BALANCING PRECISION AND RECALL WITH SELECTIVE SEARCH

A thesis presented to the faculty of
San Francisco State University
In partial fulfilment of
The Requirements for
The Degree

Master of Science
In
Computer Science

by

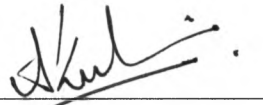
Mon-Shih Chuang

San Francisco, California

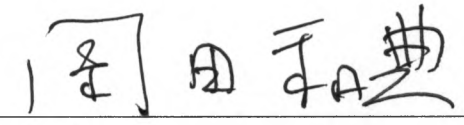
May 2018

CERTIFICATION OF APPROVAL

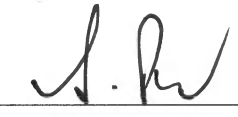
I certify that I have read *BALANCING PRECISION AND RECALL WITH SELECTIVE SEARCH* by Mon-Shih Chuang and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirements for the degree: Master of Science in Computer Science at San Francisco State University.



Anagha Kulkarni
Assistant Professor of Computer Science



Kaz Okada
Associate Professor of Computer Science



Arno Puder
Professor of Computer Science

BALANCING PRECISION AND RECALL WITH SELECTIVE SEARCH

Mon-Shih Chuang
San Francisco State University
2018

Balancing precision and recall is a long-standing problem in Information Retrieval field. We tackled this problem with *Selective Search*, which divides the large-scale document collection into small *shards* and passes the user query to only a few of those shards. In contrast, *Exhaustive Search* passes the user query through the whole collection. Selective Search has shown better efficiency and precision than Exhaustive Search because of the reduction of false positive documents but suffers from much worse recall due to the missing of relevant documents. The optimization work over Selective Search can be categorized into two tasks: (i) shard ranking and (ii) cutoff estimation. On the shard ranking task, we developed three new ranking approaches which improved both precisions and recalls over the previous works. On the cutoff estimation task, we tested three estimators to predict the number of shards ought to be searched, and we explore the finest parameter setting for precision-oriented and recall-oriented evaluation metrics.

I certify that the Abstract is a correct representation of the content of this thesis.

5/3/2018

Chair, Thesis Committee Anagha Kulkarni — Date

ACKNOWLEDGMENTS

I would like to thank Professor Anagha Kulkarni, my advisor, who supported me in all aspects of this work and the previous research work of TREC Total Recall Track. I would like to thank Professor Kaz Okada and Professor Arno Puder for being my committee member. Also, I want to thank the classmates in the same research group, Ben, Maria, Rajani, Meghana, Jose, Brooks, and Chanin, who helped me with the presentation rehearsal and exchange research ideas. I would like to thank to Professor Paul Ellison, and all the school tutors who helped me with the English grammar. Finally, I would like to thank my parents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In this work, we tackle the problem of *balancing search precision and recall* with *Selective Search*[17]. Balancing search precision and recall is an universal problem in the Information Retrieval (IR) field. Precision and recall are the two key attributes to evaluate the effectiveness of an IR system. Precision is the fraction of retrieved relevant documents among all retrieved documents. Recall is the fraction of retrieved relevant documents among all relevant documents in the document collection. Improving each of precision or recall will usually cause the other one to be less effective. In general web search services, the recall is usually given less priority than the precision. However, there are still many cases that require high recall while sustaining comparable precision such as patent search, academic paper search, law search, and medical search.

Selective Search[17] is a Distributed Information Retrieval (DIR)[6] approach that focuses on improving the search performance for large-scale collections. At the

preprocessing phase, Selective Search partitions the document collection into smaller chunks, *shards*, based on the document similarity. As a result, each partitioned shard contains documents that share the same certain topics, such as politics, music, technology, etc. At the query phase, instead of searching the whole collection, Selective Search directs the user query to a shard selector, *the broker*, and passes the query to only to a few shards selected by the broker. Selective Search has consistently shown its effectiveness on search precision[25, 27] because of the *purer search space*. Since the search space is formed by only the related shards selected by the broker, it pre-excludes large amount of non-relevant documents. As a result, the search space contains much fewer false-positive documents than the complete collection, which directly improves the precision. However, Selective Search usually struggles on search recall, which is due to the missing of relevant documents in search space. In order to improve recall, Selective Search needs to identify all the shards containing relevant documents. In our work, the shard containing at least one relevant document is considered a *relevant shard*. The goal of this work is to identify all relevant shards and distinguishing them from non-relevant ones. The goal is likely to be achieved with the shards clustered by their topic, where the relevant documents for a query are likely to be concentrated into a few shards (or one shard) because these documents are typically similar to each other [31].

In contrast of Selective Search, *Exhaustive Search* means simply passing the user query to the indexes of all divided shards, which has inferior efficiency but

does not suffer the problem of missing relevant documents in its search space. In some specific cases, if the result of Exhaustive Search returns a lot of false-positive documents before the relevant documents, Selective Search can also outperform Exhaustive Search on recall, but this has been limited to the recalls at early ranks. The problem of optimizing Selective Search on precision and recall is consisted of three main tasks: (i)Shard Representation, (ii)Shard Selection, and (iii)Result Merging.[6]. In our work, we developed ways to improve Selective Search through the shard selection task and setup experiments to evaluate those methods.

The purpose for an ideal *shard selection* is to select the shards contain the relevant documents for the query, while filtering out the shards that contain no relevant documents. Shard selection can be further divided into two sub-problems where the first one is to *rank* the shards based on their estimated relevance, and the second one is to determine a *cutoff* at certain rank to differentiate relevant shards and non-relevant shards. In our work, we compared big-document and small-document Selective Search algorithms and based on the existing algorithms we developed three shard ranking approaches that improved the selection process of topical partitioned shards. The approaches are (i) unigram+bigram indexing and query reformation, (ii) learning to rank shards, and (iii) pseudo relevance feedback with Wikipedia. We also investigated three existing cutoff estimators, each of which makes use of the distribution of the shard scores assigned by the learning to rank algorithms. For demonstrating our search algorithm, we developed the web search interface to

demonstrate our Selective Search approach.

This thesis is organized as follows: In Chapter 2, we discuss the related works in detail, which includes the formulas and parameter settings being used. In Chapter 4 we represent the information of the test environment of this work. In Chapter 3, we describe all of the techniques included in this work, purposes and formulations. In Chapter 5, we conduct nine analysis on the experiment results. In the end of the Chapter 5, we describe the web search interface of our Selective Search approach. In the last chapter, we conclude this thesis and discuss the future proposed tasks.

Chapter 2

Related Work

2.1 Selective Search

Selective Search algorithms are a part of distributed information retrieval (Distributed IR)[6]. Distributed IR can refer to search across multiple existing resources and aggregate the results (Federated search)[24] or multiple partitions from one collection (Cluster-based search) to improve efficiency and effectiveness.

The motivation behind federated searches are to reduce the limitation of page crawling methods. For example, the data resources might not be permitted to be accessed behind its own search interfaces or the bandwidth to the target server is very restricted[6]. Thus, instead of crawling from resources, federated search techniques introduce a central section called "the broker", which summarizes the characteristics of each resource. The summary of each resource, either statistics-based or sample-based, can be generated before serving the user queries. According

to the summaries stored, the broker passes the user queries only to few high relevant candidate resources to guarantee the search efficiency.

Cluster-based searches focuses on how to divide one collection into multiple partitions to improve the retrieval effectiveness. Xu and Croft[31] proposed a hypothesis that dividing the collection into multiple shards based on their related topics can improve both precision and recall. In addition to improving effectiveness, cluster-based searches also aim to reduce the query cost by passing the query parallel and selectively.

The search algorithms in Distributed IR can be roughly categorized into two groups: Big-document approaches, which are based on statistics of collections, will be discussed in subsection 2.1.1. Small-document approaches, which are based on a small subset of the collections, will be discussed in subsection 2.1.2.

2.1.1 Big-document approaches

Big-document approaches, also known as statistic-based approaches, treat each resource as an entity and use the statistics of each entity to compute the ranking. Big-document approaches can be utilized in cooperative environments, where the data providers are willing to share the whole documents collection, and allow the search algorithms to access the metadata and statistics.

GlOSS[13], and its extended versions gGlOSS [12] and vGlOSS[14] provide a solution of text database discovery problem. The GlOSS algorithms use the ratio

of term frequency, database size, and metadata of fields (title, body, links) to select the candidate resources.

CORI[5, 6] keeps document frequency (df) and shard frequency (sf) of each term and computes the score of every shard by a variation of tfidf formulas. normalized df times normalized inverted invert sf.

$$T = \frac{df_i}{df_i + 50 + 150 * sw_i / avg_s w} \tag{2.1}$$

$$I = \frac{log(\frac{S+0.5}{sf})}{log(S + 1.0)} \tag{2.2}$$

$$Score(t_k|S_i) = b + (1 - b) * T * I \tag{2.3}$$

Parameter	Description
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	the number of total words in the shard.
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

Table 2.1: CORI Parameters

To combine the term score as one query score, CORI inherited the operators from INQUERY[4] document retrieval system, where the operators are SUM, WSUM (weighted SUM), AND, OR, and NOT.

In 2013, Aly et al introduced Taily[1], which estimates the query score from

the score distribution of each single-term. According to Kanoulas et al's work [15], 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. In the experiment results, Taily showed a higher effectiveness than Rank-S when small amount of resources are searched.

2.1.2 Small-document approaches

In uncooperative environments, the search algorithms are not able to access the whole collection. Therefore, the key statistics including term frequency and total collection size are not able to be obtained. Big-document approaches are not capable to compute the shard scores. Small-document algorithms can solve this issue by approximating the document distribution inside a resource by sampling a small subset, which is called the Centralized Sample Database, or the *Centralized Sample Index (CSI)* structure.

To create CSI in the uncooperative environment, Callan and Si [28, 7] described a query-based sampling algorithm as follows:

The ReDDE[26] algorithm runs the user query against the CSI and assumes that the top n retrieved documents are relevant. The original version of ReDDE compute

Algorithm 1 ReDDE CSI sampling algorithm.

1: Select an initial query term.
2: Run a one-term query on the database.
3: Retrieve the top N documents returned by the database.
4: Update the resource description based on the characteristics of the retrieved documents.
   (a) Extract words and frequencies from the top N documents returned by the database; and
   (b) Add the words and their frequencies to the learned resource description.
5: If a stopping criterion has not yet been reached,
   (a) Select a new query term; and
   (b) Go to Step 2.

a score for each shard as follow equation:

$$Score(S_i^q) = Count(n, S_i^q) \times \frac{|S_i|}{|S_i^{CSI}|} \tag{2.4}$$

$Count(n, S_i^q)$ 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.

UUM[25] and RUM[27] learn a utility function to estimate the probabilities of relevance for each database from training data. For each training query, it first uses CORI to select the databases to be searched, then acquires relevance judgments for

the top 50 returned documents from human. A logistic model is built to transform the normalized centralized document scores in CSI to probabilities of relevance, which is given by human.

$$R(d) = P(rel|d) = \frac{exp(a_{CSI} + b_{CSI}CSI\_Score(D))}{1 + exp(a_{CSI} + b_{CSI}CSI\_Score(D)} \tag{2.5}$$

Where a_{CSI} and b_{CSI} are the two parameters to be learned from the training queries. The score of documents not in CSI also estimated by using the two sampled documents from that database with highest centralized scores.

$$\hat{CSI\_Score}(D_{i1}) = \frac{exp(\alpha_{i0} + \alpha_{i1}CSI_Score(D_{si1}) + \alpha_{i2}CSI_Score(D_{si2}))}{1 + exp(\alpha_{i0} + \alpha_{i1}CSI_Score(D_{si1}) + \alpha_{i2}CSI_Score(D_{si2})))} \tag{2.6}$$

The training goal of UUM and RUM can be either for high-recall or high-precision. The trained model can rank databases and decide how many documents for each database to be retrieve. UUM and RUM have significantly improved the precisions at 5,10,15,20,30 compare to ReDDE and CORI.

CRCS[23] passes the user queries to CSI and compute the score of each resource from the returned document rank. There are two version of CRCS approaches introduced, different from the decreasing rate of the score respect to the rank. CRCS(1) uses a simple linear decrease model:

$$R(D_j) = \begin{cases} \gamma - j & \text{if j} < \gamma \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

where R(Dj) represents the impact of document D at the jth rank of results search against CSI. CRCS(e), the exponential version, uses a exponential decay model:

$$R(D_j) = \alpha exp(-\beta \times j) \text{ if } D_j \in S_i \tag{2.8}$$

Where j is the rank of document D. Coefficient parameters α and β are two constants respectively set to 1.2 and 2.8.

From the impact $R(D_j)$, the normalized score of each shard is computed by

$$Score(S_i) = \frac{|S_i|}{|S_{max}| \times |S_i^{CSI}|} \times \sum_{D \in S_i^{CSI}} R(D_j) \tag{2.9}$$

Where $|S_i|$ is the size of shard i, $|S_{max}|$ is the size of the largest shard, and $|S_i^{CSI}|$ is the size of documents belongs to shard i in CSI.

SUSHI[30] 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. The curve model is then used to estimate the whole distribution in the shard. After computed interpolated scores from the curve, the shards are selected to optimize specific metrics. By default, SUSHI optimizes precision at 10.

Contrast from previous methods such as CORI, ReDDE, and CRCS, the number of shards selected by SUSHI is dynamic because SUSHI select shards to optimize precision at 10. In other words, SUSHI only select the shards contain documents ranked in top 10 after interpolating. If one shard is estimated to have all top 10 relevant documents, then SUSHI select only that shard. This technique is called "dynamic cutoff estimation".

SHiRE[19] is another Selective Search algorithm which applies dynamic cutoff estimation. The cutoff estimation of SHiRE is based on the hierarchical document structure, and there are three different construction rule of the hierarchical structure in total. Lex-S construct the hierarchical structure based on the lexical similarities (distances) between the vectorized documents. The documents with high similarity will be attached under the same internal node. In Rank-S, the CSI is converted to left-branched binary tree which the top ranked document is the left-most deepest node. The construction of the hierarchy of Rank-S approach is the most efficient among the 3 algorithms. The score of every shard is computed by summing up its document score in CSI times an exponential decay rate through the rank.

$$Score(S_i^q) = \sum_{d \in S_i} Score(d) * B^{-CSI\_rank(q,d)} \tag{2.10}$$

Where B is the base parameter to control the speed of decay. The dynamic cutoff estimation of by left out the shards with $Score(S_i) < th$. Where th is the converge threshold. In our test we set it to 0.0001. Conn-S uses shard membership to con-

struct the hierarchy. The top ranked document is the left-most deepest node, then for each following document sorted by rank, the algorithm attaches the document under the same internal node if it belongs to the same shard of previous document, or creates a new internal node one level higher and attaches the document under the node. To taking advantage from the cluster hypothesis, the SHiRE algorithm was tested on the topical shard partition.

SHRKC[16] learns the cutoff number of how many shard should be searched based on random forest[2] regression, using three categories of characteristics from CSI, individual shards, and the whole collection as the training inputs.

2.1.3 Ideal Resource selection method using oracle

Relevance-based ranking (RBR) is the most common estimated upper-bound for evaluating collection selection methods. In RBR, collections are ranked according to the number of relevant documents that they contain for queries.

2.2 Learning to Rank

The methods that use machine learning technologies to solve the problem of ranking learning-to-rank methods. Tie-Yan Liu[20] analyzed existing learning-to-rank algorithms and categorized them into three groups by their input representations and output targets: the pointwise, pairwise, and listwise approach.

2.2.1 Pointwise Approach

The input representation of the pointwise approach contains a feature vector of each document. The output target contains the relevance degree of each document. The relevance judgments can be converted to ground truth labels in terms of relevance degree.

2.2.2 Pairwise Approach

The input representation of the pairwise approach contains pairs of documents, both represented by feature vectors. The output target contains the pairwise preference (which takes values from $+1, -1$) between each pair of documents. The relevance judgments can be converted to ground truth labels in terms of pairwise preferences.

2.2.3 Listwise Approach

The input space of the listwise approach contains a set of documents represented by feature vectors, associated with query q. The output targets of the listwise approach contain the ranked list (or permutation) of the documents. The relevance judgments can be converted to ground truth labels in terms of a ranked list.

2.3 Summary

In this chapter, we discussed the related works of improving the precision and the recall of document rankings. (i) Selective Search and (ii) Learning-to-Rank. In the following chapters, we will develop shard ranking approaches which extend from these related works. We experimented the Selective Search approaches, CORI and ReDDE, for the test collection. The results in Section 5.1 and Section 5.2 indicate that there is a large room between the Selective Search approaches and the upper-bounds. This experiment result motivated us to develop Selective Search approaches to achieve closer precisions and recalls to the ideal upper-bounds.

Chapter 3

Shard Ranking Approaches

3.1 CORI_Uni+Bi and ReDDE_Uni+Bi

In this work, we focus on developing shard ranking approaches over the cooperative environment. In the cooperative environment, the term statistics of target collection which needed by the candidate big-document approach, CORI, are fully accessible. On the other hand, the candidate small-document approach, ReDDE, still estimates the shard relevancy from 0.5 percent of randomly sampled documents. Because only CORI has the complete information of the whole collection, one might predict that CORI will have better shard ranking estimation than ReDDE. However, after comparing the results of CORI and ReDDE, CORI showed reduced effectiveness as compare with ReDDE both on precision and recall. To explain this, we analyzed the false-positive shards that CORI retrieved. While using only unigram term statistics, we found that CORI's effectiveness is limited due to lack of the context in the

whole query. For example, when the query *arizona fish and game* decomposed into unigrams, the context, *fishing and gaming in Arizona* was lost. In this example, CORI was misled to retrieve shards with other topics related to Arizona (e.g. shards of politics or local news). These observations motivate us to develop an approach that can extract phrasal information from the user query. Thus, we implement the Unigram+Bigram (Uni+Bi) versions for CORI and ReDDE. We then reformulated the user queries into Uni+Bi representation as follows. Given a query with n terms, $n-1$ bigrams are composed of all consecutive term pairs, then the bigrams with stopwords are discarded. For example, if "arizona game and fish" is the query text, "arizona game" is the only phrase parsed since "and" is a stop word. The stop word list we are using is from the Introduction to Information Retrieval book[22]. For the single-term queries or the queries contain no phrases because of stop words (e.g. "to be or not to be"), we remain using only unigram information on them. For CORI_Uni+Bi, the bigram term frequencies must be pre-computed for each shard in addition to the unigrams. By storing the bigram term frequencies, the relevancy score of each shard can be computed by Uni+Bi terms statistics efficiently at the online phase. For ReDDE_Uni+Bi, the query against CSI needs to be reformulated to Uni+Bi version. We use the Indri query operator *#combine* to integrate unigram and bigram terms, and the *#uwX* operator is used for specifying an unordered phrase of length X. As an example, the query *obama family tree*, in the form of Indri Query Language will be: *#combine(obama family tree #uw2(obama family)*

#uw2(family tree)). After ReDDE_Uni+Bi runs the Uni+Bi query against the CSI, the rest steps of the search process are the same as the unigram version of ReDDE.

3.2 LeToR-S (Learning to Rank Shards)

The second approach of this work introduces Learning-to-Rank (LeToR) framework for the shard ranking task. The LeToR framework has shown the powerfulness on improving the document ranking task [3, 11, 32]. Recently, there were few works also applied LeToR to the shard ranking problem [9], but there are still many different settings to explore. In our LeToR approach, we compute CORI scores from query and shard pair $< Q, S_i >$ as the input features, and the number of relevant documents respects to $< Q, S_i >$ as our training target. The feature set is consisted of 30 CORI scores from five fields, three operators, and two language models. The scores are computed from the term statistics of five fields including *title, body, heading, url*, and *whole document*. For each field, three variants of CORI scores, *CORI_SUM*, *CORI_MIN*, and *CORI_VAR* are computed. The scores from unigram and phrasal language model query representations are computed separately. For single-term queries, the phrasal scores are defined as zero.

$$CORI\_SUM(Q|S_i) = \sum_{t_k \in Q} Score(t_k|S_i) \tag{3.1}$$

$$CORI\_MIN(Q|S_i) = \min_{t_k \in Q} Score(t_k|S_i) \tag{3.2}$$

$$\mu = \frac{1}{n} \sum_{t_k \in Q} Score(t_k|S_i) \tag{3.3}$$

$$CORI\_VAR(Q|S_i) = \sum_{t_k \in Q} (Score(t_k|S_i) - \mu)^2 \tag{3.4}$$

where $Score(t_k|S_i)$ is the score of one n-gram term t_k computed using the standard CORI formulation (Section 2.1.1). The ranking model is learned using the Random Forest[2] algorithm implemented in the RankLib[10] library from the Lemur Project[8]. As a listwise LeToR approach, the ranker optimizes Normalized Discounted Cumulative Gain (NDCG) at top ten ranked shards.

3.3 LeToR-SWP (LeToR-S with Wikipedia-based Pseudo Relevance Feedback)

The third approach in this work is motivated by an observation that user queries tend to have fewer than three terms, and many queries contain only one term. Thus, we developed a query expansion approach above LeToR-S. To expanse the short length queries, we choose a well-applied solution used for document ranking, *pseudo relevance feedback* (PRF). Furthermore, to avoid the problems of losing relevancy to user queries, we need to select the expansion terms which sustain the relationship of the original topic.

Figure 3.1 pictured the working flow of pseudo relevance feedback using the top-ranked shard and Wikipedia. As a pre-processing step, a *profile* for every shard

Method	N-Gram	Field
CORI SUM	Unigram	Whole Document
CORI SUM	Bigram	Whole Document
CORI SUM	Unigram	Title
CORI SUM	Unigram	Body
CORI SUM	Unigram	Heading
CORI SUM	Unigram	Url
CORI SUM	Bigram	Title
CORI SUM	Bigram	Body
CORI SUM	Bigram	Heading
CORI SUM	Bigram	Url
CORI MIN	Unigram	Whole Document
CORI MIN	Bigram	Whole Document
CORI MIN	Unigram	Title
CORI MIN	Unigram	Body
CORI MIN	Unigram	Heading
CORI MIN	Unigram	Url
CORI MIN	Bigram	Title
CORI MIN	Bigram	Body
CORI MIN	Bigram	Heading
CORI MIN	Bigram	Url
CORI VAR	Unigram	Whole Document
CORI VAR	Bigram	Whole Document
CORI VAR	Unigram	Title
CORI VAR	Unigram	Body
CORI VAR	Unigram	Heading
CORI VAR	Unigram	Url
CORI VAR	Bigram	Title
CORI VAR	Bigram	Body
CORI VAR	Bigram	Heading
CORI VAR	Bigram	Url

Table 3.1: The 30 features used in the LeToR-S approach of this work.

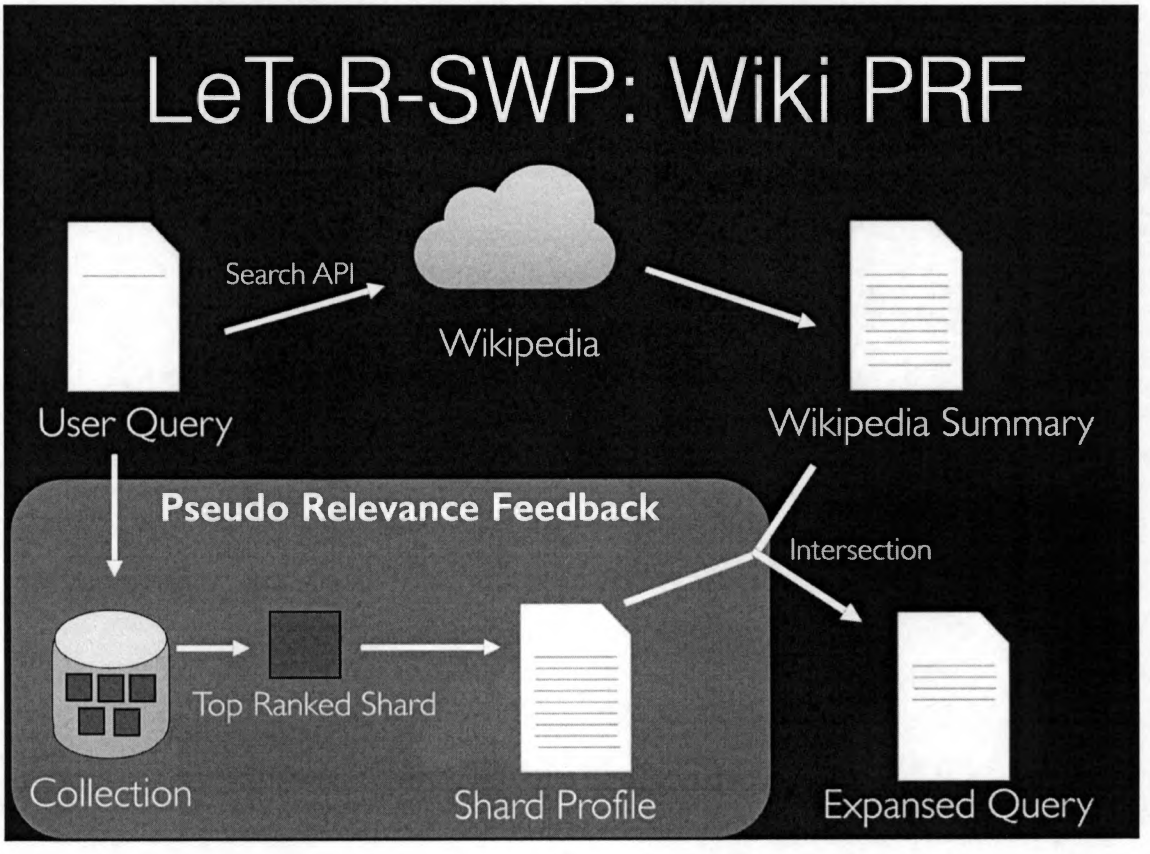


Figure 3.1: The process of LeToR-SWP.

is recorded, which consists of the top 1000 terms with highest document frequency (DF) in the shard. As an example, the top 5 terms in the profile of one of the shards are *rum piano classical acoustic composer*, which are all related to music, while for another shard the top profile terms are *pepper butter onion chop fry*, which are all related to cooking. At query time, the LeToR-S approach is employed to obtain a shard ranking for the original query. The *profile* of the top-1st ranked shard is

considered pseudo relevant, and is used to extract expansion terms. Out of the shard profile, the top 100 terms with highest $DF.ISF$ scores are selected to form a set of expansion candidate terms, where $DF.ISF$ is defined as:

$$DF.ISF(t, S_i) = \log(1 + DF(t, S_i)) \times \log(ISF(t)) \tag{3.5}$$

Where $DF(t, S_i)$ is the document frequency of the term t in the shard S_i, and ISF is:

$$ISF(t) = S/ \sum_{i=1toM} I(t, S_i) \tag{3.6}$$

Where S is the total number of shards, and I is an indicator function that returns 1 if the term t occurs in the *profile* of shard S_i, and 0 otherwise.

The top candidate terms are then filtered through Wikipedia by following steps. First, the original query is run against the English Wikipedia to retrieve the summary field of the related Wikipedia entry. If the Wikipedia search API returns a disambiguation page, it is considered no summary is retrieved. Then, only the terms that also occur in the retrieved Wikipedia summary are added to the query to generate the expanded version. Thus, the final expanded query is the intersection of expansion candidate terms and the Wikipedia summary. The expanded query is then passed to the ranking model of LeToR-S approach to get the shard ranking. After the exploration of the query needs further expansion, we found almost all single-term queries were benefited from PRF, while not all the multi-term

queries were. Therefore, we only apply this approach for single-term queries, where the information is the most insufficient to retrieve relevant shards. We called this approach as LeToR-SWP, which takes the leading characters from (LeToR-S with Wikipedia-based Pseudo Relevance Feedback).

3.4 Dynamic Shard Rank Cutoff Estimation

As mentioned in the Chapter 1, the shard selection problem is consisted of two sub-problems: shard ranking and cutoff estimation. When the cutoff rank is lower than the ideal rank, the search cost will increase and the effectiveness will reduce. Especially, the search precision will be seriously reduced due to the noise. On the other hand, if the estimated cutoff rank is above the ideal rank, i.e fewer than the number of relevant shards, the recall will be reduced since missing relevant documents in the search space. Therefore, estimating a certain cutoff rank to differentiate the relevant and non-relevant shards for the query is an important part for optimizing shard selection [1, 16, 19]. This estimation must be query-specific since the optimal cutoff of each query is based on how its related topics distribute across the shards. We evaluate the precision and recall-oriented metrics for three different shard rank cutoff estimators and compare them with the result of setting the same fixed cutoff for all queries. Showed by Fig- 5.4, from the curve of sorted LeToR shard scores, we observed the *elbow* pattern showed for most of the test queries. Moreover, the rank that the elbow located is often the optimal cutoff to differentiate the relevant and

non-relevant shards. This observation motivated us to choose the methods which aim to locate the elbow in the score curve as the dynamic cutoff estimators. In this work, we tested two estimators introduced in Kulkarni's work[18], PK2 and PK3, which estimate the elbow location by two or three consecutive scores, separately. The PK2 approach compares two consecutive shard scores to locate the elbow. PK2 computes the ratio of shard scores of the current rank and the previous rank where

$$PK2(r) = \frac{Score(r)}{Score(r-1)} \tag{3.7}$$

The second approach for cutoff estimation, PK3, compares three consecutive scores to locate the elbow. PK3 is defined as:

$$PK3(r) = \frac{2 \times Score(r)}{Score(r-1) + Score(r+1)} \tag{3.8}$$

After the PK value is computed through all shard ranks, the mean and standard deviation of the PK values are then computed. From the lowest rank to the highest rank, the algorithm tracks the PK values, and when it hits the first rank with the PK value greater than the mean plus one standard deviation, that rank is determined as the cutoff rank. This procedure will eliminate the shards with LeToR scores on the right side of the elbow location. In short, only those shards ranked before the elbow location are selected and searched. For both PK2 and PK3, the number of data points (M) over which used to compute the mean and the standard deviation

is a tunable parameter. In a curve with multiple elbow patterns, M can decide the scale of elbow we want to detect. With smaller M, the mean PK value will become higher, and the cutoff will be set on the location where the score dropping is large enough.

For the last cutoff estimation approach, we investigate a variant of the Rank-S algorithm[19], where the distribution of the shard scores is used to estimate the cutoff rank. An exponential decay function is applied to the shard scores as follows:

$$Rank - S(r) = Score(r) \times B^{-r} \tag{3.9}$$

Where r is the rank, and B is a tunable parameter to control the decaying rate. The rank at which Rank-S score decays below a certain threshold, set to 0.0001, is the estimated cutoff.

3.5 Summary

In this chapter, we compare the big-document approach and small-document approach. Based on the comparison and false-positive analysis on CORI, we introduced 3 different shard ranking approaches using (i) bigram statistics, (ii) learning-to-rank framework, and (iii) pseudo relevance feed back with Wikipedia. In the end of this chapter, we introduced 3 different shard cutoff estimators which predicts the ideal number of shards to be selected. In the following chapters, we will conduct the

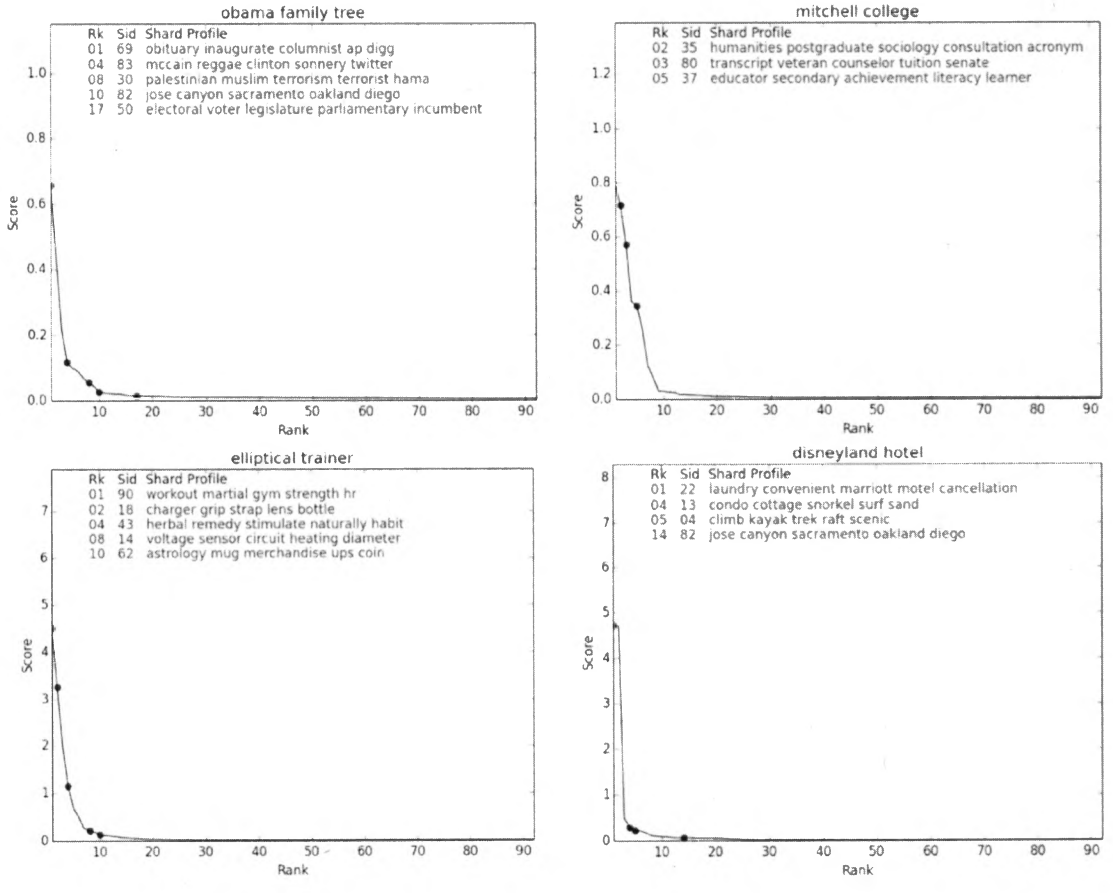


Figure 3.2: The shard score distribution of LeToR-S.

experiments on test collection and present the results.

Chapter 4

Experimental Setup

4.1 Testbed

To test our Selective Search algorithms, we use CategoryB of Clueweb09 as the test dataset. CategoryB contains the first 50,220,423 documents of Clueweb09, a larger set contains more than 300 million documents. The CategoryB dataset is divided into 92 topical shards. The user queries we used are query 1-200 from TREC Web Track 2009-2012. However, only 194 queries from query 1-200 were used because for query id = 20,95,100,112,143,152 there is no relevant document in CategoryB dataset. The average of number of relevant document is 56.9 and the query length distribution are listed in Table 4.1.

In the small-document approaches, we construct the CSI by randomly sampling 0.5 percent (1/200) of the size of documents from every shard. The search engine used in our experiment is Indri 5.9 form Lemur toolkit[29]. The indexes are built

Length	Num of Queries
1	52
2	46
3	66
4	19
>= 5	11
Total	194

Table 4.1: The distribution of the query lengths among 194 test queries.

using Krovetz stemmer and with no stopwords elimination.

4.2 Evaluation Metrics

The evaluation metrics used are precision at top 30 documents(P@30), precision at top 100 documents(P@100), mean average precision at top 1000 (MAP), recall at top 30 documents(R@30), recall at top 100 documents(R@100) and Normalized Discounted Cumulative Gain at top 1000 (NDCG). All the values in the experiment results are average values of the queries in the test query set. The purpose of choosing the six metrics is to evaluate the improvement in both precision and recall. To prove the algorithms does significant improvement or decline, we use paired t-test to compare the results of different experimental settings. If those results have confidence-level over 95% to be different, we assume the improvement/decline is significant between the 2 runs.

4.2.1 Precision and Recall

Precision at a certain rank k (P@k) is the rate of relevant documents retrieved in the top k documents. Recall at a certain rank k (R@k) is the rate of relevant documents in the top k documents among all relevant documents.

$$P@k = \frac{1}{k} \sum_{i=1}^{k} \begin{cases} 1 & \text{if } D_i \in D^{rel} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

$$R@k = \frac{1}{|D^{rel}|} \sum_{i=1}^{k} \begin{cases} 1 & \text{if } D_i \in D^{rel} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Where D_i is the document ranked at the ith rank, D^{rel} is the relevant document set, and $|D^{rel}|$ is the number of documents in the set.

4.2.2 Average Precision and Mean Average Precision

Average precision (AP) is the average of precisions when each relevant document retrieved. To compute AP at a given rank k, the precisions of relevant documents not retrieved at kth rank is set to 0.

$$AP@k = \frac{1}{|D^{rel}|} \sum_{d \in D^{rel}}^{|D^{rel}|} \begin{cases} P@rank(d) & \text{if } rank(d) <= \text{k} \\ 0 & \text{if } rank(d) > \text{k} \end{cases} \tag{4.3}$$

For example, if there are totally 5 relevant documents for the query, and the system retrieved only 3 of them at the 1st, the 4th, and the 5th rank, $AP@5 = \frac{1}{5} \times (1 + \frac{2}{4} + \frac{3}{5} + 0 + 0) = 0.42$.

Mean Average Precision (MAP) is the arithmetic mean of the AP for all queries in the query set. For example, if there is a query set contains 2 queries, $q1$ and $q2$, and AP@1000 for $q1$ is 0.5, AP@1000 for $q2$ is 0.8, MAP@1000 is $\frac{1}{2}(0.5+0.8) = 0.65$.

4.2.3 Normalized Discounted Cumulative Gain

Normalized Discounted Cumulative Gain (NDCG) is the evaluation metric using the non-binary relevance levels and relevance gains. In the TREC Web Track, the non-binary relevance levels are set from 2 (highly relevant) to -2 (highly non-relevant). Relevance gain is the pre-defined score for each relevance level, in the default trec_eval 9.0 setting we are using, the relevance gain is equal to the relevance level. Cumulative gain (CG) at rank k is the sum of relevance gain for documents retrieved before rank k.

$$CG@k = \sum_{i=1}^{k} rel\_gain(D_i) \tag{4.4}$$

In the formula of discounted cumulative gain (DCG), the relevance gains reduced logarithmically with the rank. The purpose of DCG is to penalize the search result that highly relevant documents at lower ranks than weakly relevant documents or

non-relevant documents.

$$DCG@k = \sum_{i=1}^{k} \frac{rel\_gain(D_i)}{\log_2(i+1)} \tag{4.5}$$

Ideal DCG (IDCG) is the DCG when all relevant documents are retrieved at the top and sorted by the relevance level in the descending order. Normalized DCG (NDCG) represents the DCG normalized by the ideal value, which is computed as follows.

$$NDCG@k = \frac{DCG@k}{IDCG@k} \tag{4.6}$$

For example, if there are exactly 2 relevant documents D_A and D_B for query q, D_A has relevance level 2, and D_B has relevant level 1. IDCG is DCG when D_A is ranked at the 1st, and D_B is ranked at the 2nd. $IDCG@2 = \frac{2}{\log_2 2} + \frac{1}{\log_2 3} = 2.631$. If the system ranked D_B at the 1st, and ranked D_A at the 2nd, then $DCG@2 = \frac{1}{\log_2 2} + \frac{2}{\log_2 3} = 2.262$ and $NDCG@2 = \frac{2.262}{2.631} = 0.860$.

If the system ranked all relevant documents ideally (sorted them by the relevance level in descending order), DCG is equal to IDCG and NDCG has the maximum value 1.0.

4.3 Search Cost Estimation for Small-Document Algorithms

In Macdonald et al's work[21], it evaluated the correlations of multiple query efficiency predictors and the real query response time. It demonstrated that the query response time is strongly proportional to the number of documents scored, which is the number of documents contains at least one query term. Therefore, to estimate the search cost, we compute the number of documents contains at least one query term for the index I as follows.

$$|D_{scored}(q, I)| = |\bigcup_{t \in q} D_{scored}(t, I)| \tag{4.7}$$

Where $|D_{scored}(t, I)|$ is the size of a set of documents that contain term t in index I. $|D_{scored}(q, I)|$ is the size of the union set of $|D_{scored}(t, I)|$ for each term t in query q.

For small-document approaches, the query is passed to CSI to compute the shard ranking and then passed to the selected shards. Thus, the total cost C_{total} is the query cost against CSI plus the query costs against all indexes of selected shards. The search latency $C_{latency}$ is the cost of CSI plus the maximum of the costs among selected shards, which represents the real system latency when running the query parallel across the selected shards.

$$C_t = C_{total} = \sum_{s \in S} |D_{scored}(q, s)| + |D_{scored}(q, CSI)| \tag{4.8}$$

$$C_l = C_{latency} = \max_{s \in S} |D_{scored}(q, s)| + |D_{scored}(q, CSI)| \tag{4.9}$$

Where S is the set of the indexes of selected shards.

4.4 Search Cost Estimation for Big-Document Algorithms

The big-document algorithms have no CSI structure, the statistics of terms can be stored in the hash table and the access time is O(1). Thus, C_{total} is the sum of the query costs against all the indexes of selected shards and $C_{latency}$ is the maximum query cost among the selected shards.

$$C_t = C_{total} = \sum_{s \in S} |D_{scored}(q, s)| \tag{4.10}$$

$$C_l = C_{latency} = \max_{s \in S} |D_{scored}(q, s)| \tag{4.11}$$

Chapter 5

Results and Analysis

In each of the following nine section, we conduct the experiments and analyze the possible reason of improvement/decline based on observations from the result. In Section 5.1, we will examine the Exhaustive Search and two kinds of upper-bound shard ranking methods. In Section 5.2, we are going to compare the results of state-of-the-art approaches CORI and ReDDE. In Section 5.3, we will analyze how CORI and ReDDE perform differently with single-term queries and multiple term queries. In Section 5.4, we will introduce the bigram statistics to previous methods and evaluate the results. In Section 5.5, we will evaluate how learning-to-rank framework improved the shard ranking task. In Section 5.6, we will show the most promising shard ranking result based on the fusion of LeToR and PRF. In Section 5.7, we discuss the difficulty of shard selection task of our testbed by analyzing how the relevant documents divided across the shards. In Section 5.8, we will analyze the trend of search effectiveness and efficiency respect to the number of shards

to be searched. In Section 5.9, we compare the three dynamic cutoff estimator with precision and recall-oriented parameter settings and lead to the dynamic cutoff estimation. In Section 5.10, we will demonstrate a web interface for our approaches. In Section 5.11, we will summarize the result and analysis chapter.

5.1 Upper-bound effectiveness of ideal shard selection

To evaluate how much room we can improve both precision and recall over the Exhaustive Search through Selective Search, we propose two different shard rankings based on the ground truth. The precision and recall of the two shard rankings indicate the upper-bound an optimized rank selection algorithm can achieve. The first ground truth-based algorithm is Relevance-based ranking (RBR), which is mentioned in the Subsection 2.1.3. RBR is one commonly applied way to estimate upper-bound of shard selection. For each query, RBR ranks the shards according to how many relevant documents in the shard. If the distribution of relevant documents is skewed, RBR can achieve very high effectiveness both on precision and recall. However, RBR is not guaranteed to give the most optimized document ranking. For our experimental topical shards, we found the distribution of the shard size, i.e number of total documents in each shard is also skewed. In the experiment, one certain shard was kept being selected by RBR for different queries. We found that selected shard is the largest one, and its size is 10 times larger than the average. From this observation, we noticed that since RBR compares the total num-

ber of relevant documents for a shard, it will bias shards with larger sizes. Since those large shards can contain higher ratios of non-relevant documents than others, the precision of RBR still has rooms to be improved. Thus, we tested another upper-bound ranking method according to the shard "purity", which is the ratio of relevant documents of shards. Table 5.1 compares the two upper-bound estimations to Exhaustive Search. The parameter T is the static cutoff number of how many shards is selected to pass the user query. According to the results in Table 5.1, for both of the upper-bound methods, the improvements from Exhaustive Search are very significant. When T>= 3, all six metrics of the upper-bound methods results are statically significant better than Exhaustive Search, where the p values of pair t-test are less than 0.05. Further, by comparing the two upper-bound estimations we found that when the cutoff is small (e.g T <=1,3), purity-based ranking gets better precision and recall than RBR. Although the improvements are not significant by applying paired t-test, the purity-based ranking can still be seen as a comparable alternative of RBR. When the cutoff number is large enough to include all of the relevant shards for each query, RBR and Purity will have the same result on shard selection and document ranking.

Run	T	P@30	P@100	MAP	R@30	R@100	NDCG	C_t	C_l
Exhaustive	92	0.254	0.189	0.181	0.174	0.374	0.429	12.67	0.29
Purity	1	0.329†	0.186	0.205†	0.244†	0.391†	0.398	0.22	0.22
Purity	3	0.330†	0.217†	0.240†	0.248†	0.464†	0.473†	0.54	0.25
Purity	5	0.325†	0.220†	0.244†	0.246†	0.472†	0.486†	0.75	0.25
Purity	7	0.324†	0.221†	0.247†	0.246†	0.474†	0.491†	0.88	0.25
Purity	10	0.320†	0.221†	0.246†	0.244†	0.473†	0.492†	0.98	0.26
Purity	15	0.321†	0.222†	0.248†	0.245†	0.475†	0.495†	1.08	0.26
Purity	20	0.320†	0.222†	0.249†	0.245†	0.476†	0.496†	1.16	0.26
RBR	1	0.328†	0.185	0.204†	0.241†	0.388†	0.396	0.22	0.22
RBR	3	0.330†	0.218†	0.240†	0.246†	0.465†	0.473†	0.54	0.25
RBR	5	0.324†	0.220†	0.245†	0.246†	0.471†	0.487†	0.76	0.25
RBR	7	0.323†	0.221†	0.246†	0.245†	0.473†	0.491†	0.88	0.26
RBR	10	0.320†	0.221†	0.247†	0.244†	0.474†	0.492†	0.98	0.26
RBR	15	0.321†	0.222†	0.248†	0.245†	0.476†	0.495†	1.09	0.26
RBR	20	0.320†	0.222†	0.249†	0.245†	0.476†	0.496†	1.16	0.26

Table 5.1: RBR and Purity-based with fixed cutoff (T) = 1,3,5,7,10,15,20. †:Significant improvements from Exhaustive Search. Statistical significance testing was performed using paired t-test at $p<0.05$.

5.2 Comparison of Big-document and Small-document algorithms on Topical Shards

To understand which of big-document and small-document approaches performs better the topical clustered shards, we choose CORI and ReDDE algorithms that described in Chapter 2 to represent the two group of approaches.

CORI with unweighted-sum operator is chosen as the big-document approach, we choose . The score of particular user query on each shard are summed up from

the scores of each query term t_k (see equation. (2.3)).

$$Score(S_i^q) = \sum_{t_k \in q} Score(t_k|S_i) \tag{5.1}$$

A variation version of ReDDE is chosen as the small-document approach to fit in the cooperative environment. The size of each shard is accessible information and the CSI is constructed by sampling certain percentage (0.5%) of documents of each shard. For a specific query, The document scores form searching against CSI are used as relevancy, then the score of every shard is computed by summing up the score of its document in top n rank of CSI:

$$Score(S_i^q) = \sum_{d \in S_i} I(q,d) Score(d) \tag{5.2}$$

I(q,d) is the indicator function that:

$$I(q,d) = \begin{cases} 1 & \text{if } CSI\_rank(q,d) \leq n \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

In the original works of ReDDE, n is set to 50, but due to our experiments run on a large-scale testbed, we consider top n=1000 documents to be relevant in our implementation.

Table 5.2 shows the precision and recall metrics for CORI and ReDDE with

fixed cutoff T=1,3,5,7,10,15, and 20. As compare with Exhaustive Search, both CORI and ReDDE have never achieved better precision or recall at rank 100, even 20 shards are searched. Although our testbed is cooperative and CSI was randomly sampled, ReDDE is performing better on recalls, MAP. and NDCG than CORI, with any number of cutoff T. Also, ReDDE has better precisions than CORI with T<=10. With T=1, CORI has results that obvious lower than other cutoffs, which shows the shard ranked 1st by CORI is not relevant for most of the queries. These experiment results can support the conclusions from the previous shard selection works[23, 26, 30] that ReDDE performs better than CORI in most of the test setups.

We need to explain the reason that CORI has lower effectiveness than ReDDE while CORI computes the score from statistics of the whole collection, and ReDDE estimates the ranking from 0.5% randomly sampled documents of the collection. From the case analysis on false-positive shards CORI retrieved, we found that CORI is not able to retrieve relevant shards for multi-term queries most of the time. The shard retrieved by CORI usually has very high term frequency on one of the multiple terms, but not all of them. From These results, we generate a hypothesis that CORI might have less effective results because it divides the multi-term queries into unigrams and aggregate the unigram scores. Therefore, we test this hypothesis for all queries in our testbed, and the experiment result is showed in next section.

Run	T	P@30	P@100	MAP	R@30	R@100	NDCG	C_t	C_l
Exhaustive	92	0.254	0.189	0.181	0.174	0.374	0.429	12.67	0.29
CORI	1	0.156	0.092	0.082	0.084	0.151	0.178	0.20	0.20
CORI	3	0.218	0.144	0.127	0.131	0.251	0.278	0.56	0.22
CORI	5	0.241	0.159	0.145	0.151	0.294	0.321	0.89	0.24
CORI	7	0.251	0.171	0.156	0.150	0.307	0.347	1.20	0.24
CORI	10	0.255	0.178	0.164	0.158	0.329	0.370	1.67	0.25
CORI	15	0.263	0.186	0.172	0.162	0.344	0.387	2.41	0.26
CORI	20	0.262	0.187	0.175	0.174	0.357	0.397	3.12	0.26
ReDDE	1	0.222	0.133	0.123	0.134	0.230	0.259	0.24	0.24
ReDDE	3	0.259	0.168	0.157	0.171	0.324	0.350	0.61	0.26
ReDDE	5	0.263	0.174	0.166	0.177	0.343	0.377	0.95	0.27
ReDDE	7	0.260	0.180	0.172	0.178	0.355	0.391	1.27	0.27
ReDDE	10	0.256	0.182	0.172	0.175	0.360	0.400	1.73	0.28
ReDDE	15	0.257	0.184	0.175	0.177	0.363	0.409	2.47	0.29
ReDDE	20	0.259	0.186	0.178	0.179	0.367	0.415	3.20	0.29

Table 5.2: CORI and ReDDE with fixed cutoff (T) = 1,3,5,7,10,15,20. None of CORI and ReDDE results has significant better result than Exhaustive Search. Statistical significance testing was performed using paired t-test at $p<0.05$.

5.3 Effect of Query Length

In this section, we test our hypothesis that CORI's lower effectiveness on shard ranking comes from the unigram representation on the multi-term user queries. Thus, we divide the test queries into 52 single-term ones and 142 multi-term ones and compute their metrics separately. Table 5.3 are results for 52 single-term queries. CORI is higher across all the metrics the search effectiveness with than that with ReDDE, which is exactly the opposite trend in 142 multi-term queries shown by table 5.4. From these results, we confirmed that CORI's lower effectiveness comes

from the multi-term queries. Also, we found ReDDE has lower effectiveness than CORI in single-term queries, mostly come from terms with double meanings such as *avp* or *iron*. 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 search intention of the user for that query. Shards with the major meaning have higher document frequency (df) than shards with the minor meaning, and thus documents with major meaning only are searched. This reduces the false-positive documents (documents related to the minor meaning of query) from the result, and thus improves the search precision.

When comparing the search cost of single-term queries and multi-term queries, we found their posting list sizes have large differences in average. By further analysis of each query, the largest part of posting list sizes come from stop words in the queries. Since single-term queries must not contain stopwords, the posting list sizes and estimated search costs are much smaller than multi-term queries.

5.4 Unigram+Bigram approaches

As mentioned in Section 3.1, we develop the Uni+Bi approach due to CORI's lower precision and recall in multi-term queries. Table 5.5 shows the results with CORI and ReDDE after applied Uni+Bi query reformation. As expected from the hypothesis that CORI requires phrasal information, CORI_Uni+Bi has significant improve-

Run	T	P@30	P@100	MAP	R@30	R@100	NDCG	C_t	C_l
Exhaustive	92	0.240	0.188	0.161	0.127	0.296	0.422	0.59	0.04
CORI	1	0.172	0.101	0.096	0.093	0.156	0.192	0.04	0.04
CORI	3	0.256	0.173	0.148	0.138	0.263	0.318	0.08	0.04
CORI	5	0.256	0.174	0.150	0.138	0.269	0.342	0.12	0.04
CORI	7	0.265	0.184	0.156	0.127	0.277	0.360	0.15	0.04
CORI	10	0.260	0.189	0.163	0.132	0.296	0.391	0.18	0.04
CORI	15	0.264†	0.197	0.168	0.139	0.311	0.408	0.23	0.04
CORI	20	0.260†	0.198	0.166	0.139†	0.309	0.409	0.27	0.04
ReDDE	1	0.224	0.127	0.116	0.113	0.184	0.242	0.03	0.03
ReDDE	3	0.268	0.170	0.151	0.138	0.255	0.333	0.08	0.04
ReDDE	5	0.263	0.175	0.151	0.140	0.273	0.361	0.10	0.04
ReDDE	7	0.254	0.178	0.154	0.138	0.283	0.374	0.12	0.04
ReDDE	10	0.250	0.184	0.155	0.131	0.295	0.387	0.15	0.04
ReDDE	15	0.248	0.184	0.157	0.133	0.293	0.396	0.20	0.04
ReDDE	20	0.253	0.191	0.166	0.134	0.302	0.414	0.24	0.04

Table 5.3: The evaluation metrics on single-term queries of CORI and ReDDE with fixed cutoff (T) = 1,3,5,7,10,15,20. †:Significant improvements from Exhaustive Search. Statistical significance testing was performed using paired t-test at $p<0.05$.

ment. P@30 is significantly better than Exhaustive Searchafter T=7, and R@30 is significantly better than Exhaustive Searchafter T=15. P@100, R@100, and MAP are marginally better than Exhaustive Search after T=15. The improvement of CORI with T=1 shows that CORI_Uni+Bi has better capability to rank relevant shards at 1st as compare with the unigram version. Although it was not statistically significant, CORI_Uni+Bi has better results than ReDDE_Uni+Bi after T=3. Clearly, CORI benefits much more from the phrasal query representation as compare with ReDDE. ReDDE also benefits from using bigram queries against CSI, but the improvements are relatively small since ReDDE has less room to improve for

Run	T	P@30	P@100	MAP	R@30	R@100	NDCG	C_t	C_l
Exhaustive	92	0.259	0.189	0.189	0.192	0.403	0.432	17.2	0.38
CORI	1	0.150	0.089	0.077	0.081	0.150	0.173	0.27	0.27
CORI	3	0.204	0.133	0.119	0.129	0.247	0.263	0.75	0.30
CORI	5	0.236	0.154	0.142	0.156	0.303	0.313	1.21	0.32
CORI	7	0.245	0.166	0.156	0.158	0.317	0.343	1.64	0.32
CORI	10	0.254	0.173	0.164	0.167	0.341	0.362	2.28	0.33
CORI	15	0.262	0.182	0.173	0.171	0.356	0.379	3.31	0.35
CORI	20	0.263	0.184	0.178	0.186	0.375	0.392	4.30	0.35
ReDDE	1	0.221	0.134	0.125	0.141	0.247	0.266	0.31	0.31
ReDDE	3	0.255	0.167	0.159	0.184	0.349	0.356	0.80	0.34
ReDDE	5	0.262	0.174	0.172	0.191	0.368	0.383	1.27	0.35
ReDDE	7	0.262	0.181	0.178	0.193	0.381	0.397	1.70	0.36
ReDDE	10	0.258	0.181	0.178	0.191	0.384	0.404	1.32	0.36
ReDDE	15	0.260	0.183	0.182	0.194	0.388	0.413	3.32	0.37
ReDDE	20	0.261	0.184	0.182	0.195	0.391	0.416	4.31	0.38

Table 5.4: The evaluation metrics on multiple term queries of CORI and ReDDE with fixed cutoff (T) = 1,3,5,7,10,15,20. None of CORI and ReDDE results on multi-term queries has significant better result than Exhaustive Search. Statistical significance testing was performed using paired t-test at $p<0.05$.

multi-term queries.

In summary, CORI_Uni+Bi provides the best search effectiveness until now. In the next section, we will investigate if we can improve the performance further.

5.5 LeToR-S

The table 5.6 shows the results of LeToR-S (Learning to Rank for Shards) approach.

The first promising observation in these results is that LeToR-S significantly

Run	T	P@30	P@100	MAP	R@30	R@100	NDCG	C_t	C_l
Exhaustive	92	0.254	0.189	0.181	0.174	0.374	0.429	12.67	0.29
CORI	1	0.215	0.124	0.120	0.133	0.223	0.247	0.21	0.21
CORI	3	0.267	0.174	0.166	0.174	0.327	0.353	0.57	0.23
CORI	5	0.272	0.182	0.174	0.177	0.342	0.375	0.91	0.25
CORI	7	0.275†	0.187	0.178	0.171	0.350	0.393	1.23	0.25
CORI	10	0.271†	0.188	0.181	0.180	0.363	0.408	1.68	0.26
CORI	15	0.274†	0.194	0.187	0.186†	0.381	0.421	2.43	0.26
CORI	20	0.270†	0.195	0.187	0.185†	0.383	0.423	3.14	0.27
ReDDE	1	0.218	0.130	0.121	0.134	0.226	0.254	0.23	0.23
ReDDE	3	0.263	0.170	0.161	0.171	0.320	0.346	0.58	0.26
ReDDE	5	0.265	0.177	0.169	0.177	0.345	0.375	0.90	0.26
ReDDE	7	0.264	0.181	0.172	0.179	0.355	0.386	1.24	0.27
ReDDE	10	0.263	0.184	0.175	0.182	0.363	0.398	1.66	0.27
ReDDE	15	0.262	0.186	0.177	0.183	0.368	0.407	2.36	0.28
ReDDE	20	0.262	0.186	0.178	0.183	0.369	0.415	3.04	0.28

Table 5.5: Modified CORI and ReDDE using Bigram+Unigram statistics with fixed cutoff (T) = 1,3,5,7,10,15,20. †:Significant improvements from Exhaustive Search. Statistical significance testing was performed using paired t-test at $p<0.05$.

outperforms the current best approach, CORI_Uni+Bi, at lower ranks when the Cutoff T $\leq$ 10. At early ranks, the two approaches have non-significant different precision and recall. To understand how LeToR-S is the current best approach, we analyze the certain queries which have much better result in LeToR-S as compare with CORI_Uni+Bi.

For one of the queries, *getting organized*, CORI_Uni+Bi ranks the shard with the most number of relevant documents at 11th position, while LeToR-S ranks it at 3rd. For this query, LeToR-S benefit from the meta-data features. The ranker learned one of the meta-data features, *url* field score, can distinguish the relevant shards

Run	T	P@30	P@100	MAP	R@30	R@100	NDCG
Exhaustive	92	0.254	0.189	0.181	0.174	0.374	0.429
CORI_Uni+Bi	1	0.215	0.124	0.120	0.133	0.223	0.247
CORI_Uni+Bi	3	0.267	0.174	0.166	0.174	0.327	0.353
CORI_Uni+Bi	5	0.272	0.182	0.174	0.177	0.342	0.375
CORI_Uni+Bi	7	0.275†	0.187	0.178	0.171	0.350	0.393
CORI_Uni+Bi	10	0.271†	0.188	0.181	0.180	0.363	0.408
CORI_Uni+Bi	15	0.274†	0.194	0.187	0.186†	0.381	0.421
CORI_Uni+Bi	20	0.270†	0.195	0.187	0.185†	0.383	0.423
LeToR-S	1	0.230	0.139¶	0.124	0.139	0.246¶	0.266
LeToR-S	3	0.281†	0.183	0.174	0.180	0.342	0.380¶
LeToR-S	5	0.275†	0.185	0.176	0.180	0.351	0.396¶
LeToR-S	7	0.274†	0.190	0.181	0.177	0.365	0.406¶
LeToR-S	10	0.270†	0.194¶	0.186	0.179	0.377¶	0.417¶
LeToR-S	15	0.268†	0.194	0.185	0.177	0.376	0.420
LeToR-S	20	0.264†	0.190	0.183	0.176	0.373	0.421

Table 5.6: LeToR-S statistics with fixed cutoff (T) = 1,3,5,7,10,15,20. †:Significant improvements from Exhaustive Search. ¶:Significant improvements from CORI_Uni+Bi to LeToR-S. Statistical significance testing was performed using paired t-test at $p<0.05$.

and non-relevant shards for this query. The shards which gathered documents that contain *getting organized* in their *url* field are relevant to the query. A few more examples that benefit from 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 lower rank than LeToR-S. The common words such as *war* or *city* in the queries length more than 2 reduced the ranking precision of CORI_Uni+Bi, while LeToR-S ranked the relevant shards by the feature of field scores.

Another feature group plays an important role in LeToR-S will be the CORI

MIN score features. The CORI MIN score is the minimum CORI score of one term in a multi-term query. High CORI MIN score indicates that every term in the query has high CORI score. For one test query "martha stewart and imclone", the most relevant shard is raked 13th by CORI_Uni+Bi method and 7th by LeToR-S. By analyzing different feature settings on this query, we found that LeToR-S benefits from CORI SUM score and CORI MIN score features, the most relevant shard has both high CORI SUM scores and high MIN scores, which the minimum score is from the term "imclone". Through the CORI MIN score feature, several false-positive shards are eliminated by LeToR-S from the shard ranking. These false-positive shards have high CORI SUM score due to only few terms in the long query have high frequencies. With the CORI MIN score, the ranker can pull down the ranks of false-positive shards that do not have high CORI scores for every query term.

The results in the table 5.6 also indicate that at early ranks LeToR-S performs significantly better than Exhaustive Search. The reason is the same as CORI, where the significant improvement comes from single-term queries with double meanings. Filtering out the shards related to the minor meaning of the single-terms reduces a lot of false-positive documents. By comparing the results of different cutoff T, P@30 and R@30 have the best result with T=3, and the precision and R@100 have the best result with T=10. MAP has the highest value when T=10, then it decreases while the T increase. However, after T>10, most of the metrics are marginally worse than CORI_Uni+Bi. The reason is that the training purpose was set to maximize

the NDCG@10 for shard ranking, the shard ranking after rank 10 does not affect the training output, which results in the performance after T>10 degrades sooner than CORI_Uni+Bi. The effect of the target metric to be optimized in the LeToR approach will require further exploration in the future.

In summary, we found the CORI MIN scores and CORI field scores are important input features of learning to rank shards model for the topical shards. Learning the decision boundary among the combination of above CORI scores turns out to have better results than the CORI_Uni+Bi score.

5.6 LeToR-SWP

The table 5.7 records the results for shard ranking approach with PRF described in Section 3.3. As mentioned in Section 3.3, we apply this approach only to single-term queries in order to minimize the side effects of applying PRF. Out of the 194 queries, 52 are single-term queries, out of which 34 have a Wikipedia entry. 26 out of these 34 queries get expanded because for the 8 remaining queries, the *profile* of the top-ranked shard does not share any term with the corresponding Wikipedia summary. Although this is a highly restrictive approach and only expands a small fraction of the queries, it provided consistent improvements in performance, as evidenced by the results in Table 5.7. At early ranks, the improvements over the previous best approach, LeToR-S, are statistically significant.

Table 5.8 shows the terms included in the 26 queries before and after expansion.

Using this technique one of the queries, *iron* is expanded to *iron powder weld carbon temperature* and leads to 118% improvement over Exhaustive Search in MAP@1000 at T=10. There are obvious queries have their meanings enriched such as *joints*, is expanded to *joints brain bone* and *sat*, is expanded to *sat academy admission mathematics assessment.* Many of these single-term queries have multiple meanings, and the expansion added the terms related to the most specific meaning by intersecting PRF and Wikipedia summary. Some expended queries have more than 10 terms such as *starbucks*, is expanded to *starbucks taste snack chip juice bean cream beverage fresh tea serving coffee.* Although the long expended query seems not precise, MAP@1000 for this query is improved 18% as compare with Exhaustive Search at T=10. For all results at T=10, LeToR-SWP is 11%, 4%, 6%, 7%, 2% higher than Exhaustive Search in P@30, P@100, MAP@1000, R@30, and R@100, respectively. The only metric that is not improved with LeToR-SWP is NDCG@1000.

By looking at the expanded queries results, we found PRF improved the search effectiveness significantly. 25 of the 26 expanded queries have better or the same p@30 and r@30 as compare with applying original queries. The only query has reduced result is "gps", which expanded to "gps limitation authorize accurate transmit". Since GPS is a term that has no word ambiguity, the expended query reduced the precision and did not improve the recall as much.

Run	T	P@30	P@100	MAP	R@30	R@100	NDCG
Exhaustive	92	0.254	0.189	0.181	0.174	0.374	0.429
LeToR-S	1	0.230	0.139	0.124	0.139	0.246	0.266
LeToR-S	3	0.281†	0.183	0.174	0.180	0.342	0.380
LeToR-S	5	0.275†	0.185	0.176	0.180	0.351	0.396
LeToR-S	7	0.274†	0.190	0.181	0.177	0.365	0.406
LeToR-S	10	0.270†	0.194	0.186	0.179	0.377	0.417
LeToR-S	15	0.268†	0.194	0.185	0.177	0.376	0.420
LeToR-S	20	0.264†	0.190	0.183	0.176	0.373	0.421
LeToR-SWP	1	0.237	0.142	0.132	0.145	0.252	0.274
LeToR-SWP	3	0.282†	0.181	0.170	0.176	0.333	0.370
LeToR-SWP	5	0.288†	0.190	0.182	0.189†¶	0.360	0.400
LeToR-SWP	7	0.282†	0.193	0.184	0.186¶	0.370	0.410
LeToR-SWP	10	0.281†¶	0.197†	0.191†	0.186†¶	0.382	0.423
LeToR-SWP	15	0.273†	0.193	0.186	0.181	0.376	0.424
LeToR-SWP	20	0.271†¶	0.192	0.186	0.180¶	0.376	0.425

Table 5.7: LeToR-SWP statistics with fixed cutoff (T) = 1,3,5,7,10,15,20. †:Significant improvements from Exhaustive Search. ¶:Significant improvements from LeToR-S to LeToR-SWP. Statistical significance testing was performed using paired t-test at $p<0.05$.

5.7 Effect of Distribution of Relevant Documents

In this section, we want to analyze the reason that Selective Search approaches still have lower effectiveness than Exhaustive Search at lower ranks, especially on recalls and NDCGs. Our hypothesis is that the effectiveness of shard ranking algorithm is related to how the relevant documents spread across the shards. If the relevant documents are spread across only a few shards(i.e few relevant shards), it is easier to rank all relevant shards before non-relevant shards. Otherwise, ranking the shards will be a much more difficult task. In order to test this hypothesis on the collection,

Single-term queries	Expanded queries
toilet	toilet cap battery adjust portable empty heat
map	map commerce interior elements dynamic publishing...
dinosaurs	dinosaurs exploration discovery evolution context...
volvo	volvo ford truck
euclid	euclid geometry elements
starbucks	starbucks taste snack chip juice bean cream beverage...
inuyasha	inuyasha anime monster tale episode powerful
atari	atari console arcade publisher
gps	gps limitation authorize accurate transmit
iron	iron powder weld iron carbon temperature
moths	moths insect species
pvc	pvc rubber pipe leather plastic
tornadoes	tornadoes southeast
dieting	dieting calorie fat diet sustain healthy treat
afghanistan	afghanistan muslim regime islam arab iran pakistan...
joints	joints bone brain
ocd	ocd difficulty inhibit medication symptom disorder ...
rice	rice culinary
sat	sat academy admission mathematics assessment
figs	figs shrub genus wildlife native fruit zone species
grilling	grilling fry pan grill roast pork
arkansas	arkansas mississippi
barbados	barbados atlantic sq situated caribbean 400 tourist
lipoma	lipoma tumor tissue
vanuatu	vanuatu guinea fiji solomon
fybromyalgia	fybromyalgia diagnostic affected disorder symptom...

Table 5.8: Expanded queries

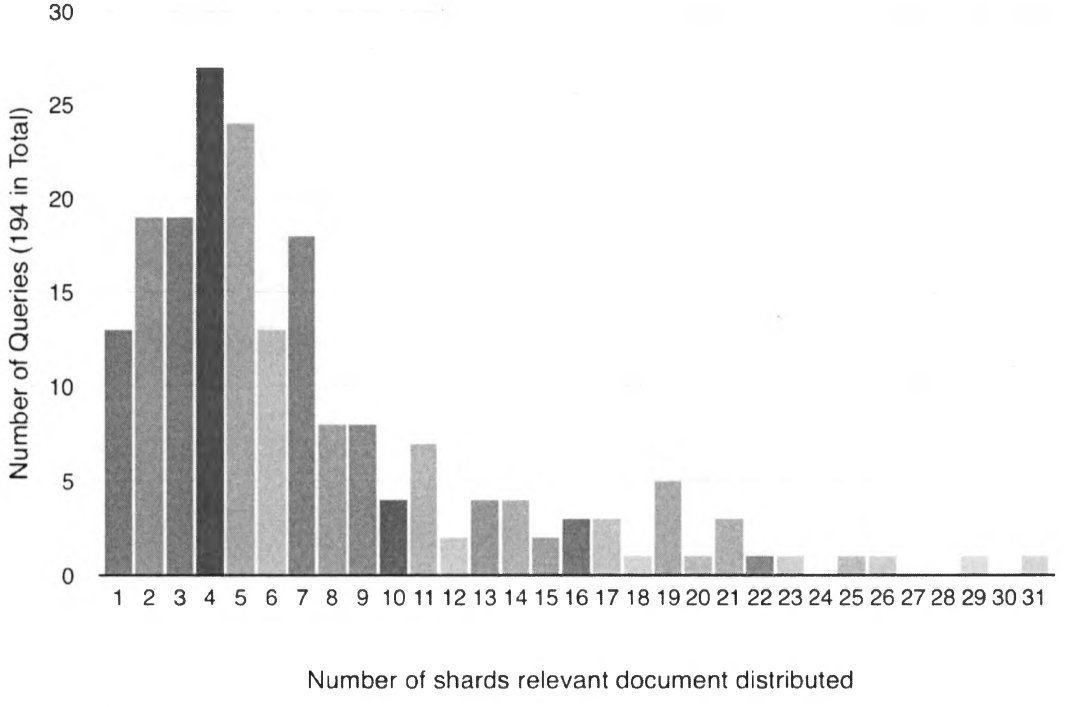


Figure 5.1: Histogram of number of shards containing relevant documents for the query.

we conduct the experiment on queries of two groups, separated by the number of relevant shards. Also, we plotted a histogram to display the distribution of relevant shards.

Figure 5.1 is 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

Search Approach	T	P@30	P@100	MAP	R@30	R@100	NDCG
Exhaustive	92	0.218	0.153	0.166	0.198	0.399	0.405
CORI	10	0.229	0.154	0.165	0.178	0.361	0.369
ReDDE	10	0.225	0.154	0.169	0.202	0.397	0.391
CORI_Uni+Bi	10	0.241†	0.163	0.179†	0.205	0.401	0.400
ReDDE_Uni+Bi	10	0.233†	0.156	0.171	0.209	0.400	0.385
LeToR-S	10	0.239†	0.164†	0.178†	0.204	0.411	0.401
LeToR-SWP	10	0.245†	0.165†	0.179†	0.209	0.413	0.404

Table 5.9: 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	T	P@30	P@100	map	R@30	R@100	NDCG
Exhaustive	92	0.332	0.265	0.215	0.124	0.320	0.482
CORI	10	0.312	0.228	0.162	0.114	0.258	0.372
ReDDE	10	0.322	0.243	0.179	0.118	0.279	0.418
CORI_Uni+Bi	10	0.336	0.243	0.187	0.125	0.282	0.424
ReDDE_Uni+Bi	10	0.331	0.246	0.185	0.122	0.283	0.424
LeToR-S	10	0.340	0.261 ¶	0.204 ¶	0.124	0.303 ¶	0.452 ¶
LeToR-SWP	10	0.357	0.263 ¶	0.209 ¶	0.130	0.305 ¶	0.457 ¶

Table 5.10: Results on 61 Queries with number of relevant shard > 7. ¶ indicates statistically significant improvement when comparing LeToR approaches with MTD_Uni+Bi. Underline indicates significantly worse values when compared to Exhaustive Search. Statistical significance testing was performed using paired t-test at $p<0.05$.

as 31 shards, indicating that the topic-based sharding approach still has limitation to aggregate the relevant documents into few shards for some queries. However, we believe that the approaches we developed should have generally improved precisions and recalls for the 70% of the queries with less than 7 relevant shards. To prove this capability of our approaches, we separate the 194 queries into two groups based on the spread cutoff of ≤ 7. This divides the queries into one group of 133 queries with 7 or fewer relevant shards, and the other group of 61 queries with > 7 relevant shards.

The results for the two query groups with the various search approaches are given in Tables 5.9 and 5.10. For the first group of queries (Table 5.9) LeToR approaches (and also CORI_Uni+Bi) provides statistically-significant improvements over Exhaustive Search in precision at both, high and low ranks. Furthermore, the improvements in precision do not sacrifice the recall. For the LeToR approaches, the recall is not significantly different from Exhaustive Search at all ranks, and even NDCG is not statistically worse than Exhaustive Search. In the queries with relevant shards <= 7, the LeToR approaches can balance precision and recall of the document ranking.

On the other hand, Table 5.10 shows the results on "difficult queries". The results of all Selective Search approaches have lower P@100, R@100, MAP, NDCG than Exhaustive Search. The results reminded that the Selective Search approaches benefit from effective topical clustering methods. If the clustering methods had dis-

T	A	Search Approach	P@30	P@100	MAP	R@30	R@100	NDCG
92	50.22	Exhaustive	0.254	0.189	0.181	0.174	0.374	0.429
1	0.561	CORI_Uni+Bi	0.215	0.124	0.120	0.133	0.223	0.247
	0.560	LeToR-SWP	0.231	0.139¶	0.127	0.139¶	0.244	0.265
3	1.672	CORI_Uni+Bi	0.267	0.174	0.166	0.174	0.327	0.353
	1.668	LeToR-SWP	0.280†	0.181	0.170	0.175	0.333	0.373¶
5	2.773	CORI_Uni+Bi	0.272	0.182	0.174	0.177	0.342	0.375
	2.775	LeToR-SWP	0.285†	0.189	0.180	0.187	0.356	0.399¶
7	3.874	CORI_Uni+Bi	0.275†	0.187	0.178	0.171	0.350	0.393
	3.867	LeToR-SWP	0.282†	0.192	0.183	0.181	0.367¶	0.408¶
10	5.513	CORI_Uni+Bi	0.271†	0.188	0.181	0.180	0.363	0.408
	5.508	LeToR-SWP	0.280†	0.196 ¶	0.188	0.184†	0.379¶	0.421¶
15	8.247	CORI_Uni+Bi	0.274†	0.194	0.187	0.186†	0.381	0.421
	8.237	LeToR-SWP	0.273†	0.193	0.186	0.181	0.376	0.424
20	10.970	CORI_Uni+Bi	0.270†	0.195	0.187	0.185†	0.383	0.423
	10.957	LeToR-SWP	0.271†	0.192	0.186	0.180	0.376	0.425

Table 5.11: Results for Exhaustive, and Selective Search with CORI_Uni+Bi and with LeToR-SWP at various shard cutoffs (T). A: The average search space size per query in million documents. ¶ indicates statistically significant improvement when comparing LeToR-SWP with CORI_Uni+Bi. † indicates statistically significant improvement when comparing LeToR-SWP 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 $p<0.05$.

tinguished the topical-related documents concentrate in a few shards, the Selective Search can achieve closer results to the ideal upper-bounds as well.

5.8 Effect of Number of Shards Searched

In the previous experiments we use fixed shard cutoff (T) at 1,3,5,7,10,15, and 20 as the standard settings. In this Section, we are going to explore the effect of parameter T on the Selective Search approaches, CORI_Uni+Bi and LeToR-SWP, and compare them to Exhaustive Search.

Table 5.11 provides the results of this analysis. At early ranks, LeToR-SWP performs on par with Exhaustive Search while searching only the top three shards. The corresponding search cost, approximated by A, is orders of magnitude lower for LeToR-SWP than for Exhaustive Search. When comparing LeToR-SWP with CORI_Uni+Bi, the former consistently outperforms the latter at all the shard cutoff values. Even the average sizes of the searched shards for LeToR-SWP are marginally lower than those with CORI_Uni+Bi, indicating CORI_Uni+Bi might bias larger shards rather than shard condensed with relevant documents. As more shards are searched, LeToR-SWP achieves higher recall at the low ranks. With T>=10, LeToR-SWP becomes not significantly worse than Exhaustive Search on NDCG.

5.9 Dynamic Shard Rank Cutoff Estimation

In the previous experiments, all of the Selective Search approaches are evaluated among the fixed shard rank cutoffs, 1,3,5,7,10,15, and 20. In this section, the experiments are aiming to explore the effect of different dynamic cutoff estimators stated

in Section 3.4. Tables 5.12 and 5.13 show the results with dynamic shard rank cutoff estimators. For comparison, the result of Exhaustive Search and Selective Search with fixed cutoff T=10 are also showed in the first two rows. For all cutoff estimators, we use the current best shard ranking approach, LeToR-SWP, to compare the performance of prediction. Table 5.12 gives the precision-oriented experiments where the cutoff estimators were optimized for search effectiveness at early ranks. The results for the corresponding recall-oriented experiments are in Table 5.13. This separate optimization of cutoff estimators is necessary because much fewer shards have to be searched for a precision-oriented task than for a recall-oriented task. The values reported in the second column (AvgT) is the average shard cutoff prediction of each estimator.

To achieve two different search goals, precision-oriented and recall-oriented, we modified two parameters, hard cutoff upper-bound and the M value defined in Section 3.4. We set an upper-bound 10 as the maximum shards selected for the precision-oriented experiments, and 50 for the recall-oriented experiments. The M value is set to 20 for the precision-oriented experiments and 92(total number of shards) for the recall-oriented experiments. However, the effects on the precision and recall of these two parameters will require generalized exploration in the future.

Table 5.12 shows that all the cutoff estimators have significantly better precision-oriented metrics than Exhaustive Search. Among the three estimators, PK2 has the best overall result. Although PK2 searches 7 shards averagely, the corresponding

Cutoff Approach	AvgT	P@30	MAP@30	R@30	NDCG@30
Exhaustive	92	0.254	0.071	0.174	0.215
Fixed	10	0.281†	0.084†	0.186†	0.244†
PK2 M=20	7	0.289†¶	0.089†¶	0.187	0.252†¶
PK3 M=20	8	0.285†	0.084†	0.181	0.244†
Rank-S B=3	8	0.283†	0.085†	0.187	0.245†

Table 5.12: Precision-oriented metrics for Exhaustive Search, LeToR-SWP with fixed cutoff, and with precision-oriented optimization shard rank cutoff approaches. AvgT: Average shard rank cutoff. †: statistically significant improvements over Exhaustive Search. ¶: statistically significant improvements as compared to Fixed Cutoff of 10. Statistical significance testing was performed using paired t-test at $p<0.05$.

Cutoff Approach	AvgT	P@1000	MAP@1000	R@1000	NDCG@1000
Exhaustive	92	0.042	0.181	0.755	0.429
Fixed	25	0.041▽	0.185†	0.728▽	0.427
PK2 M=92	26	0.041	0.191¶†	0.731▽	0.432
PK3 M=92	25	0.041	0.189†	0.727▽	0.429
Rank-S B=1.3	22	0.041	0.187†	0.724▽	0.428

Table 5.13: Recall-oriented metrics for Exhaustive Search, LeToR-SWP with fixed cutoff, and with recall-oriented optimization shard rank cutoff approaches. † and ▽: statistically significant improvement or degradation over Exhaustive Search, respectively. ¶: statistically significant improvements as compared to Fixed Cutoff of 25. Statistical significance testing was performed using paired t-test at $p<0.05$.

P@30, MAP@30, and NDCG@30 values are significantly higher than the fixed cutoff T=10. This AvgT column indicates the dynamic cutoff estimators reduced the false-positive documents in the document ranking by searching fewer shards. The improvements with PK2 over Exhaustive Search are 14%, 25%, 7%, 17% in P@30, MAP@30, R@30, and NDCG@30, respectively. PK3 has better P@30 but worse

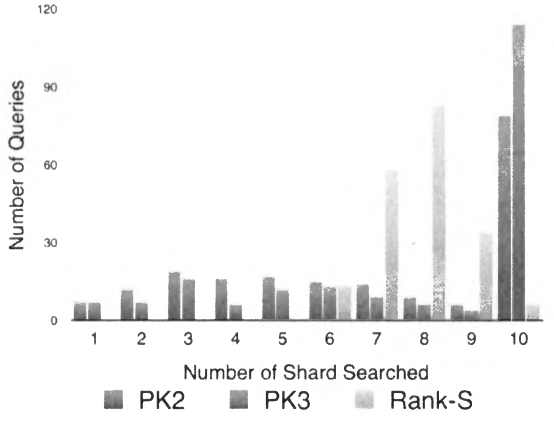


Figure 5.2: Distribution of precision-oriented shard rank cutoff predictions by PK2, PK3, and Rank-S.

R@30 than the fixed cutoff setting. Rank-S has every metric improved from the fixed cutoff setting, but not significantly.

The recall-oriented results in Table 5.13 show that PK2 has P@1000 and NDCG@1000 similar to Exhaustive Searchand Fixed, but still has significant lower R@1000 as compare with Exh. However, PK2 improves MAP@1000 significantly, and achieved the highest value among all approaches have been developed. PK3 and Rank-S improved MAP and NDCG from Fixed, but the R@1000 is worse. It indicates PK3 and Rank-S did not determine the ideal cutoff and removed some relevant shards for several queries.

Figures 5.2 and 5.3 shows the distribution of rank cutoffs predicted by PK2, PK3, and Rank-S for the precision-oriented setting and recall-oriented setting. For precision-oriented settings, PK3 estimates 10 or more cutoff for most of the queries

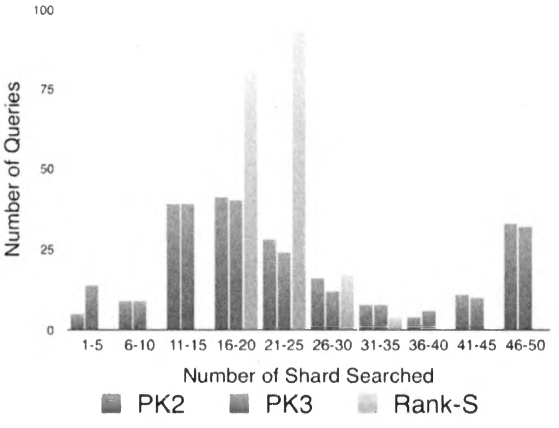


Figure 5.3: Distribution of recall-oriented shard rank cutoff predictions by PK2, PK3, and Rank-S.

and hit the boundary we set, which makes it improved less from the fixed cutoff 10. The Rank-S shows a prediction concentrated from 6 to 10 shards, which indicates the exponential decaying method did not give a high variation of prediction for different score distribution. For recall-oriented settings, the PK2 and PK3 have very similar cutoff distribution, while Rank-S shows a different distribution that much more narrow spread. From the above comparison, we found that the current best estimator, PK2, is also the estimator that gives the widest range of cutoffs for different queries. This distribution helps us to understand the effectiveness of PK2. As plotted in Section 5.7, the number of relevant shards is very different from the queries, which matches the distribution of prediction given by PK2.

5.10 Implementation

The web interface is implemented in Javascript and Node.js, the back-end query processing infrastructure is written in Python and Unix Shell Script. The data visualization part exploits Bootstrap and D3 library.

Figure 5.5 shows the flowchart of the implementation, which consists of two phases:- 1. Preprocessing phase, and 2. Online phase. The pre-processing phase is responsible for organizing the data and is executed only once. The online phase is responsible for query execution and thus is run multiple times.

Figure 5.6 shows the snapshot of returned search results from searching a 2-term query "Artificial Intelligence" in CategoryB dataset. The system executes both Exhaustive Search and our approaches and generates the comparison between them. The result page provides visualized analysis information for each user query. The returned results are documents with Rank, Document Title, Document ID, and Shard ID that contains the document. The content of documents can be retrieved through the link of its Title or ID. In this example, Selective Search approach searched 4 of 92 shards. By comparing with the Exhaustive Search result, the documents from rank 2 to 4 was excluded in the Selective Search result, since the shards contain them are not searched.

Figure 5.7 shows the shard profile chart, which can be accessed through the link of Shard IDs. This bar chart displays top related shard keywords sorted by their DFISF score. Figure 5.8 shows the bar chart of scores of each shard sorted in

the descending order, which can be accessed by clicking the "Shard Rank Detail" button on the result page. The bar graph is scaled according to the minimum and maximum score to strengthen the visualization of score dropping.

Figure 5.9 shows the snapshot of returned search results from searching a single-term query "Robot". Single-term queries are expanded by PRF mentioned in Section 3.3. The system shows the expanded query in the stemmed format: "robot worker assist manufacture force".

5.11 Summary

In this chapter, we conducted nine experiments to answer the questions of how the shard ranking approaches and dynamic cutoff approaches improve the precision and recall of the document ranking. In the end, our current best approaches achieved the precision significantly higher than Exhaustive Search while sustaining the comparable recall. Finally, we made the web interface to display the improvements of our approaches from Exhaustive Search.

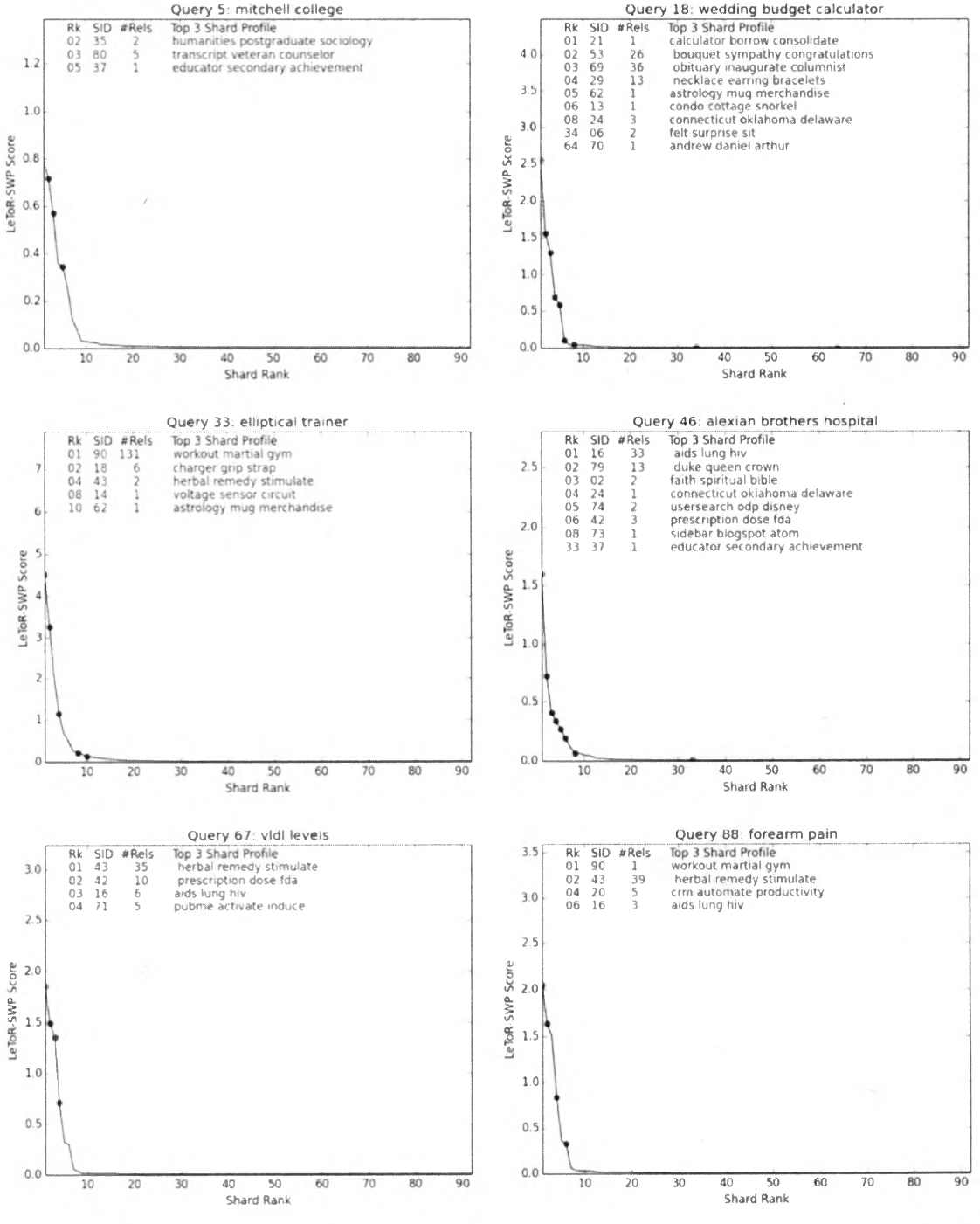


Figure 5.4: *Elbow* formed by LeToR-SWP shard scores. Blue dots represent the shards that contain relevant documents, and the text shows the ranks and profiles of those shards.

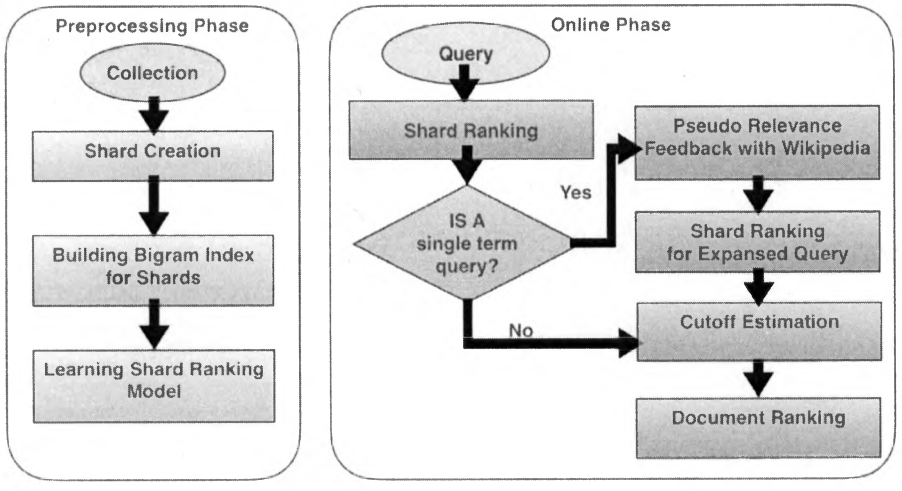


Figure 5.5: Approach flowchart of our system.

Figure 5.6: The snapshot of the search result page for query "artificial intelligence".

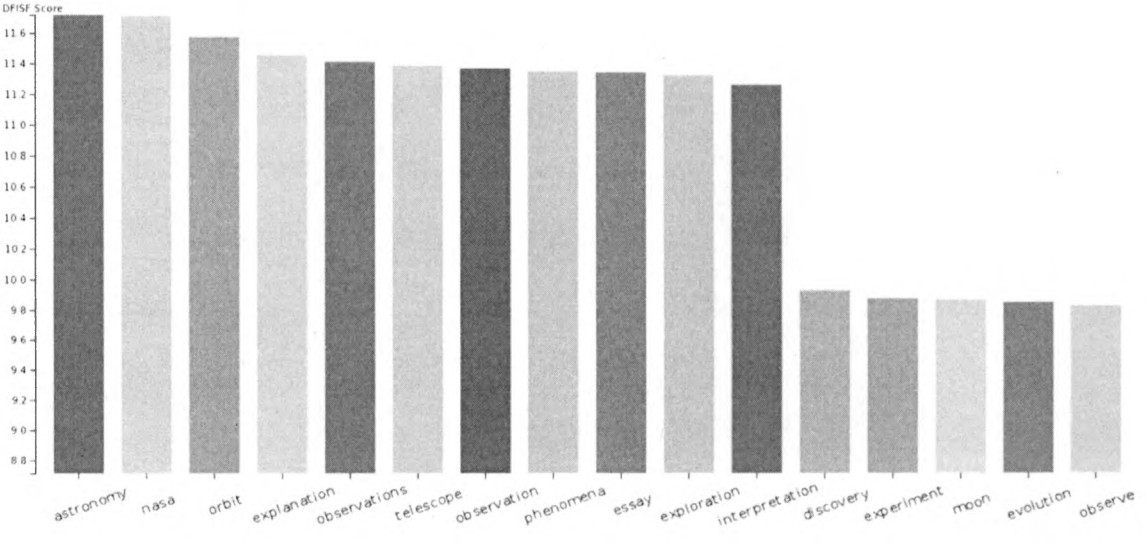


Figure 5.7: The snapshot of the shard profile chart.

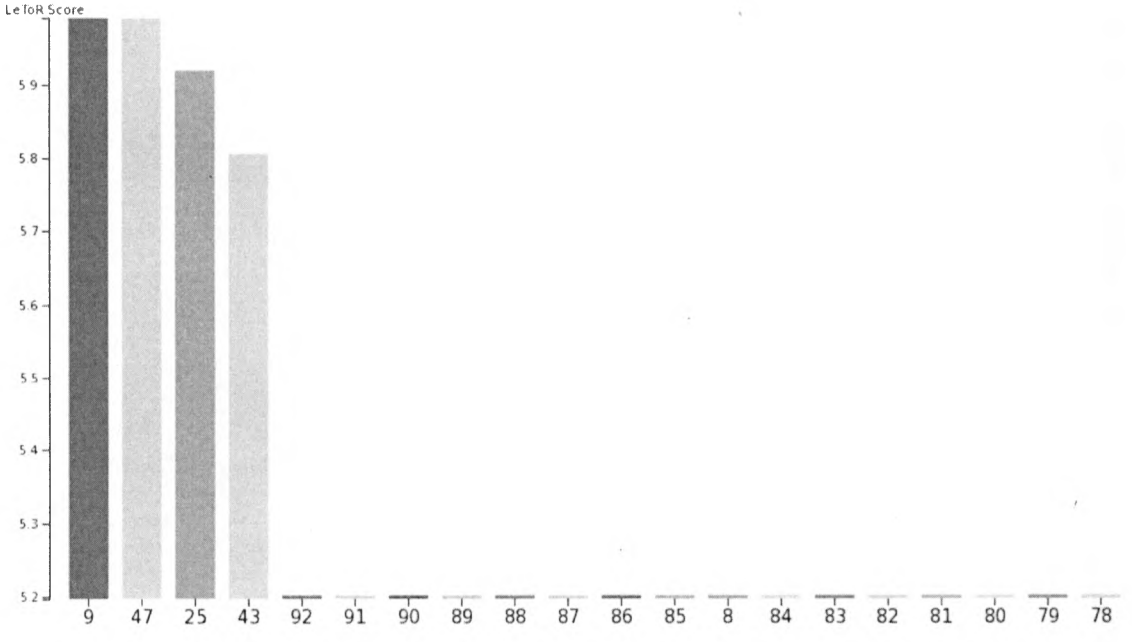


Figure 5.8: The snapshot of the shard score distribution chart.

Figure 5.9: The snapshot of the search result page for query “robot”.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this work, we first analyzed the effectiveness of big-document approach, CORI, and the small-document approach, ReDDE variant, on our test environment. By the analysis we approved that ReDDE has consistently better results while CORI's ranking tends to be more misleading. Then, the main contribution in this work is that we developed three approaches that achieve closer precisions and recalls to the ideal upper-bounds. In the first approach, we make use of the bigram statistic information and improved CORI approach significantly. In the second approach, we introduced learning-to-rank framework for shard ranking, using different CORI scores as the input features and generated better ranking than previous approaches. In the third approach, we applied pseudo relevance feedback and improved the search effectiveness for almost all of the single-term test queries.

As discussed in Si and Callan's work [25, 27], Selective Search is considered useful on improving precision-oriented metrics since it filtered out non-relevant documents in low score shards. After the result analysis, we found P@30 have the most obvious improvements than Exhaustive Search in most of the approaches. This result approved the strength of the Selective Search on the aspect of precision. Furthermore, we extended the potential of Selective Search to achieve higher NDCG in the standard return number of documents, 1000, than Exhaustive Search. The best result comes from LeToR-SWP with PK2 cutoff estimation, which has marginally higher NDCG than Exhaustive Search while only searching 26 shards among 92 shards averagely.

6.2 Future Work

6.2.1 Possible Purposes of Machine Learning using Neural Networks

We have tested SVM binary classification on distinguishing relevant/non-relevant shards. It is considered one useful approach but is less precise than the LeToR-S approach. However, classification or regression using Neural Network has not been tested. Different from SVM, which is an inherently two-class classifier, Neural Network can be applied to multi-class classification and regression tasks.

Multiclass Classification If the shards are classified as more than 2 relevance level, e.g. Highly-Relevant, Relevant, Non-Relevant, it might generate in better

prediction on test queries than the binary classifier and may have competitive result compare to Listwise LeToR approach.

Regression The shard relevancy can be regressed from training query, input features, and target outputs. This approach shares the same spirit of the Pointwise Learning to Rank approach.

Semantic Analysis When the query is formed as a sentence, the semantic analysis might help for the search system to know the purpose of the search. Recurrent Neural Network(RNN) is widely applied in natural language processing tasks such as translation. It takes word inputs sequentially and uses the output for the previous word as the input of next round, which is similar to humans, who can tell the different meaning of same words from their contexts.

6.2.2 Shard Profile Representation

The parameter of how many top DF terms should be selected to compute DFISF score need more exploration. But before exploring the parameters, we need to define a way to evaluate if the shard summary is effective or not. The final metrics on document rankings after WikiPRF is indirect and affected by any other part of the system (Source Selection, CORI score, Ranking model), thus it is not a good way to evaluate the parameter. Besides sorting the unigram terms by their DFISF score, we must explore more other different forms of shard profiles which are also representative of the shards.

Appendix A

Shard Profiles of CategoryB Dataset

SID	Top 5 Terms
1	stocks/billion/commodity/shares/earnings/
2	faith/spiritual/bible/jesus/christ/
3	drum/piano/classical/acoustic/composer/
4	climb/kayak/trek/raft/scenic/
5	appeal/amendment/statute/judicial/litigation/
6	felt/surprise/sit/kept/isnt/
7	lack/decade/poor/struggle/oppose/
8	pepper/butter/onion/chop/fry/
9	astronomy/nasa/orbit/explanation/observations/
10	variable/parameter/instance/initial/integer/
11	ford/honda/brake/toyota/bmw/
12	phrase/hindi/translator/grammar/spoken/
13	condo/cottage/snorkel/surf/sand/
14	voltage/sensor/circuit/heating/diameter/
15	appearance/fictional/portray/cast/comic/
16	aids/lung/hiv/asthma/infectious/
17	ringtone/wanna/yeah/favourite/spotlight/
18	charger/grip/strap/lens/bottle/
19	soil/habitat/insect/binomial/gardening/
20	crm/automate/productivity/outsourc/leverage/
21	calculator/borrow/consolidate/fraud/transaction/
22	laundry/convenient/marriott/motel/cancellation/
23	rip/avi/spyware/antiviru/registry/

Table A.1: The Shard Profiles of topical shards 1-23 of CatgoryB dataset.

SID	Top 5 Terms
24	connecticut/oklahoma/delaware/massachusetts/oregon/
25	molecule/compound/electron/experimental/particle/
26	railway/junction/exit/lane/interstate/
27	nintendo/playstation/ds/console/ps3
28	hi5/none/width/em/beta
29	necklace/earring/bracelets/pendant/diamond/
30	palestinian/muslim/terrorism/terrorist/hama/
31	demography/utc/median/poverty/reside/
32	volleyball/softball/athletic/roster/fame/
33	gossip/horror/globe/critic/kate/
34	della/che/dei/tutti/nel/
35	humanities/postgraduate/sociology/consultation/acronym/
36	dogs/breed/puppy/breeder/groom/
37	educator/secondary/achievement/literacy/learner/
38	otro/nuevo/sus/noticia/inicio/
39	photographer/catalogue/bibliography/curator/photographic/
40	intel/sql/cpu/mysql/programmer/
41	pussy/babe/hardcore/lesbian/naked/
42	prescription/dose/fda/generic/mg/
43	herbal/remedy/stimulate/naturally/habit/
44	machinery/leads/textile/exporter/plastics/
45	fellowship/proposal/completion/thesis/contribution/
46	commune/sainte/michel/mont/boi/

Table A.2: The Shard Profiles of topical shards 24-46 of CatgoryB dataset.

SID	Top 5 Terms
47	node/proceedings/tackle/symposium/rod/
48	alien/bubble/mahjong/solitaire/racer/
49	moderator/phpbb/prev/unanswered/signature/
50	electoral/voter/legislature/parliamentary/incumbent/
51	hostel/airfare/depart/edinburgh/dublin/
52	seo/carbon/emission/ranking/forecast/
53	bouquet/sympathy/congratulations/romance/arrangement/
54	gamble/bonus/slot/dairy/blackjack/
55	bb/linkback/cfm/gmt/vbulletin/
56	follower/sidebar/blogspot/blogg/atom/
57	autre/jour/sont/lien/cette/
58	ecommerce/designing/405/231/413
59	missile/weblog/uncategorized/meta/blogroll/
60	certify/regulatory/accident/assess/reduction/
61	glitter/outreach/blackboard/webmail/shtml/
62	astrology/mug/merchandise/ups/coin/
63	stumbleupon/stumble/buttons/appearances/medal/
64	collected/disclose/personally/identifiable/confidential/
65	nba/nascar/nhl/boxing/rugby/
66	cid/skydrive/footwear/sock/isn/
67	commander/corps/1945/chronology/headquarters
68	liable/damages/imply/arise/limitation/
69	obituary/inaugurate/columnist/ap/digg/

Table A.3: The Shard Profiles of topical shards 47-69 of CatgoryB dataset.

SID	Top 5 Terms
70	andrew/daniel/arthur/walter/patrick/
71	pubme/activate/induce/receptor/cellular/
72	zu/mit/ist/auf/den/
73	sidebar/blogspot/atom/blogger/donation/
74	usersearch/odp/disney/mujere/cu/
75	bid/memorabilia/mem/paypal/acceptance/
76	00pm/30pm/venue/fri/wed
77	treasurer/agenda/vice/chairman/advisory/
78	protocol/configure/ethernet/cisco/modem/
79	duke/queen/crown/medieval/pope/
80	transcript/veteran/counselor/tuition/senate/
81	recycle/clerk/agenda/-9/inspection
82	jose/canyon/sacramento/oakland/diego/
83	mccain/reggae/clinton/sonnery/twitter/
84	contactu/2010/plaza/eur/usd
85	draw/uefa/talent/broadway/fixture/
86	meetup/hq/inappropriate/organized/playgroup/
87	joke/freeware/kb/gadget/shareware/
88	bc/rk/bahasa/conquer/emperor/
89	caico/pitcairn/mariana/norfolk/principe/
90	workout/martial/gym/strength/hr/
91	rio/seu/uma/pt/jogo/
92	canvas/painter/brush/wooden/quilt/

Table A.4: The Shard Profiles of topical shards 70-92 of CatgoryB dataset.

Bibliography

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

[2] Leo Breiman, *Random forests*, Machine learning **45** (2001), no. 1, 5–32.

[3] Christopher JC Burges, *From ranknet to lambdarank to lambdamart: An overview*, Learning **11** (2010), no. 23-581, 81.

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

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

[6] Jamie Callan, *Distributed information retrieval*, Advances in information retrieval, Springer, 2002, pp. 127–150.

[7] Jamie Callan and Margaret Connell, *Query-based sampling of text databases*, ACM Transactions on Information Systems (TOIS) **19** (2001), no. 2, 97–130.

[8] Bruce Croft and Jamie Callan, *Lemur project*, 2000.

[9] Zhuyun Dai, Yubin Kim, and Jamie Callan, *Learning to rank resources*, Proceedings of the SIGIR Conference, ACM, 2017, pp. 837–840.

[10] Van Dang, *Lemur project components: Ranklib*, 2013.

[11] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer, *An efficient boosting algorithm for combining preferences*, vol. 4, 2003, pp. 933–969.

[12] Luis Gravano and Hector Garcia-Molina, *Generalizing gloss to vector-space databases and broker hierarchies*, Tech. report, Stanford InfoLab, 1999.

[13] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic, *The effectiveness of gioss for the text database discovery problem*, ACM SIGMOD Record, vol. 23, ACM, 1994, pp. 126–137.

[14] Luis Gravano, Héctor García-Molina, and Anthony Tomasic, *Gloss: text-source discovery over the internet*, ACM Transactions on Database Systems (TODS) **24** (1999), no. 2, 229–264.

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

[16] Anagha Kulkarni, *Shrkc: Shard rank cutoff prediction for selective search*, International Symposium on String Processing and Information Retrieval, Springer, 2015, pp. 337–349.

[17] Anagha Kulkarni and Jamie Callan, *Selective search: Efficient and effective search of large textual collections*, ACM Transactions on Information Systems (TOIS) **33** (2015), no. 4, 17.

[18] Anagha Kulkarni and Ted Pedersen, *How many different" john smiths", and who are they?*, AAAI, 2006, pp. 1885–1886.

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

[20] Tie-Yan Liu et al., *Learning to rank for information retrieval*, Foundations and Trends® in Information Retrieval **3** (2009), no. 3, 225–331.

[21] Craig Macdonald, Nicola Tonellotto, and Iadh Ounis, *Learning to predict response times for online query scheduling*, Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, ACM, 2012, pp. 621–630.

[22] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al., *Introduction to information retrieval*, vol. 1, Cambridge university press Cambridge, 2008.

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

[24] Milad Shokouhi, Luo Si, et al., *Federated search*, Foundations and Trends® in Information Retrieval **5** (2011), no. 1, 1–102.

[25] Luo Si and Jamie Callan, *Unified utility maximization framework for resource selection*, Proceedings of the CIKM Conference, ACM, 2004, pp. 32–41.

[26] Luo Si and Jamie Callan, *Relevant document distribution estimation method for resource selection*, Proceedings of the SIGIR Conference, ACM, 2003, pp. 298–305.

[27] Luo Si and Jamie Callan, *Modeling search engine effectiveness for federated search*, Proceedings of the SIGIR conference, ACM, 2005, pp. 83–90.

[28] Luo Si and Jamie Callan, *Using sampled data and regression to merge search engine results*, Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2002, pp. 19–26.

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

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

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

[32] Jun Xu and Hang Li, *Adarank: a boosting algorithm for information retrieval*, Proceedings of the SIGIR Conference, ACM, 2007, pp. 391–398.