the shard's relevance score. Query-based cutoffs produced by algorithms such as SUSHI, Taily, and Rank-S are appealing, however there has been little study of the prediction accuracy.

Another limitation in past work is the assumption that single-pass retrieval using BM25 or language models and focusing on early precision is sufficient. In this scenario, only a few shards are required for most queries, and less accurate cutoff predictions tend to not hurt efficiency. However, complex multi-stage ranking pipelines are now common [9, 37]. If selective search is used in a pipeline for early-stage retrieval, recall should be a priority [31], and maximizing for recall often means searching more shards initially.

Query-specific shard cutoff prediction – the problem of predicting the number of shards to search for each query – depends on the size of the desired result set. Thus, we distinguish between *early precision* and *high recall* search requirements. An *early precision* scenario measures accuracy in the first few ranked documents (e.g., 1 . . . 10), and thus is likely to be more efficient because fewer shards are searched. This is the scenario studied most often in prior work. In contrast, a *high recall* scenario attempts to find all relevant documents for a query, and can require thousands of documents to be returned. Thus, it is likely to require more shards to be searched. A robust shard cutoff prediction method should be effective, stable, and usable for both early precision and high recall search scenarios.

This paper presents a new, feature-based approach to query-specific shard cutoff prediction that is easily tuned for early precision or high recall, and can be used in conjunction with *any* resource selection algorithm. The research and experiments presented in this paper are designed to answer the following five research questions.

*The first two authors contributed equally.

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5657-2/18/07. . . $15.00
https://doi.org/10.1145/3209978.3210005

- **RQ1:** How accurate are existing shard cutoff predictions?
- **RQ2:** How accurate are existing shard rankings?
- **RQ3:** Are ranker-independent cutoff predictions effective?
- **RQ4:** How do the competing goals of precision-oriented and recall-oriented selective search affect tradeoffs between efficiency and effectiveness?
- **RQ5:** Is it necessary or useful for the shard cutoff prediction algorithm to be trained for a specific resource selection algorithm?

The next section reviews prior research on selective search, resource ranking, and measuring the similarity of search results. Section 3 describes our new approach to predicting shard cutoffs. Section 4 describes our experimental methodology and evaluation. Section 5 reports experimental results. Section 6 concludes.

2 RELATED WORK

Two types of prior research relate to query-specific shard cutoff prediction. Resource selection algorithms rank shards for a query; algorithms often used with selective search also make query-specific decisions about how many shards to search. Rank similarity measures are also related, since the goal of selective search is to produce rankings equivalent to exhaustive search, albeit with less effort.

2.1 Resource Selection for Selective Search

Resource selection (resource ranking) estimates the relevance of index shards for a specific query, and imposes an ordering on shard traversal. There are three general approaches to resource selection for selective search: *term-based*, *sampled-based* and *feature-based*.

Term-based methods use summary term statistics to model each shard. These are used to estimate the shard's relevance to a query [4, 15]. For example, Taily [1] uses the mean and variance of term frequency (*TF*) within a shard to estimate the number of its documents that would be highly-ranked by an exhaustive search system; shards that are likely to contain more than a specified number number of highly-ranked documents (e.g., $v = 50$) are selected.

Sample-based methods combine samples of documents from each shard into a common index, known as a centralized sample index (CSI). The query is run against the CSI. Each top-ranked document is treated as a vote for the shard from which it was sampled. Algorithms such as ReDDE [41], SUSHI [45], CRCS [40] and Rank-S [25] differ primarily in how they conduct voting. Markov and Crestani [33] and Sener et al. [39] provide detailed analyses of these algorithms. The ReDDE score for shard R is $n_R \cdot w_R$, where n_R is the number of documents that R contributed to the top-n of the CSI ranking, and w_R is the ratio of shard size to sample size. CRCS considers the rank of the document in the CSI ranking, so that documents at higher ranks contribute more than those at lower ranks. Rank-S uses an exponential function, $score_{CSI}(q, d_i) \cdot B^{-i}$, to discount the contribution of the i·th ranked document from the CSI ranking, where B controls the rate of decay. Rank-S selects all shards with scores above a threshold.

Feature-based methods use a variety of features, such as summary statistics, the scores of term-based algorithms, the scores of sample-based algorithms, the query's category, and presence of certain terms, to estimate a shard's relevance to the query. Binary classification [2], regression models [7], and learning-to-rank [12] have been used to learn the models.

Usually the number of shards to search is either a static parameter [2, 12, 41] or is tightly-integrated with the resource selection algorithm [1, 25]. One exception is ShRkC [22], a feature-based, regression-based shard cutoff predictor. Although independent of any resource selection algorithm, ShRkC is trained using data from a desired resource selection algorithm. As in most work on selective search, it was trained and evaluated for early-precision metrics.

2.2 Rank Similarity

Prior studies have also explored how to measure the similarity between two ranked result lists. These approaches can be used to compare search results without explicitly requiring relevance judgments [3, 14, 18, 26, 34, 42]. More recently, Webber et al. [46] proposed *Rank-Biased Overlap* (RBO), which calculates the expected average overlap that the user observes in comparing the two lists. The key difference between RBO and previous approaches is that a user bias to higher ranking documents is incorporated, and the lists being compared can be *disjoint*. This means the metric can compare incomplete rankings. This idea was generalized by Tan and Clarke [43] who showed that the idea can be used for any utility based evaluation metric. The resulting family of metrics, called Maximized Effectiveness Difference (MED), can be computed using any gain function, and target specific metrics such as ERR [8], DCG [17], or RBP [36]. Unlike previous approaches, MED directly transfers assumptions about user behavior from any chosen effectiveness measure to maximize a similarity score that serves as the corresponding rank similarity measure. For example, MED_{RBP} maximizes: $S(A) - S(B) = (1-\phi)(\sum_{i=1}^{K}(a_i - b_i)\phi^{i-1} + \sum_{i=K+1}^{\infty} \phi_{i-1})$, where A and B represent two ranking lists, K stands for the maximum depth for calculation and ϕ is the user persistence for RBP. In this work, we use MED_{RBP} in a manner originally described by Clarke et al. [10] to measure stagewise loss between an exhaustive run and the subset of documents aggregated by selective search, as this allows us to experiment with larger sets of queries that do not have relevance judgments associated with them.

3 QUERY-SPECIFIC SHARD CUTOFF PREDICTION

Query-specific shard cutoff prediction can be framed as a machine learning problem: Given a query q and a set of index shards S, train a model that can predict the number of shards K that should be searched to optimize a given metric. This framework requires defining a set of features that will provide clues about the number of shards to search for query q; obtaining training data; and selecting a machine learning algorithm. The metric to be optimized is assumed to be related to task requirements, for example, NDCG@10 (an *early precision* scenario) or MAP (a *high recall* scenario), and thus outside of our control. Among the different requirements, prior research provides the least guidance about the training data.

3.1 Features

The features used in this research are motivated by two ideas about what affects the number of shards to search for a query. The first idea is based on the premise that *difficult* queries need to search

Table 1: Term statistics used to generate query-dependent features. Statistics 2-8 are computed for all 7 similarity measures. Each term has 51 of these features in total.

Term Statistics
1. Number of documents containing the term
2. Maximum similarity score
3. First quartile similarity score
4. Third quartile similarity score
5. Arithmetic mean of similarity scores
6. Harmonic mean of similarity scores
7. Median of similarity scores
8. Variance of similarity scores
9. Geometric mean aggregation of all 49 scores in 2-8

Table 2: The 147 query-dependent features. The numbers in brackets show the number of features for that type. Feature types 2-8 are computed for all 7 similarity measures using term statistics from Table 1. Interquartile range is the difference between the first and third quartile similarity score.

Query Features
1. Query length (1)
2. Arithmetic mean of document frequency (1)
3. Arithmetic mean of Geometric mean aggregation (1)
4. Arithmetic mean of maximum scores (7)
5. Arithmetic mean of median score (7)
6. Arithmetic mean of mean scores (14)
7. Arithmetic mean of score variances (7)
8. Arithmetic mean of score interquartile ranges (7)
9. For each feature in Table 1, the minimum across terms (51)
10. For each feature in Table 1, the maximum across terms (51)

more shards while *simpler* queries do not. The second idea is that the number of shards being search also depends on the distribution of the similarity scores of a query with the documents across all the shards. Thus, two different types of features were investigated.

Corpus features describe how well query q matches the corpus. We incorporate 147 corpus features into our model which have previously been used for query difficulty prediction and other pre-retrieval tasks [6, 11, 30, 32]. These features describe query characteristics (e.g. length), and aggregated statistics for each query term (e.g. maximum score, harmonic/arithmetic mean/median score, and geometric mean) from a range of similarity functions (TF·IDF, BM25, query likelihood, term probability, Bose-Einstein, DPH, and DFR). When the index is constructed, features are easily pre-computed for each term. Table 1 outlines the term-specific features used in this work. During retrieval, features for query terms are fetched from a term dictionary and combined to produce simple, query-specific features. Table 2 shows the query-specific features used.

Shard-distribution features characterize the distribution of a shard-specific feature across the set of shards. For each term, three aggregated shard-level statistics (maximum, mean, and variance) are constructed using a range of similarity functions as above (TF·IDF, BM25, query likelihood, term probability, Bose-Einstein, DPH and DFR). For a given query, we find the arithmetic mean of these term-level statistics for each shard and then normalize the

Table 3: The 42 shard-distribution features. The numbers in brackets show the number of features for that type.

Shard Distribution Features
1. Entropy of arithmetic mean of mean scores across shards (7)
2. Entropy of arithmetic mean of maximum scores across shards (7)
3. Entropy of arithmetic mean of score variances across shards (7)
4. KL-Divergence of arithmetic mean of mean scores with a uniform reference distribution (7)
5. KL-Divergence of arithmetic mean of maximum scores with a uniform reference distribution (7)
6. KL-Divergence of arithmetic mean of score variances with a uniform reference distribution (7)

mean scores to form a valid probability distribution across all of the shards. The cross-entropy and KL-Divergence of these distribution scores across the set of shards form our feature set. We use a total of 42 shard distribution features (Table 3). We also investigated shard distribution features based on the maximum, cross-entropy, and KL-Divergence of Taily scores across shards. They performed about the same as the features in Table 3, and combining the two sets provided little gain, thus we omit those results. We surmise that the two sets of feature captured the same information.

These shard-distribution features provide better signals to the learning algorithm when changing the search goal from early-precision to recall-oriented. When the distribution is heavily skewed, the majority of the relevant documents will be concentrated into a few shards, and hence fewer shards can be searched; but when it is less skewed or more uniform, relevant documents will be scattered across many shards, and more shards should be searched.

3.2 Training Data

Typically selective search systems are compared to *exhaustive search* systems that search all shards. The goal of selective search is to deliver results that are at least as accurate, but at a lower (typically *much* lower) computational cost. In principle, it is possible for selective search to be more accurate than exhaustive search, but in prior research this behavior was observed only occasionally and only at early precision cut-offs (e.g. $1 \dots 5$). Outperforming exhaustive search in terms of effectiveness appears to be possible, but remains an elusive goal in practice. In this work, exhaustive search results are treated as "gold standard" results. If we are able to achieve the same effectiveness as exhaustive search while searching only a subset of the collection, efficiency is improved.

Here, a training instance is a tuple $(\Phi(q, S), K)$, where $\Phi(q, S)$ is a set of features extracted for query q and a set of shards S, and K is the number of shards to search. The value of K is determined by a three-step process. First, an exhaustive search for query q retrieves n documents. Second, a resource-selection algorithm produces a shard ranking for q. Third, the documents returned are analyzed to determine the number of shards K that must be searched to produce results comparable to the reference – exhaustive search. A complete discussion of exhaustive search is deferred until Section 4 because it is an experimental detail, and any ideal document ranking over the entire collection can be used in practice.

Shard Ranking. Shards can be ranked by resource selection algorithms such as ReDDE, Taily, and Rank-S. However, using a specific

```
FIND_K ($r_{d,e}$, $r_s$, $\epsilon$, $K_{max}$):
  # $r_{d,e}$:   A document ranking produced by exhaustive search
  # $r_s$:       A ranking of shards
  # $\epsilon$:  Maximum acceptable difference between doc rankings
  # $K_{max}$: A maximum value for $K$
  $K = 1$
  while ($K \leq K_{max}$) {
    Use the top $K$ shards in $r_s$ to create a document ranking $r_{d,K}$
    if (difference ($r_{d,K}$, $r_{d,e}$) $< \epsilon$)
      break;
    Increment $K$
return $K$
```

Figure 1: Algorithm to calculate the query-specific shard cutoff K.

resource selection algorithm makes the training data sensitive to that algorithm, which may have unintended consequences. For example, different algorithms produce shard rankings of different lengths. Taily ranks all shards; however, ReDDE, Rank-S, and other sample-document algorithms rank only the shards that contributed matching documents to a centralized sample index. The ranking may not contain all of the K shards necessary to produce a document ranking comparable to exhaustive search.

We define an algorithm that generates a shard ranking compatible with an exhaustive search document ranking. Given query q and exhaustive search ranking $r_{d,e}$, the weight of shard s is:

$$W_{q,s} = \sum_{i=1}^{depth} I_s\left(r_{d,e}[i]\right) * p^{i-1}$$

where I_s indicates whether document $r_{d,e}[i]$ is located in shard s, p is the user persistence from MED_{RBP}, and *depth* is the maximum depth for computing weights. We denote a ranking of shards by $W_{q,s}$ – a ranking compatible with exhaustive search – as $r_{s,e}$. The value of p and *depth* depend on the evaluation metric being targeted.

Some queries require *many* shards to be searched in order to achieve a $MED_{\text{RBP}} \leq \epsilon$. This occurs when index partitioning scatters a topic across many shards rather than concentrating it in a few shards, or when the query and relevant documents have little or no overlapping vocabulary. Usually the percentage of such queries is small; however they can have a disproportionate effect on the learning algorithm because the cutoff labels (K) for these queries differ so dramatically from the labels for other training data. In order to minimize the effect of such outliers, we set $K_{max} = 16$ in our recall-driven experiments, and $K_{max} = 8$ in our early-precision experiments. These parameters were chosen empirically.

Query-Specific Shard Cutoff. A shard cutoff K is calculated using a simple iterative algorithm, shown in Figure 1. Beginning with $K = 1$, the top K shards are used to produce a document ranking. The similarity of r_K, the document ranking produced with K shards, is compared to r_e, the exhaustive search document ranking produced by searching all shards. If the two rankings are sufficiently similar, or if a maximum has been reached, the algorithm stops and reports K. Otherwise, K is incremented and the next value is tested. As r_K is always a subset of r_e, an efficient implementation creates r_K by removing from r_e documents that are in unselected shards.

The similarity between two search lists, $r_{d,K}$ and $r_{d,e}$, is measured using MED_{RBP}. Clarke et al. [10] showed that the effectiveness loss between multi-stage retrieval results can be accurately measured without explicitly requiring relevance judgments. A small MED_{RBP} value indicates that $r_{d,K}$ agrees with exhaustive search and large value denotes that they are different. A threshold ϵ is used to find the label K. When the value of MED_{RBP} is lower than the threshold ϵ, it indicates the K shards are sufficient to generate a result comparable to exhaustive search. Previous experiments showed that $\text{MED}_{\text{RBP}} < 0.2$ correlates with no important difference between the two lists [10], so we use a target of $\epsilon < 0.2$.

Summary. Training data is produced using only queries, exhaustive search, a shard ranking, a target similarity metric, and a task-specific similarity threshold that adjusts training data for shallow or deep evaluation metrics. Relevance judgments for the queries are not required, making it easier to produce large, task-specific training data for this problem.

3.3 Learning Algorithms

The distribution of K cutoffs is expected to be skewed since selective search is effective at concentrating most topics in fewer shards than random allocations. However, skewed data can be difficult for some classes of machine learning algorithms, thus we explore the label distribution effects on two classes of regression algorithms.

Random Forest (RF) regression [27] was used to directly predict the value of K. A random forest is a meta regressor that fits multiple regression models on sub-samples of the data and uses averaging to improve the predictive accuracy and control over-fitting.

Quantile Regression (QR) is a modification of random forest regression that estimates the conditional median. This model better handles outliers in heavy-tailed distributions. Since the distribution of K can be heavily skewed for certain target metrics, this method tends to work well empirically. The parameter used to tune quantile regression is τ, which controls the relative importance of quantile loss and standard regression loss.

4 EXPERIMENTAL METHODOLOGY

Dataset. Experiments were conducted on two widely used dataset collections: ClueWeb09-B, which contains around 50 million web pages and Gov2 which contains around 25 million documents from the US government web domains. Stopwords were removed using the default Indri stoplist. Stemming was done using the Krovetz stemmer. For ClueWeb09-B, the spam documents were removed during document retrieval using Waterloo spam scores and the score threshold for filtering spam was set to 50.

Dataset Partitions. We used the QKLD-QInit partition defined by Dai et al. [12], which divides ClueWeb09-B into 123 shards and the Gov2 dataset into 199 shards of approximately equal size. These partitionings represent the state-of-the-art for these datasets for selective search, and are available on the authors' website[1].

Training and Testing Queries. For ClueWeb09-B, the 40,000 queries from the 2009 TREC Million Query Track (MQT) were filtered and used for training, and 200 queries from the 2009-2012 TREC Web Track (WT) were used for testing. Note that the 200 WT queries are contained in the original MQT query set, and were removed for the experiments, and queries with no matches in the index were also

[1]http://boston.lti.cs.cmu.edu/appendices/SIGIR2017-Zhuyun-Dai/

removed. In total, 1,140 queries were removed and the final training corpus was a set of 38,600 queries. For Gov2, the 20,000 queries from the 2007 and 2008 TREC Million Query Track were used for training, and 150 queries from the 2004 and 2005 TREC Terabyte track were used for testing. As above, testing queries and queries with no matches in the index were removed from the training set, leaving a training corpus of 19,800 queries.

Exhaustive Search (*Gold Standard*) Document Rankings. The full index was searched by the Indri search engine. For ClueWeb09-B, the unstructured queries were transformed into structured queries in the Indri query language using two techniques that have been effective in TREC evaluations: multiple representations, and sequential dependency models (SDM) [35]. For example, the 3-term query "`ape bat cat`" transformation is shown below.

```
#weight(
  α1 #combine( ape.title    bat.title    cat.title )
  α2 #combine( ape.inlink  bat.inlink  cat.inlink )
  α3 #weight(  β1 #combine( ape.body bat.body cat.body )
               β2 #combine( #1( ape.body  bat.body )
                             #1( bat.body  cat.body ) )
               β3 #combine( #uw8( ape.body  bat.body )
                             #uw8( bat.body  cat.body ) ) ) )
```

α_1, α_2 and α_3 control the weight given to each representation. β_1,β_2 and β_3 control the weight the sequential dependency model places on matching anywhere in the body, in bigrams (#1), and in 8-term unordered windows (#uw8) [35]. Parameters were determined using a parameter sweep on 200 queries from the TREC 2009-2012 Web Track topics for ClueWeb09-B. $\alpha_1 = 0.20, \alpha_2 = 0.05, \alpha_3 = 0.75, \beta_1 = 0.8, \beta_2 = 0.1$, and $\beta_3 = 0.1$. Note that it is acceptable to use test data to set the exhaustive search query template parameters ($\alpha_1, \ldots, \beta_3$) because in an operational environment the query template parameters would be carefully-tuned, fixed, and known.

For Gov2, the unstructured bag-of-words queries were transformed into more effective structured queries by the sequential dependency model (SDM) with parameters (0.85, 0.1, 0.05) [35]. Our goal was to construct a competitive exhaustive baseline that is used as the reference to measure an upper bound on the effectiveness loss in the selective search environment.

Metrics. Early-precision experiments used P@5, NDCG@10, and Overlap@100 to measure accuracy. Recall-oriented experiments used Mean Average Precision (MAP) at rank 1000, RBP with $p = 0.95$ and Overlap@5000. P@5, NDCG@10, and MAP are included to enable comparison with prior research. In general care should be taken when evaluating ClueWeb collections deeply [29]. After exploring many options, we found Overlap@n to be the best metric for evaluating selective search in an early-stage retrieval setting. Given rankings $r_{d,e}$ and $r_{d,s}$ of length n for exhaustive and selective search, Overlap@$n = Count(r_{d,e} \cap r_{d,s})/n$. Results for multiple queries are macro-averaged. We used this metric because it measures (only) how well selective search mimics exhaustive search. We assume that the first stage of retrieval is a filtering step, where the goal is to quickly find a set of candidate documents that will be reordered by later retrieval stages, for example, using learning-to-rank. Thus, the order of documents does not matter.

Table 4: Differences between labels and predicted cutoffs in Figures 2a-3b. Lower mean absolute error (MAE) and higher Pearson correlation coefficient (PCC) indicate better prediction.

ClueWeb09-B	Early-Precision					High-Recall				
	Rank-S	Taily	ShRkC	RF	QR	Rank-S	Taily	ShRkC	RF	QR
MAE	1.31	1.34	2.99	1.67	**1.14**	2.91	2.84	4.85	2.31	**1.94**
PCC	0.37	0.34	0.26	0.41	**0.44**	0.38	0.39	0.28	0.53	**0.64**

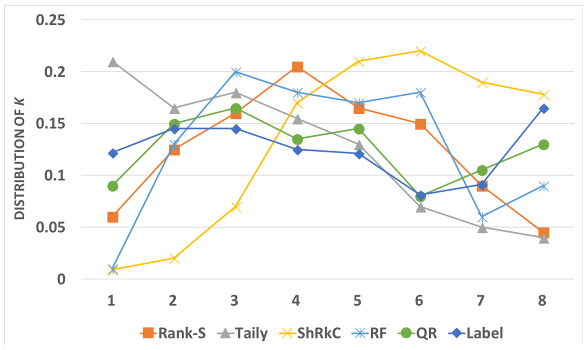
Gov2	Early-Precision					High-Recall				
	Rank-S	Taily	ShRkC	RF	QR	Rank-S	Taily	ShRkC	RF	QR
MAE	1.62	1.59	3.46	1.72	**1.42**	2.97	2.99	4.87	2.24	**2.12**
PCC	0.37	0.40	0.29	0.48	**0.52**	0.41	0.39	0.28	0.52	**0.59**

Efficiency was measured using C_{RES} and C_{LAT} [1]. C_{RES} calculates resource usage for a query as the upper bound on the number of documents that match. C_{LAT} measures query latency as the maximum number of documents that match in any selected shard. A two one-sided test (TOST) of equivalence [38] was used to compare results between exhaustive and selective search. The threshold for equivalence was set as $0.05 \cdot \mu$, where μ is the mean value of exhaustive search for a specific metric. Equivalence is established by rejecting the null hypothesis that selective search is at least 5% worse than exhaustive search with a 95% confidence interval.

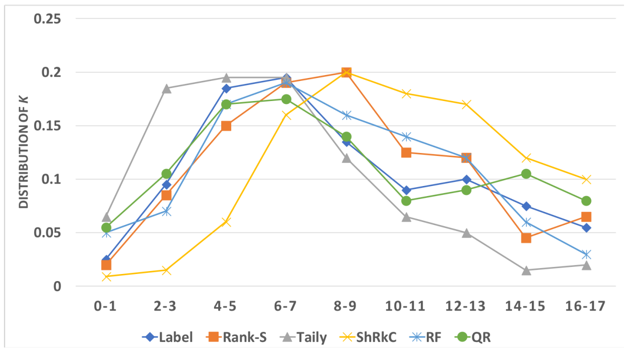
Baselines. Five resources selection methods were compared: Taily [1], ReDDE [41], Rank-S [25], learning to rank resources (L2RR) [13] and ShRkC [22], a random forest based shard cutoff predictor that uses ReDDE to generate the training data and shard cutoff estimator features. A sample size of 1% was used for CSI-based experiments. ShRkC's random forest predictor has two parameters, *mtry* (the number of features to sample at each split in the learning process), and *ntree* (number of decision trees to fit for ensemble learning). We used *mtry=p/3* where *p* is the total number of features, and *ntree=500*, which is consistent with prior work.

Shard Cutoffs. Rank-S and Taily compute query-specific shard cutoffs that are influenced by parameters. Taily's parameters include v, the estimated number of relevant documents. Rank-S uses B, which controls the exponential decay of scores. ReDDE and L2RR use a query-independent shard cutoff (also pre-defined). In our experiments, each resource selection algorithm used its own strategy, but the parameters were tuned to produce shard cutoffs compatible with early-precision and high-recall task requirements.

For early-precision search, we used $v = 25$ (Taily) and $B = 3.2$ (Rank-S) when searching ClueWeb09-B; and $v = 25$ (Taily) and $B = 2.6$ (Rank-S) when searching Gov2. These differ from the $v = 45$ and $B = 5$ parameter values used in most prior Rank-S and Taily studies [21]. In our setting, the default parameter values were less effective and easier to beat. 10-fold cross validation on the test set produced parameters that most often gave Taily and Rank-S the maximum performance. For recall-oriented selective search, which has not been studied previously, we found empirically that 8 shards for ClueWeb09-B, and 10 shards for Gov2 were usually enough to achieve a result comparable with exhaustive search. The parameters for Taily and Rank-S were tuned to search a similar number of

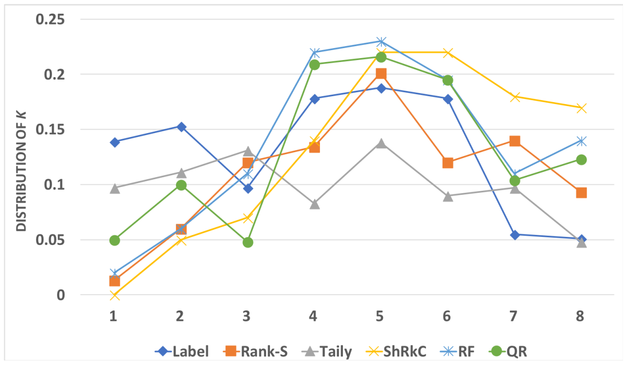


(a) Early-precision

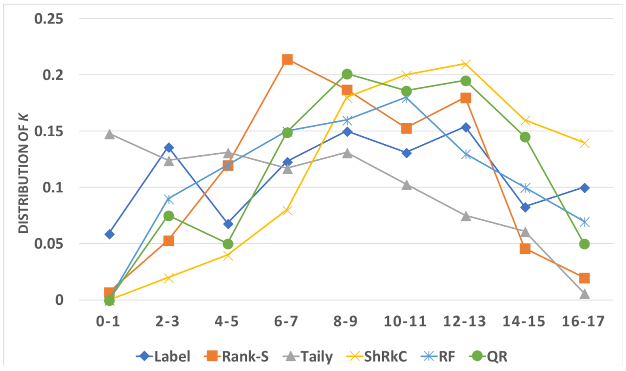


(b) High-recall

Figure 2: The distributions of shard cutoff predictions for ClueWeb09-B under early-precision and high-recall conditions. The x axis shows predicted cutoff values. The y axis shows the percentage of queries with each prediction.



(a) Early-precision



(b) High-recall

Figure 3: The distributions of shard cutoff predictions for Gov2 under early-precision and high-recall conditions. The x axis shows predicted cutoff values. The y axis shows the percentage of queries with each prediction.

shards for each of the datasets. For this search scenario, we used $v = 11$ (Taily) and $B = 1.6$ (Rank-S) for ClueWeb09-B and $v = 12$ (Taily) and $B = 1.6$ (Rank-S) for Gov2. The query-independent shard cutoffs used by ReDDE and L2RR were set based on the average number of shards searched by other resource selection methods in each search scenario for both the datasets.

Search Scenarios. Query-specific shard cutoff prediction was studied in early-precision and recall-oriented settings. The following parameters were used: p: The RBP user persistence; ϵ: The MED_{RBP} threshold; and *depth*: The depth at which the document rankings $r_{d,e}$ and $r_{d,s}$ are compared. Our precision-oriented search parameters were $p = 0.80$, $\epsilon = 0.08$, and $depth = 100$. Our recall-oriented search parameters were $p = 0.95$, $\epsilon = 0.06$, and $depth = 1,000$. A 5-fold cross validation was done on the test set to find the ϵ parameters that most often gave the maximum performance. Also, the variance in this parameter across several folds was almost negligible. It may seem counter-intuitive that early-precision uses a larger ϵ than high-recall; this is due to the sensitivity of MED_{RBP} to ranking depth. MED_{RBP} assumes that all documents deeper than *depth* are a mismatch, which is a worst-case scenario. When *depth* is increased, actual mismatches are accounted for. A good MED_{RBP} value is easier to achieve with higher *depth*, and harder for lower *depth*, thus ϵ and *depth* are inversely-related.

5 EXPERIMENTAL RESULTS

Four experiments investigated our research questions.

Cutoff Prediction Comparisons. First we investigated the accuracy of query-specific cutoff prediction methods used with existing resource selection methods (RQ1) and our new cutoff prediction methods (RQ3). Each method was tuned or trained for early-precision and recall-oriented search on the ClueWeb09 and Gov2 datasets, as described in Sections 3 and 4.

Table 4 shows the mean average error (MAE) and Pearson correlation coefficient (PCC) for each method. MAE differences don't seem large for early-precision, but QR is clearly better under high-recall. Pearson correlation coefficients vary from $[0.26 \ldots 0.29]$ (ShRkC) and $[0.34 \ldots 0.41]$ (Taily and Rank-S) to $[0.44 \ldots 0.64]$ (QR).

Figures 2 and 3 show how well each method matches the distribution of the ground truth cutoff labels (`Label`) determined by the FIND_K algorithm (Figure 1). Higher agreement indicates more accurate predictions. All methods are more accurate for the ClueWeb09 dataset than for the Gov2 dataset. In three out of four conditions, Taily is biased towards under-prediction. Rank-S, ShRkC, and RF over-predict the number of shards in all cases. QR is the most accurate of these methods at predicting shard cutoff labels; however, it over-predicts in three out of the four conditions.

ClueWeb09-B					
Early-Precision Oriented					
Shard Ranking	P@5	NDCG @10	Overlap @100	C_{RES}	C_{LAT}
Taily	.370	.214	.623	.508	.180
Rank-S	.375	.229	.673	.517	.178
ReDDE	.386	.229	.708	.551	.190
L2RR	.389	.234	.734	.560	.189
$r_{s,e}$	.409	.247	.818	.534	.187
Exhaustive	.390	.240	-	5.24	.330
High-Recall Oriented					
Shard Ranking	MAP @1000	$RBP_{0.95}$	Overlap @5000	C_{RES}	C_{LAT}
Taily	.180	.261 (.339)	.599	.811	.187
Rank-S	.181	.279 (.349)	.612	.811	.190
ReDDE	.182	.281 (.345)	.618	.853	.198
L2RR	.196	.293 (.304)	.626	.896	.199
$r_{s,e}$	.202	.301 (.286)	.709	.850	.195
Exhaustive	.202	.292 (.309)	-	5.24	.330

Gov2					
Early-Precision Oriented					
Shard Ranking	P@5	NDCG @10	Overlap @100	C_{RES}	C_{LAT}
Taily	.515	.340	.583	.191	.070
Rank-S	.519	.342	.620	.189	.069
ReDDE	.520	.363	.672	.237	.092
L2RR	.597	.442	.711	.193	.067
$r_{s,e}$	.602	.447	.832	.194	.071
Exhaustive	.612	.441	-	2.655	.282
High-Recall Oriented					
Shard Ranking	MAP @1000	$RBP_{0.95}$	Overlap @5000	C_{RES}	C_{LAT}
Taily	.229	.461 (.049)	.544	.346	.081
Rank-S	.233	.479 (.054)	.567	.324	.076
ReDDE	.244	.483 (.045)	.578	.373	.098
L2RR	.314	.499 (.034)	.619	.325	.071
$r_{s,e}$	.323	.517 (.031)	.701	.325	.077
Exhaustive	.339	.508 (.030)	-	2.655	.282

Table 5: Comparisons of shard ranking methods. RBP user persistence is 0.95. The number in brackets denotes the residual.

Note that the upward trend for `Label` at the right side of two figures is due to a long tail of queries with values above the $K = 8$ and $K = 16$ maximums for early-precision and high-recall conditions.

Shard Ranking Comparisons. The second experiment explores shard ranking accuracy independently of shard cutoff estimates (RQ2), because an algorithm's accuracy at ranking shards may not match its accuracy at predicting the cutoff. This experiment uses the labels produced by the Find_k algorithm (Figure 1) as the cutoff prediction for all shard ranking algorithms. Thus, the only difference is how accurately different methods rank shards. We include $r_{s,e}$, an ideal shard ranking produced from the exhaustive search document ranking (Section 3), to show the best-case scenario.

Rank-S and ReDDE may not generate a full shard ranking because they are sample-driven. Rank-S rankings are always less than or equal in length to ReDDE rankings, due to its exponential decay. When Rank-S returns fewer than K shards, its ranking is extended by appending shards from a ReDDE ranking, which performs similarly to Rank-S. When REDDE returns fewer than K shards, the Rank-S and ReDDE shard rankings are shorter than desired.

Table 5 shows the effectiveness and efficiency of each algorithm when searching the same number of index shards. L2RR produces rankings closest to exhaustive search, as measured by Overlap@*n* and $RBP_{0.95}$; its values are higher than Taily, Rank-S and ReDDE. It also generates more accurate document rankings for both early-precision and recall-oriented metrics.

The residual values for $RBP_{0.95}$ are high, which is expected in deeper evaluation scenarios when the relevance judgment pool depth is shallow. The number of unjudged documents is high in the recall-driven search scenario, and this effect should be explored further in future work. We have also observed that high residuals correlate with low accuracy scores, implying that shard ranking accuracy could be impacting the results.

All algorithms search the same number of shards, thus variations in C_{RES} and C_{LAT} are due to the sizes of selected shards. L2RR selects larger shards than Taily, Rank-S, and ReDDE for ClueWeb09-B, but smaller shards for Gov2. L2RR has many term-based features, thus it may favor shards with longer posting lists. This would be more likely to affect ClueWeb09, which has larger average shard sizes and a more skewed distribution of shard sizes.

Overall, L2RR balances effectiveness and efficiency better than Taily, Rank-S and ReDDE. Rank-S and ReDDE perform similarly and both outperform Taily. These results answer RQ2.

Early Precision versus Recall Driven Search. The third experiment investigated the document ranking accuracy and efficiency of each method under early-precision and recall-oriented conditions. Each method used its own shard rankings and query-specific (Rank-S, Taily) or query-independent (ReDDE, L2RR) shard cutoff predictions. This experiment also included ReDDE shard ranking with ShRkC cutoff prediction as proposed by Kulkarni [22], and L2RR shard ranking with the new quantile regression (QR) and random forest (RF) cutoff predictions (Section 3). Cutoff prediction accuracy was measured using mean absolute error (MAE) relative to the test labels. Table 6 summarizes the results.

Taily, Rank-S, ReDDE, and L2RR produce higher accuracy, overlap, and computational costs in this experiment than in the second experiment; this is not surprising. The second experiment used the Find_k shard cutoffs (Figure 1), which assumes perfect shard ranking; however, none of the rankers are perfect. When using their own shard cutoffs, they search more shards, which produces higher accuracy and overlap at higher computational expense.

Taily was the most efficient, as indicated by low C_{RES}; however, it was also the least accurate, as indicated by relevance and overlap metrics. This result is consistent with the first experiment, which showed that Taily frequently under-predicts the cutoff.

ReDDE with ShRkC cutoff predictions produces higher accuracy and overlap than ReDDE with query-independent cutoffs, but at higher computational cost. When shard rankers are inaccurate,

ClueWeb09-B							
Early-Precision Oriented							
	P@5	NDCG @10	Overlap @100	$\bar{K}$	MAE	C_{RES} (M)	C_{LAT} (M)
Taily	.371	.221	.645	4.52	1.34	.521	.189
Rank-S	.393*	.237*	.690	4.37	1.31	.570	.189
ReDDE	.391*	.229	.711	5	1.72	.559	.199
L2RR	.409*	.243*	.744	5	1.72	.559	.201
ShRkC	.410*	.244*	.752	7.21	2.99	1.07	.231
L2RR+QR$_{45}$	**.413***	**.249***	**.792**	4.4	**1.14**	.530	.190
L2RR+RF	.400*	.241*	.789	5.1	1.67	.590	.211
Exhaustive	.390	.240	-	123	-	5.24	.330
High-Recall Oriented							
	MAP	RBP$_{0.95}$	Overlap @5000	$\bar{K}$	MAE	C_{RES} (M)	C_{LAT} (M)
Taily	.173	.288(.318)*	.611	7.72	2.84	.841	.200
Rank-S	.182	.289(.308)*	.642	7.90	2.91	.843	.201
ReDDE	.181	.289(.338)*	.644	8	2.42	.871	.201
L2RR	.192*	.296(.317)*	.653	8	2.42	.893	.199
ShRkC	.197*	**.299(.308)***	.664	11.2	4.85	1.52	.230
L2RR+QR$_{45}$	**.198***	**.299(.308)***	**.706**	7.99	**1.94**	.872	.200
L2RR+RF	.193*	.294(.306)*	.703	8.60	2.31	.940	.205
Exhaustive	.202	.292 (.309)	-	123	-	5.24	.330

Gov2							
Early-Precision Oriented							
	P@5	NDCG @10	Overlap @100	$\bar{K}$	MAE	C_{RES} (M)	C_{LAT} (M)
Taily	.587	.400	.594	5.48	1.59	.204	.071
Rank-S	.590*	.411	.621	5.59	1.62	.231	.072
ReDDE	.582	.410	.683	5	1.58	.216	.068
L2RR	.595*	.438*	.721	5	1.58	.218	.071
ShRkC	.597*	.420*	.743	7.20	3.46	.276	.106
L2RR+QR$_{45}$	**.616***	**.446***	**.826**	5.46	**1.42**	.224	.069
L2RR+RF	.596*	.430*	.824	6.30	1.72	.250	.091
Exhaustive	.612	.441	-	199	-	2.655	.282
High-Recall Oriented							
	MAP	RBP$_{0.95}$	Overlap @5000	$\bar{K}$	MAE	C_{RES} (M)	C_{LAT} (M)
Taily	.291	.499(.044)*	.567	9.80	2.99	.341	.078
Rank-S	.300	.501(.042)*	.580	9.4	2.97	.353	.072
ReDDE	.301	.498(.058)*	.595	10	2.96	.391	.080
L2RR	.321*	.511(.049)*	.622	10	2.96	.381	.081
ShRkC	.320*	.511(.044)*	.632	12.1	4.87	.683	.118
L2RR+QR$_{45}$	**.325***	**.516(.035)***	**.690**	9.71	**2.12**	.355	.074
L2RR+RF	.324*	.511(.043)*	.682	10.02	2.24	.430	.087
Exhaustive	.339	.508 (.030)	-	199	-	2.655	.282

Table 6: Comparison of document ranking accuracy. $\bar{K}$ is the average number of shards searched. MAE is the mean absolute error in predicting the number of shards that should be searched. $*$ indicates statistical non-inferiority relative to exhaustive search. The best result for each metric is marked bold.

searching more shards improves accuracy but also increases costs. ShRkC greatly over-predicts shard cutoffs.

The L2RR, QR, and RF results use the L2RR shard ranking, but with different shard cutoff predictions. QR delivers the most accurate predictions, as measured by mean average error (MAE), which results in the highest accuracy and overlap values and some of the most efficient C_{RES} and C_{LAT} efficiency values. The RF predictor is less accurate than QR, which is consistent with the expectation that quantile regression is more effective when the underlying distribution is skewed.

QR was slightly more effective than exhaustive search for all relevance-based metrics in the early-precision experiments, and for RBP$_{0.95}$ in high-recall experiments. We don't want to over-emphasize these results, because beating exhaustive isn't the right goal for an early-stage ranker. However, these results remind us that it isn't necessary for Overlap@*n* to be 100% for selective search to deliver high-quality documents to the next stage rankers. Learned shard rankers with learned cutoffs are becoming very effective.

Figures 4a – 5b show effectiveness vs. efficiency tradeoffs among the different methods, and for QR with different values of its τ parameter. The x-axis shows resource usage (C_{RES}). The y-axis shows an early-precision or recall-oriented effectiveness metric. The goal is accuracy and computational costs close to $r_{s,e}$.

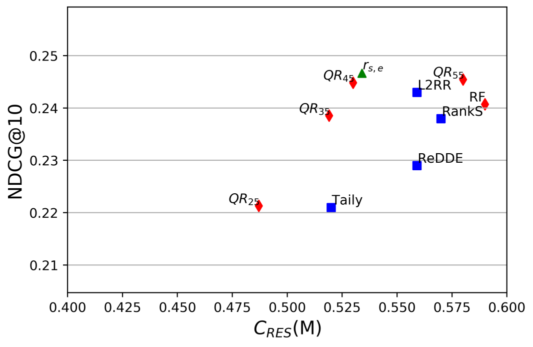
Taily is more efficient than Rank-S, ReDDE and L2RR but less effective. L2RR is more effective than Taily, Rank-S, and ReDDE, but usually computationally expensive.

Quantile regression's τ parameter enables tuning the efficiency vs. effectiveness tradeoff. A smaller τ focuses more on efficiency; a larger τ focuses more on recall. A reasonable range of parameter values produce better accuracy and efficiency than baseline algorithms. $\tau = 0.45$ – the value chosen using 10-fold cross-validation on the training set to minimize the mean average error (MAE) – best balances efficiency and effectiveness in both scenarios.

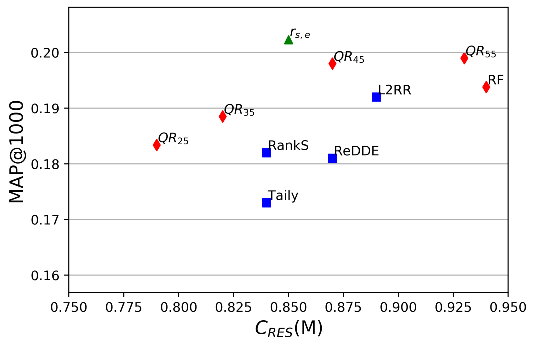
This third experiment shows that both quantile regression and random forests with the features described in Section 3 produce shard cutoff predictions that deliver substantially higher Overlap@*n* values than all baseline methods on both datasets under early-precision and high-recall conditions. QR gives a better mix of effectiveness and efficiency than RF, and is easily tuned to give greater control over the competing goals of efficiency and effectiveness.

Training Labels Comparisons. The experiments above use $r_{s,e}$, a shard ranking generated from exhaustive search results, to produce the 'gold standard' shard cutoffs used for training and testing. They show that those cutoffs can be too aggressive for less perfect shard ranking algorithms.

The Find_k algorithm (Figure 1) can use any shard ranking to generate training data for the QR predictor; it need not be $r_{s,e}$. This experiment investigates using shard rankings produced by Taily and L2RR ($r_{s,Taily}$ and $r_{s,L2RR}$) to train QR predictors that may be more compatible with those algorithms (RQ5). This experiment omits Rank-S and ReDDE because they are unable to generate complete shard rankings. Results are shown in Table 7. Exhaustive search results are shown for comparison, as in previous experiments.

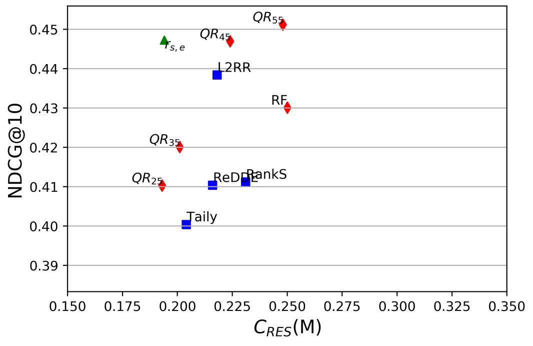


(a) Early-precision oriented search.

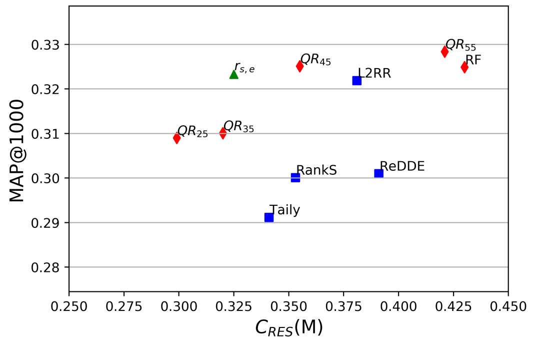


(b) High-recall oriented search.

Figure 4: Efficiency-effectiveness tradeoffs for ClueWeb09-B. $r_{s,e}$ indicates using test labels with $r_{s,e}$ shard rankings.



(a) Early-precision oriented search.



(b) High-recall oriented search.

Figure 5: Efficiency-effectiveness tradeoffs for Gov2. $r_{s,e}$ indicates using test labels with $r_{s,e}$ shard rankings.

Training with shard rankings matched to exhaustive search ($r_{s,e}$) always produces more aggressive shard cutoff predictions and much more efficient search than training with shard rankings produced by Taily and L2RR. This result is to be expected, because $r_{s,Taily}$ and $r_{s,L2RR}$ cannot be better orderings than $r_{s,e}$.

Combining Taily with a QR predictor trained for Taily is more effective across all relevance and overlap metrics than combining it with a general QR predictor trained from $r_{s,e}$; perhaps this is not surprising. However, combining L2RR with a QR predictor trained for L2RR is a little *less* accurate across most relevance and overlap metrics than combining it with a general QR predictor trained from $r_{s,e}$; in this case, pairing more accurate shard cutoff estimates with a more accurate shard ranker produces slightly better search results at much lower computational cost.

In answering RQ5, we conclude that if more accurate shard rankings (L2RR) are used, training with ranker-independent labels is more accurate. If less accurate resource selection algorithms are used, training with ranker-specific labels is more effective.

6 CONCLUSION

Previous studies treat selective search as a single stage retrieval method, and thus focus on optimizing early-precision metrics. We argue that selective search is a better choice for early-stage retrieval, thus evaluation should focus on high recall and how well selective search reproduces exhaustive search. We also argue that shard ranking and deciding how many shards to search should be studied separately, because they are separate sources of error.

There is substantial variation in the accuracy of shard cutoff decisions made by Taily and Rank-S, two algorithms often used for selective search; and by ShRkC, a newer shard cutoff predictor used with ReDDE. When attention is focused on ranking accuracy, Taily is the least accurate of the algorithms studied, the older ReDDE is surprisingly competitive, and the newer L2RR is the most effective.

This paper presents a new feature-based method of making query-specific shard cutoff decisions that can be trained for use with different shard ranking algorithms and/or tuned to satisfy early-precision or high-recall retrieval goals by adjusting (only) how training data is generated. The QR predictor produces higher agreement with exhaustive search results (Overlap@n) for Taily and L2RR than their default methods of predicting shard cutoffs.

Finally, although previous studies focused almost entirely on the accuracy of selective search at ranks 5-30, we show that selective search can deliver about 70% agreement with exhaustive search down to about rank 5,000, while requiring only 16-18% of the computational effort on two widely-studied datasets. These results support the argument that selective search is a good choice for early-stage retrieval in sophisticated multi-stage retrieval pipelines.

7 ACKNOWLEDGEMENTS

This research was supported by National Science Foundation (NSF) grant IIS-1302206 and the Australian Research Council's *Discovery Projects* Scheme (DP170102231). Any opinions, findings, and conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors.

ClueWeb09-B

Early-precision search

	Training Data	$\bar{K}$	P@5	NDCG @10	Overlap @100	C_{RES}	C_{LAT}
Taily	$r_{s,e}$	4.40	.371	.219	.643	.49	.175
Taily	$r_{s,\text{Taily}}$	7.89	.389	.221	.689	.88	.189
L2RR	$r_{s,e}$	4.40	.413	.245	.792	.53	.190
L2RR	$r_{s,\text{L2RR}}$	7.23	.412	.218	.772	.87	.193
Exhaustive		123	.390	.240	-	5.24	.330

Recall-oriented search

	Training Data	$\bar{K}$	MAP	$\text{RBP}_{0.95}$	Overlap @5000	C_{RES}	C_{LAT}
Taily	$r_{s,e}$	7.99	.180	.270 (.339)	.617	.83	.199
Taily	$r_{s,\text{Taily}}$	11.52	.189	.272 (.310)	.646	1.28	.199
L2RR	$r_{s,e}$	7.99	.198	.299 (.308)	.706	.84	.200
L2RR	$r_{s,\text{L2RR}}$	11.48	.199	.289 (.312)	.707	1.32	.201
Exhaustive		123	.202	.292 (.309)	-	5.24	.330

Gov2

Early-precision search

	Training Data	$\bar{K}$	P@5	NDCG @10	Overlap @100	C_{RES}	C_{LAT}
Taily	$r_{s,e}$	5.48	.516	.341	.592	.222	.072
Taily	$r_{s,\text{Taily}}$	7.46	.549	.342	.683	.284	.102
L2RR	$r_{s,e}$	5.46	.615	.447	.826	.224	.069
L2RR	$r_{s,\text{L2RR}}$	7.84	.615	.390	.812	.292	.103
Exhaustive		199	.612	.441	-	2.655	.282

Recall-oriented search

	Training Data	$\bar{K}$	MAP	$\text{RBP}_{0.95}$	Overlap @5000	C_{RES}	C_{LAT}
Taily	$r_{s,e}$	9.71	.230	.474 (.031)	.564	.342	.072
Taily	$r_{s,\text{Taily}}$	13.2	.297	.508 (.041)	.608	.712	.118
L2RR	$r_{s,e}$	9.71	.325	.516 (.035)	.690	.355	.074
L2RR	$r_{s,\text{L2RR}}$	12.9	.324	.498 (.039)	.699	.709	.118
Exhaustive		199	.339	.508 (.030)	-	2.655	.282

Table 7: A comparison of using different shard rankings to generate training data for the QR predictor.

REFERENCES

[1] R. Aly, D. Hiemstra, and T. Demeester. Taily: Shard selection using the tail of score distributions. In *Proc. SIGIR*, pages 673–682, 2013.

[2] J. Arguello, F. Diaz, J. Callan, and J. Crespo. Sources of evidence for vertical selection. In *Proc. SIGIR*, pages 315–322, 2009.

[3] C. Buckley. Topic prediction based on comparative retrieval rankings. In *Proc. SIGIR*, pages 506–507, 2004.

[4] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proc. SIGIR*, pages 21–28, 1995.

[5] B. B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, and J. Degenhardt. Early exit optimizations for additive machine learned ranking systems. In *Proc. WSDM*, pages 411–420, 2010.

[6] D. Carmel and E. Yom-Tov. *Estimating the Query Difficulty for Information Retrieval.* Morgan & Claypool, 2010.

[7] S. Cetintas, L. Si, and H. Yuan. Learning from past queries for resource selection. In *Proc. CIKM*, pages 1867–1870, 2009.

[8] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proc. CIKM*, pages 621–630, 2009.

[9] R.-C. Chen, L. Gallagher, R. Blanco, and J. S. Culpepper. Efficient cost-aware cascade ranking in multi-stage retrieval. In *Proc. SIGIR*, pages 445–454, 2017.

[10] C. L. A. Clarke, J. S. Culpepper, and A. Moffat. Assessing efficiency–effectiveness tradeoffs in multi-stage retrieval systems without using relevance judgments. *Inf. Retr.*, 19(4):351–377, 2016.

[11] J. S. Culpepper, C. L. A. Clarke, and J. J. Lin. Dynamic cutoff prediction in multi-stage retrieval systems. In *Proc. ADCS*, pages 17–24, 2016.

[12] Z. Dai, C. Xiong, and J. Callan. Query-biased partitioning for selective search. In *Proc. CIKM*, pages 1119–1128, 2016.

[13] Z. Dai, Y. Kim, and J. Callan. Learning to rank resources. In *Proc. SIGIR*, 2017.

[14] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. *SIAM J. Discrete Math.*, 17(1):134–160, 2003.

[15] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss: Text-source discovery over the internet. *ACM Trans. Database Systems*, 24(2):229–264, 1999.

[16] F. Hafizoglu, E. C. Kucukoglu, and I. S. Altingovde. On the efficiency of selective search. In *Proc. ECIR*, pages 705–712, 2017.

[17] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Information Systems*, 20(4):422–446, Oct. 2002.

[18] M. G. Kendall. *Rank correlation methods.* Griffin, 1948.

[19] Y. Kim, J. Callan, J. S. Culpepper, and A. Moffat. Load-balancing in distributed selective search. In *Proc. SIGIR*, pages 905–908, 2016.

[20] Y. Kim, J. Callan, J. S. Culpepper, and A. Moffat. Does selective search benefit from WAND optimization? In *Proc. ECIR*, pages 145–158, 2016.

[21] Y. Kim, J. Callan, J. S. Culpepper, and A. Moffat. Efficient distributed selective search. *Inf. Retr.*, 20(3):221–252, 2017.

[22] A. Kulkarni. Shrkc: Shard rank cutoff prediction for selective search. In *Proc. SPIRE*, pages 337–349, 2015.

[23] A. Kulkarni and J. Callan. Document allocation policies for selective searching of distributed indexes. In *Proc. CIKM*, pages 449–458, 2010.

[24] A. Kulkarni and J. Callan. Selective search: Efficient and effective search of large textual collections. *ACM Trans. Information Systems*, 33(4):17:1–17:33, 2015.

[25] A. Kulkarni, A. S. Tigelaar, D. Hiemstra, and J. Callan. Shard ranking and cutoff estimation for topically partitioned collections. In *Proc. CIKM*, pages 555–564, 2012.

[26] R. Kumar and S. Vassilvitskii. Generalized distances between rankings. In *Proc. WWW*, pages 571–580, 2010.

[27] A. Liaw and M. Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[28] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends in Inf. Ret.*, 3 (3):225–331, 2009.

[29] X. Lu, A. Moffat, and J. S. Culpepper. The effect of pooling and evaluation depth on IR metrics. *Inf. Retr.*, 19(4):416–445, 2016.

[30] C. Macdonald, N. Tonellotto, and I. Ounis. Learning to predict response times for online query scheduling. In *Proc. SIGIR*, pages 621–630, 2012.

[31] C. Macdonald, R. L. T. Santos, and I. Ounis. The whens and hows of learning to rank for web search. *Inf. Retr.*, 16(5):584–628, 2013.

[32] J. Mackenzie, J. S. Culpepper, R. Blanco, M. Crane, C. L. A. Clarke, and J. Lin. Query driven algorithm selection in early stage retrieval. In *Proc. WSDM*, pages 396–404, 2018.

[33] I. Markov and F. Crestani. Theoretical, qualitative, and quantitative analyses of small-document approaches to resource selection. *ACM Trans. Information Systems*, 32(2):9:1–9:37, 2014.

[34] M. Melucci. Weighted rank correlation in information retrieval evaluation. In *Proc. AIRS*, pages 75–86, 2009.

[35] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *Proc. SIGIR*, pages 472–479, 2005.

[36] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Information Systems*, 27(1):2:1–2:27, 2008.

[37] J. Pedersen. Query understanding at Bing. *Invited talk, SIGIR*, 2010.

[38] D. J. Schuirmann. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Pharmacodynamics*, 15(6):657–680, 1987.

[39] E. Sener, I. H. Toroslu, and I. S. Altingovde. An analysis of resource selection with rank cut-off estimation in meta-search. In *SIGIR Workshop on Heterogeneous Information Access*, 2016.

[40] M. Shokouhi. Central-rank-based collection selection in uncooperative distributed information retrieval. In *Proc. ECIR*, pages 160–172, 2007.

[41] L. Si and J. P. Callan. Relevant document distribution estimation method for resource selection. In *Proc. SIGIR*, pages 298–305, 2003.

[42] M. Sun, G. Lebanon, and K. Collins-Thompson. Visualizing differences in web search algorithms using the expected weighted hoeffding distance. In *Proc. WWW*, pages 931–940, 2010.

[43] L. Tan and C. L. A. Clarke. A family of rank similarity measures based on maximized effectiveness difference. *Trans. on Know. and Data Eng.*, 27(11):2865–2877, 2015.

[44] N. Tax, S. Bockting, and D. Hiemstra. A cross-benchmark comparison of 87 learning to rank methods. *Inf. Proc. & Man.*, 51(6):757–772, 2015.

[45] P. Thomas and M. Shokouhi. SUSHI: scoring scaled samples for server selection. In *Proc. SIGIR*, pages 419–426, 2009.

[46] W. Webber, A. Moffat, and J. Zobel. A similarity measure for indefinite rankings. *ACM Trans. Information Systems*, 28(4):20:1–20:38, 2010.