Balancing Precision and Recall with Selective Search

Mon Shih Chuang
Department of Computer Science
San Francisco State University
1600 Holloway Ave, San Francisco
CA, USA, 94132
mchuang@mail.sfsu.edu

Anagha Kulkarni
Department of Computer Science
San Francisco State University
1600 Holloway Ave, San Francisco
CA, USA, 94132
ak@sfsu.edu

Abstract

This work revisits the age-old problem of balancing search precision and recall using the promising new approach of Selective Search which partitions the document collection into topic-based shards and searches a select few shards for any query. In prior work Selective Search has demonstrated strong search precision, however, this improvement has come at the cost of search recall. In this work, we test the hypothesis that improving the effectiveness of shard selection can better balance search precision and recall. Toward this goal we investigate two new shard selection approaches, and conduct a series of experiments that lead to three new findings:- 1. Big-document based shard selection approaches can substantially outperform the small-document approaches when provided with richer query representation, 2. Applying Learning-To-Rank approach for shard ranking provides the most effective Selective Search setup, 3. If the relevant documents for a query are spread across less than 10% of the shards then Selective Search can successfully balance precision and recall.

1 Introduction

The problem of balancing search precision and recall is not new to the IR (Information Retrieval) community. Improving one, almost always results in degrading the other. The most prominent example of IR, Web search, makes it seem that search precision is more important than recall. However, there are many other use cases of IR where search recall is equally, if not more, important than precision. Medical or health information retrieval, prior-art search, and legal information retrieval are a few examples of the same. Motivated by these observations, we take a fresh look at the problem of balancing search precision and recall through the lenses of a promising new approach of *Selective Search* (Kulkarni and Callan, 2015). Selective Search is a distributed query processing approach that has been shown to improve search efficiency tremendously while sustaining the effectiveness. To accomplish this, at indexing time Selective Search partitions the document collection into shards based on document similarity. The resulting shards are topically homogeneous, that is, documents about the same or related topics are in the same shard. At query time, Selective Search exploits this topical organization by restricting query processing to a select few shards. This is contrary to the traditional distributed query processing approach where the query is processed at all the shards (*Exhaustive Search*). This is needed because Exhaustive Search uses *random shards* where documents are allocated to shards at random. As a result, the relevant documents to a query may be spread across many (or all) shards. For topical shards, however, the relevant documents for a query are likely to be concentrated into a few (or one) shards because these documents are typically similar to each other (Xu and Croft, 1999). These shards that are likely to contain relevant documents to the query are identified using *shard ranking* algorithms.

We believe that the topical organization of the document collection along with the selective nature of this search approach, can support a search environment that can balance precision and recall. We test this hypothesis in this paper through a series of experiments where we apply well-established shard ranking approach, and propose improvements to these algorithms that leverage the topic-based organization of the documents. Selective Search has consistently demonstrated good performance on precision-oriented metrics.

The reason for this trend is the purity of the search space. The few topical shards that are selected and searched for a query contain much less noise, that is, false-positive documents, than the complete collection. A purer search space reduces the chances of non-relevant documents being included in the search results, which directly improves the precision. In order to improve recall, Selective Search needs to identify all the shards containing relevant documents. Thus when optimizing for recall, more than a few shards need to be searched. As such the accuracy of shard ranking algorithm at deeper ranks also becomes critical. This observation suggests that an effective shard ranking algorithm would be able to improve search recall without degrading precision. We test this hypothesis thoroughly using empirical evaluation as early and deeper ranks. Finally, we introduce a novel shard ranking approach, *Learning to Rank Shards* (LeToR-S), that is based on the successful Learning-to-Rank document approach (Qin et al., 2010).

This paper is organized as follows. The prior work that has informed and influenced our work is described in the next section. The proposed shard ranking approaches are described in Sections 3 and 4. The experimental setup used for evaluation is described in Section 5, followed by Results and Analysis, Section 6. The conclusions we draw from this work are provided in Section 7.

2 Related Work

Selective Search falls under the subfield of distributed information retrieval (Callan, 2002; Shokouhi et al., 2011) where the goal is to search across multiple existing resources, and to aggregate the results. One of the important research problems in distributed IR is that of ranking the resources (shards) based on the number of relevant documents they contain for the query. A long line of research has studied this problem. The proposed algorithms can be roughly categorized into two groups: Big-document approaches, which are based on term frequency statistics of the resources. And Small-document approaches, which are based on a small sample of documents from each resource. Algorithms from both these categorize are described next.

2.1 Big-document approaches

In cooperative environments, the providers of target resources are willing to share the term statistics and metadata of the documents collections. The methods ranking resources based on the term statistics are called big-document approaches since the model can be extended from document retrieval model by treat each resource as a entity.

GlOSS(Gravano et al., 1994), and its extended versions gGloss (Gravano and Garcia-Molina, 1999) and vGlOSS(Gravano et al., 1999) provide a solution of text database discovery problem. The algorithms uses the ratio of term frequency and database size, also other metadata like field information(title, body, links) for choosing the candidate resources.

CORI(Callan et al., 1995)(Callan, 2002) algorithm keeps document frequency(df) and shard frequency(sf) of each term, and computes the score of every shard by a variation of tf.idf formula.

$$T = \frac{df_i}{df_i + 50 + 150 * sw_i/avg\_sw} \quad (1)$$

$$I = \frac{log(\frac{S+0.5}{sf})}{log(S+1.0)} \quad (2)$$

$$Score(t_k|S_i) = b + (1-b) * T * I \quad (3)$$

df_i: the document frequency of the term t_k in shard i.

sf: the shard frequency of the term t_k (The number of shards contain t_k).

sw_i: the number of total words in the shard i.

avg_sw: the average number of total words in one shard.

S: the number of shards.

t_k: the kth term in the user query.

b: the minimum belief component, set to 0.4

CORI inherited the query operators from INQUERY(Callan et al., 1992) document retrieval system which based upon Bayesian inference network model. The operator set used by INQUERY [sum, wsum, and, or, not] can work unchanged for ranking both documents and databases.

Taily(Aly et al., 2013) is another big document approach. According to Kanoulas et al's work (Kanoulas et al., 2010), the term frequency based document score across whole collection can be modeled by gamma curve distribution. Thus, Taily pre-computes two parameters scaler θ and K of gamma distribution to fit the document score for every single-term query against every shard. By storing the score distribution of single-term query against every shard, it can estimate the score distribution of user query with multiple terms.

2.2 Small-document approaches

In uncooperative environments, the important statistics such as term frequency or collection size can not be obtained. Therefore, big-document approaches are not capable to compute the shard scores. Small-document algorithms solve this issue by approximating the document distribution inside a resource by sampling a small subset, which is called centralized sample database, or centralized sample index (CSI) structure.

The ReDDE(Si and Callan, 2003) algorithm runs the user query against the CSI and assumes that the top n retrieved documents are relevant. The original version of ReDDE compute a score for each shard as follow equation:

$$Score(S_i^q) = Count(n, S_i^q) \times \frac{|S_i^{CSI}|}{|S_i|} \qquad (4)$$

$Count(n, S_i^q)$ is the count of documents occurred in top n retrieved documents in CSI $|S_i|$ is the size of the shard and $|S_i^{CSI}|$is the size of its sample . The shard scores are then normalized to obtain a valid probability distribution used to rank the shards.

CRCS(Shokouhi, 2007) passes the user queries to CSI and compute the score of each resource from the returned document rank. Two version of CRCS are introduces by modifying the function of rank. CRCS(1) uses a simple linear decreasing model, and CRCS(e) uses an exponential decaying model for the document score.

SUSHI(Thomas and Shokouhi, 2009) passes the user queries to CSI and uses the returned document scores to estimate the score distribution. For each shard, SUSHI fits one of three types of distribution curves, linear, logarithmic, and exponential to the scores of returned documents in CSI.

3 CORI_Uni+Bi & ReDDE_Uni+Bi

The scenario in which Selective Search operates is cooperative, that is, CORI has access to the complete term statistics for each shard. While ReDDE estimates the shard ranking based on the small sample of each shard. As such, one would expect CORI to perform at least on par, if not outperform, ReDDE. However, the term statistics that CORI uses, although complete are highly misleading. This is so because traditional CORI uses unigram model where each query term is treated as an individual entity. This leads to a very poor representation of the query. For example, the query *arizona fish and game* when decomposed into unigrams, looses the central topic *fishing in arizona*. CORI cannot distinguish between shards are on topic, and shards that contain documents about *arizona* in some other context, and about *fishing* in some other state.

These observations motivated the following investigation were a richer representation of the query is used by both CORI and ReDDE. Given a query with n terms, $n - 1$ bigrams are generated by enumerating all pairs of consecutive terms. Bigrams with stopwords are discarded, and the remaining bigrams are added to the original unigram query. As an example, for query *obama family tree*, this approach generates the following query representation using the Indri Query Language: *#combine(obama family tree #uw2(obama family) #uw2(family tree))*, where the *#combine* operator coalesces the scores from all the element of the query, and the *#uwX* operator is used for specifying an unordered phrase of length X. The ReDDE_Uni+Bi runs the query generated using the above procedure against the CSI, and the rest of the search process is same as before. In case of CORI_Uni+Bi, frequency statistics for bigrams, in addition to unigrams, are used in order to evaluate the richer query representation. Since the bigram statistics can be precomputed off-line, the response time of shard ranking, and query evaluation is not affected. Single-term queries or the queries containing no phrases because of stop words (e.g. ”to be or not to be”), remain unchanged. Higher-order n-grams, such as, trigrams were not included due to two reasons: 1. cost of the computing the statistics for trigrams is substantially high, and 3. the benefits from trigram representation are expected to be marginal because most queries are short, two or fewer terms. In search scenarios where the user queries are longer (legal or medical retrieval), trigram query representation could be worth the additional cost. This is part of future work.

4 Learning to Rank Shards (LeToR-S)

The Learning-to-Rank approaches have been successfully used to improve the document ranking task (Freund et al., 2003)(Xu and Li, 2007)(Burges, 2010). We wished to investigate if a ranking model can be learned for the shard ranking task as well. To test this intuition we started by defining the set of features for the ranking model, and the target variable. The number of relevant documents in the shard for the query, is used as the target. Instead of using integer rank values, this target captures more information. The shards are ranked in the descending order of the predicted target value.

LeToR-S Features

The different fields or sections of the document inform the features. The *title, body, heading, url, whole document* each generate a separate set of features. Several variants of CORI scores for the query are evaluated against the document fields, and are used as features. More specifically, CORI_SUM, CORI_MIN, and CORI_VAR are the three variants of CORI as defined in Equations 5 through 7. Each of these scores is computed for two different query representations:- unigram, and phrasal, against all of the different document fields.

$$CORI\_SUM(Q|S_i) = \sum_{t_k \in Q} Score(t_k|S_i) \quad (5)$$

$$CORI\_MIN(Q|S_i) = \min_{t_k \in Q} Score(t_k|S_i) \quad (6)$$

$$\mu = \frac{1}{n} \sum_{t_k \in Q} Score(t_k|S_i) \quad (7)$$

$$CORI\_VAR(Q|S_i) = \sum_{t_k \in Q} (Score(t_k|S_i) - \mu)^2 \quad (8)$$

Where $Score(t_k|S_i)$ is the same as Equation 3, and $\mu = \frac{1}{n} \sum_{t_k \in Q} Score(t_k|S_i)$

For each query and shard pair, the above vector of features is compiled, in order to learn the ranking model or to predict the ranking using the RandomForest(Breiman, 2001) model implemented by RankLib(Dang, 2013) from Lemur Project(Croft and Callan, 2000).

5 Experimental Setup

For the empirical evaluation the experimental setup that was undertaken is described next. We use the CategoryB dataset of ClueWeb09, which contains 50,220,423 documents. The 92 topical shards created for CategoryB dataset by Kulkarni and Callan(Kulkarni et al., 2012) are used in this work. The evaluation queries are from TREC Web Track 2009-2012. Out of the 200 queries, 6 queries do not contain any relevant document in this dataset, and thus are discarded. The remaining 194 queries are divided into 10-fold for the LeToR-S experiment to facilitate 10-fold cross validation. For the small-document approach, ReDDE, we construct the CSI by randomly sampling 0.5% of the documents from every shard. For all the ReDDE experiments the same CSI was employed, to minimize any spurious effects caused by sampling. The search engine used in our experiment is Indri 5.9 (Strohman et al., 2005) from Lemur Project. For all the Selective Search experiments reported in Sections 6.1 through 6.5, the top 10 shards were searched for each query. This corresponds to a search space of about 5.5 million documents. This is an order of magnitude smaller than the search space of Exhaustive Search (50+ million documents).

6 Results and Analysis

This section describes the experiments that were conducted to test the various hypotheses about improving search effectives by leveraging topical shards. Table 1 presents a comprehensive set of results for four different investigations that we undertook. We describe these next in the following subsections, and analyze the results in Table 1.

6.1 Big-Document versus Small-Document Shard Ranking Approach

Traditionally the choice of small- or big-document approach was dictated by the type of search environment:- uncooperative, or cooperative, respectively. Although, our scenario would categorize as cooperative, we choose to also experiment with small-document approaches because the conventional belief has been that small-document approaches provide superior search effectiveness than big-document approaches. This first experiment empirically tests this belief, specifically in the context of topical shards. We also compare Exhaustive Search with Selective Search with big-document, and small-document approaches. The first three rows in Table 1 are the focus of this analysis.

As compared to Exhaustive Search, CORI and

#	#Qrys	Run	P@30	P@100	map	R@30	R@100	ndcg
1	194	Exh	0.254	0.189	0.181	0.174	0.374	0.429
2	194	CORI	0.255	$\underline{0.178}$	$\underline{0.164}$	0.158	$\underline{0.329}$	$\underline{0.370}$
3	194	ReDDE	0.256	$\underline{0.182}$	$\underline{0.172}$	0.175	0.360†	$\underline{0.400}$†
4	52 (len=1)	CORI	0.260	0.189	0.163	0.132	0.296	$\underline{0.391}$
5	52 (len=1)	ReDDE	0.250	0.184	0.155	0.131	0.295	$\underline{0.387}$
6	142 (len>1)	CORI	0.254	$\underline{0.173}$	$\underline{0.164}$	$\underline{0.167}$	$\underline{0.341}$	$\underline{0.362}$
7	142 (len>1)	ReDDE	0.258	$\underline{0.181}$	$\underline{0.178}$	0.191†	0.384†	$\underline{0.404}$†
8	194	CORI_Uni+Bi	0.271§‡	0.188§	0.181§	0.180§	0.363§	$\underline{0.408}$§
9	194	ReDDE_Uni+Bi	0.263	0.184	0.175	0.182	0.363	$\underline{0.398}$
10	194	LeToR-S	0.270‡	0.194¶	0.186¶	0.179	0.377¶	$\underline{0.417}$¶

Table 1: Search Effectiveness for Exhaustive Search, and for Selective Search with CORI and ReDDE under various configurations. † indicates statistically significant improvements when comparing ReDDE with CORI. § indicates statistically significant improvement when comparing MTD_Uni+Bi with MTD. ‡ indicates statistically significant improvement over Exhaustive Search. ¶ indicates statistically significant improvement when comparing LeToR-S with MTD_Uni+Bi. Underline indicates significantly worse values when compared to Exhaustive Search. Statistical significance testing was performed using paired T-test at $p<0.05$.

ReDDE, both struggle at deeper ranks. CORI, the big document approach, is consistently inferior to ReDDE, the small document approach, across all the metrics. In fact, at deeper ranks (R@100 and ndcg) the improvements over CORI, with ReDDE are statistically significant. These results confirm that the conventional unigram language model based shard ranking approach adopted by CORI struggles to differentiate the relevant shards from non-relevant shards. This is so even for topical shards where the distribution of relevant documents across shards is highly skewed. Also, note that CORI has access to the vocabulary of the complete collection whereas ReDDE is only using 0.5% subset of the collection for estimating the shard ranking. Thus CORI's inferior performance is especially surprising. These observations motivate the experiment described next.

6.2 Effect of Query Length

Our hypothesis that we test in this section is that *for multi-term queries the unigram language model used by CORI severally misinforms the shard ranking estimation.* This is especially true for multi-term queries which consist of phrase(s). For example, in the query, *obama family tree*, it is critically important to treat the terms *family* and *tree* as a phrase and not as unigrams. The results in rows 4 through 7 in Table 1 provide evidence in support of the above hypothesis. Rows 4 and 5 are results for 52 queries, all of which are singleton queries. Across all the metrics the search effectiveness with CORI is higher in magnitude than that with ReDDE, which is exactly the opposite trend seen with multi-term queries. These results establish that CORI's subpar performance is restricted to multi-term queries. On the other hand ReDDE struggles more with singleton queries because estimation errors due to under-sampling are more likely when there is only one term in the query to inform the shard ranking.

6.3 Effect of Richer Query Representation

CORI's inferior performance with multi-term queries motivates the investigation in this section. The results with CORI and ReDDE when using this richer query representation are given in rows 8 and 9 of Table 1. These results show an opposite trend as that with unigram query representation (rows 2 and 3). The big-document approach (CORI_Uni+Bi) performances better, although not significantly, than the small-document approach (ReDDE_Uni+Bi). CORI clearly benefits more from the richer query representation than ReDDE. CORI_Uni+Bi results are significantly better than those with CORI. This is not the case for ReDDE_Uni+Bi. At early ranks, CORI_Uni+Bi is significantly better than even Exhaustive Search. This indicates substantial reduction in false-positives in the retrieved documents at early ranks. This is facilitated by two factors:- topic-based shards reduce the noise (false-positive matches) in each shard, and CORI_Uni+Bi selects the shards such that the resulting search space is *purer* than that used by Exhaustive Search.

Although, not shown in the table, the results for multi-term queries show similar trends as before:- for all the metrics the magnitude of search effectiveness with CORI_Uni+Bi is higher than with ReDDE_Uni+Bi, and CORI_Uni+Bi is significantly better than CORI. However, for ReDDE that is not the case. For singleton queries the results do not change because the richer query representation does not yield a different query. In summary, CORI_Uni+Bi provides the best search effectiveness until now. In the next section we investigate if we can improve the performance further.

6.4 Learning to Rank Shards

The last row in Table 1 reports the results with Learning to Rank Shards approach (LeToR-S). The first obvious trend in these results is that LeToR-S significantly outperforms the current best, CORI_Uni+Bi, at deeper ranks. At early ranks, however, the two approaches provide comparable precision and recall. To understand these results better we analyze a few queries in detail.

For one of the queries, *getting organized*, CORI_Uni+Bi ranks the shard with most number of relevant documents at 11th position, while LeToR-S ranks it at 3rd. This is a difficult query because both the query terms are common terms. Even when the terms are treated as a phrase, it is still not a focused query. This is reflected in the low scores assigned to relevant shards by CORI_Uni+Bi. LeToR-S, however, uses meta-data in addition to the document contents for defining the features. One particular meta-data feature, *url* field, proves to be especially valuable for this query that consists of common terms. Documents that contain *getting organized* in their *field* are relevant to the query. In turn, shards that contain such documents should be ranked higher too. In short, LeToR-S benefits from having the field score features, while CORI_Uni+Bi suffers because it only uses document contents for shard ranking. A few more example queries that highlight the value of the field score features are *battles in the civil war* and *kansas city mo*. For both queries, CORI_Uni+Bi ranks the most relevant shard at a much deeper rank than LeToR-S.

Another feature category that helps LeToR-S outperform CORI_Uni+Bi is the CORI minimum score features. Recall that the CORI minimum score feature is lowest CORI score computed for the individual query terms against a shard. This feature models the intuition that all of the query terms should have high CORI score for a relevant shard. Low CORI score, even if only for one of the query terms, indicates less likelihood of shard relevance. For query, *pacific northwest laboratory* only one shard contains all the relevant documents, LeToR-S ranks this shard at 8th place, while CORI_Uni+Bi ranks it at 11. Through the CORI minimum score feature, several false-positive shards are eliminated by LeToR-S from the shard ranking. These false-positive shards have high overall CORI score because some of the query terms have high CORI score, and thus dominate the cumulative score. However, the CORI minimum score captures that some query terms have low CORI score for these false-positive shards and thus push them down in the shard ranking.

The results in Table 1 also indicate that at early ranks LeToR-S performs significantly better than Exhaustive Search. This improvement often but not always comes from single term queries that may have one than one meaning or aspect associated with them (*euclid*, *avp*, *iron*, *unc*). The topic-based partitioning of the collection organizes the documents with similar meaning or aspect into the same shard. Often one of the meanings is more dominant than others in the collection, that is also often the relevant meaning for the query. Shards with the dominant meaning have higher document frequency (*df*) than shards with the rare meaning, and thus documents with dominant meaning only are searched. This reduces the false-positive documents (documents with rare meaning) from the result, and thus improves the search precision.

6.5 Effect of Distribution of Relevant Documents

When comparing the best performing Selective Search approach, LeToR-S, with Exhaustive Search we see in Table 1 that at early ranks, LeToR-S performs at par or better than Exhaustive Search in precision and recall both. However, at deeper ranks, LeToR-S struggles on recall more than precision, which is indicated by the significantly lower ndcg value. Our hypothesis for the reason behind this trend is that LeToR-S is unable to retrieve all the shards containing relevant docu-

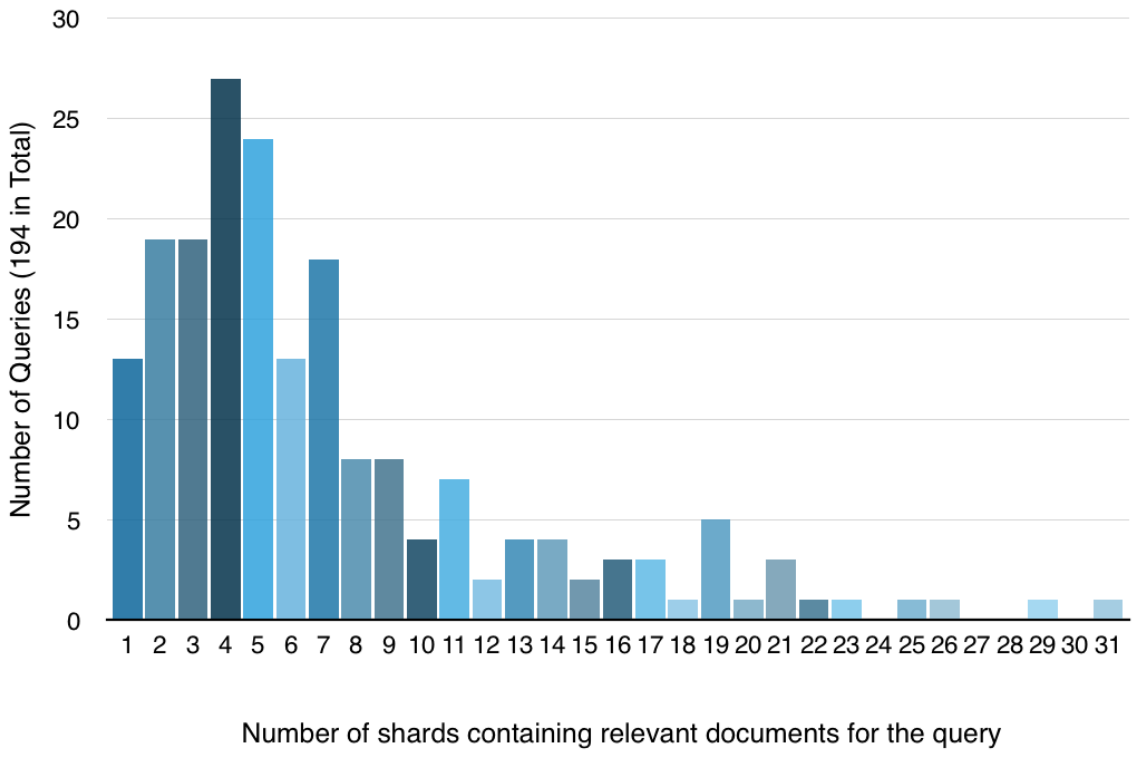

Figure 1: Histogram of number of shards containing relevant documents for the query.

ments. In order to test this hypothesis we conduct the following experiment.

The effectiveness of shard ranking algorithm is dependent on the distribution of the relevant documents across shards. If all the relevant documents are concentrated in a few shards then the shard ranking task is straightforward, however if the relevant documents are spread across many shards then task is much more challenging for any shard ranker. Figure 1 provides the histogram of the spread of relevant documents for the 194 queries. For a large fraction of the queries (27) the spread of the relevant documents are restricted to 4 shards. 70% of the total queries have a spread of 7 or less. For the remaining 30% of the queries the relevant documents can be spread across as many as 31 shards, indicating that the topic-based sharding approach failed to concentrate the relevant documents into few shards for these queries. We are however interested in the 70% of the queries for which the spread of relevant documents is restricted to 7 shards. We believe that LeToR-S should be able to provide effective shard ranking for these subset of queries, which in turn should help improve both, precision and recall. To test this intuition we separate the 194 queries into two groups based on the spread cutoff of ≤ 7. This gives us one group of 133 queries for which the relevant documents are spread across 7 or fewer shards, and the other group contains 61 queries.

The results for the two query groups with the various search approaches are given in Tables 2 and 3. For the first group of queries (Table 2) LeToR-S (and also CORI_Uni+Bi) provides statistically significant improvement over Exhaustive Search in precision at both, early and deep ranks. Furthermore these improvements in precision do not come at the cost of recall, the recall (at all ranks) with LeToR-S stays comparable to that with Exhaustive Search. Even ndcg with LeToR-S is not statistically different from that of Exhaustive Search. This is a rare phenomenon:- a search approach being able to balance precision and recall.

The results in Table 3 tell a different story. At deeper ranks, precision and recall, both suffer with all the Selective Search approaches. These results clearly establish the importance of concentrating the relevant documents into few shards. Doing so not only reduces the search cost but substantially

Search Approach	P@30	P@100	map	R@30	R@100	ndcg
Exh	0.218	0.153	0.166	0.198	0.399	0.405
CORI	0.229	0.154	0.165	0.178	0.361	0.369
ReDDE	0.225	0.154	0.169	0.202	0.397	0.391
CORI_Uni+Bi	0.241‡	0.163	0.179‡	0.205	0.401	0.400
ReDDE_Uni+Bi	0.233‡	0.156	0.171	0.209	0.400	0.385
LeToR-S	0.239 ‡	0.164‡	0.178‡	0.204	0.411	0.401

Table 2: Results on 133 Queries with number of relevant shards ≤ 7. ‡ indicates statistically significant improvement over Exhaustive Search. Underline indicates significantly worse values when compared to Exhaustive Search. Statistical significance testing was performed using paired T-test at $p<0.05$.

Search Approach	P@30	P@100	map	R@30	R@100	ndcg
Exh	0.332	0.265	0.215	0.124	0.320	0.482
CORI	0.312	0.228	0.162	0.114	0.258	0.372
ReDDE	0.322	0.243	0.179	0.118	0.279	0.418
CORI_Uni+Bi	0.336	0.243	0.187	0.125	0.282	0.424
ReDDE_Uni+Bi	0.331	0.246	0.185	0.122	0.283	0.424
LeToR-S	0.340	0.261 ¶	0.204 ¶	0.124	0.303 ¶	0.452 ¶

Table 3: Results on 61 Queries with number of relevant shard > 7. ¶ indicates statistically significant improvement when comparing LeToR-S with MTD_Uni+Bi. Underline indicates significantly worse values when compared to Exhaustive Search. Statistical significance testing was performed using paired T-test at $p<0.05$.

improves search precision without degrading the recall.

6.6 Effect of Number of Shards Searched

For all the experiments until now we have held the parameter, shard cutoff (T), constant at 10. That is, for all the Selective Search experiments the top 10 shards, out of 92 shards, were searched for each query. Changing this parameter directly affects the cost of Selective Search, and it also influences the search effectiveness. The influence of this parameter in the general distributed search seup has been extensively investigated by Markov and Crestani (Markov and Crestani, 2014). In this section we study the effect of parameter T on the two best performing Selective Search approaches, CORI_Uni+Bi and LeToR-S, and compare them to Exhaustive Search.

Table 4 provides the results for this analysis. At early ranks, LeToR-S performs on par with Exhaustive Search while searching just the top three shards. The corresponding search cost, approximated by A, is orders of magnitude lower for LeToR-S than for Exhaustive Search. When comparing LeToR-S with CORI_Uni+Bi, the former consistently outperforms the latter at all the shard cutoff values. Even the search cost for LeToR-S are marginally lower than those with CORI_Uni+Bi, indicating a bias toward smaller shards in case of LeToR-S.

As more shards are searched the recall at deeper ranks with LeToR-S becomes on par with Exhaustive. The analysis in the previous section demonstrated that LeToR-S becomes comparable to Exhaustive Search even on the ndcg metric if the spread of the relevant documents is restricted. The corresponding search cost of Selective Search approaches is at least an order of magnitude lower than that of Exhaustive Search.

7 Conclusion

Our goal for this work was to investigate ways to balance search precision and recall. Selective Search proved to be an effective search environment for this investigation, and focusing on the shard ranking problem to achieve this goal also proved to the correct choice. We revived an old shard selection approach, CORI, which supported competitive search performance when it was provided with richer query representation. We also introduced a novel shard ranking algorithm based on the well-established Learning-To-Ranking ap-

T	A	Search Approach	P@30	P@100	map	R@30	R@100	ndcg
92	50.22	Exh	0.254	0.189	0.181	0.174	0.374	0.429
1	0.561	CORI_Uni-Bi	$\underline{0.215}$	$\underline{0.124}$	$\underline{0.120}$	$\underline{0.133}$	$\underline{0.223}$	$\underline{0.247}$
	0.560	LeToR-S	0.230	$\underline{0.139}$¶	$\underline{0.124}$	$\underline{0.139}$¶	$\underline{0.246}$	$\underline{0.266}$
3	1.672	CORI_Uni-Bi	0.267	$\underline{0.174}$	$\underline{0.166}$	0.174	$\underline{0.327}$	$\underline{0.353}$
	1.666	LeToR-S	0.281‡	0.183	0.174	0.180	$\underline{0.342}$	$\underline{0.380}$¶
5	2.773	CORI_Uni-Bi	0.272	0.182	0.174	0.177	$\underline{0.342}$	$\underline{0.375}$
	2.768	LeToR-S	0.275‡	0.185	0.176	0.180	$\underline{0.351}$	$\underline{0.396}$¶
7	3.874	CORI_Uni-Bi	0.275‡	0.187	0.178	0.171	0.350	$\underline{0.393}$
	3.869	LeToR-S	0.274‡	0.190	0.181	0.177	0.365	$\underline{0.406}$
10	5.513	CORI_Uni+Bi	0.271‡	0.188	0.181	0.180	0.363	$\underline{0.408}$
	5.510	LeToR-S	0.270‡	0.194 ¶	0.186	0.179	0.377¶	$\underline{0.417}$¶

Table 4: Results for Exhaustive, and Selective Search with CORI_Uni+Bi and with LeToR-S at various shard cutoffs (T). A: The average search space size per query in million documents. ¶ indicates statistically significant improvement when comparing LeToR-S with CORI_Uni+Bi. ‡ indicates statistically significant improvement when comparing LeToR-S or CORI_Uni+Bi with Exhaustive. Underline indicates significantly worse values when compared to Exhaustive Search. Statistical significance testing was performed using paired T-test at $p<0.05$.

proach, which provided the best search precision while also sustaining the recall. A thorough analysis of the results showed that simply searching more shards does not necessarily increase search effectiveness. Instead the two factors that are critically important for Selective Search to successfully balance precision and recall are:- 1. partitioning the collection such that the relevant documents for the query are spread across less than 10% of the shards, and 2. to employ an effective shard ranking approach, like the ones proposed in this work.

References

Robin Aly, Djoerd Hiemstra, and Thomas Demeester. 2013. Taily: shard selection using the tail of score distributions. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 673–682.

Leo Breiman. 2001. Random forests. *Machine learning* 45(1):5–32.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11(23-581):81.

James P Callan, W Bruce Croft, and Stephen M Harding. 1992. The inquery retrieval system. In *Database and expert systems applications*. Springer, pages 78–83.

James P Callan, Zhihong Lu, and W Bruce Croft. 1995. Searching distributed collections with inference networks. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 21–28.

Jamie Callan. 2002. Distributed information retrieval. In *Advances in information retrieval*, Springer, pages 127–150.

Bruce Croft and Jamie Callan. 2000. Lemur project. https://www.lemurproject.org/.

Van Dang. 2013. Lemur project components: Ranklib. https://www.lemurproject.org/ranklib.php.

Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of machine learning research* 4(Nov):933–969.

Luis Gravano and Hector Garcia-Molina. 1999. Generalizing gloss to vector-space databases and broker hierarchies. Technical report, Stanford InfoLab.

Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. 1994. The effectiveness of gioss for the text database discovery problem. In *ACM SIGMOD Record*. ACM, volume 23, pages 126–137.

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

Evangelos Kanoulas, Keshi Dai, Virgil Pavlu, and Javed A Aslam. 2010. Score distribution models: assumptions, intuition, and robustness to score manipulation. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 242–249.

Anagha Kulkarni and Jamie Callan. 2015. Selective search: Efficient and effective search of large textual collections. *ACM Transactions on Information Systems (TOIS)* 33(4):17.

Anagha Kulkarni, Almer S Tigelaar, Djoerd Hiemstra, and Jamie Callan. 2012. Shard ranking and cutoff estimation for topically partitioned collections. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, pages 555–564.

Ilya Markov and Fabio Crestani. 2014. Theoretical, qualitative, and quantitative analyses of small-document approaches to resource selection. *ACM Transactions on Information Systems (TOIS)* 32(2):9.

Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13(4):346–374.

Milad Shokouhi. 2007. Central-rank-based collection selection in uncooperative distributed information retrieval. In *European Conference on Information Retrieval*. Springer, pages 160–172.

Milad Shokouhi, Luo Si, et al. 2011. Federated search. *Foundations and Trends® in Information Retrieval* 5(1):1–102.

Luo Si and Jamie Callan. 2003. 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*. ACM, pages 298–305.

Trevor Strohman, Donald Metzler, Howard Turtle, and W Bruce Croft. 2005. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*. Citeseer, volume 2, pages 2–6.

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

Jinxi Xu and W Bruce Croft. 1999. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 254–261.

Jun Xu and Hang Li. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 391–398.