UTILIZING QUERY PERFORMANCE PREDICTORS FOR EARLY TERMINATION IN META-SEARCH

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMRE ŞENER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2016

Approval of the thesis:

UTILIZING QUERY PERFORMANCE PREDICTORS FOR EARLY TERMINATION IN META-SEARCH

submitted by **EMRE ŞENER** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. İsmail Sengör Altıngövde _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu _____
Computer Engineering Department, METU

Assoc. Prof. Dr. İsmail Sengör Altıngövde _____
Computer Engineering Department, METU

Assoc. Prof. Dr. Pınar Karagöz _____
Computer Engineering Department, METU

Assist. Prof. Dr. Selim Temizer _____
Computer Engineering Department, METU

Assist. Prof. Dr. Engin Demir _____
Computer Engineering Department, UTAA

Date: 22.08.2016

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Emre ŞENER

Signature:

ABSTRACT

UTILIZING QUERY PERFORMANCE PREDICTORS FOR EARLY TERMINATION IN META-SEARCH

Şener, Emre

M.S., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. İsmail Sengör Altıngövde

August 2016, 153 pages

In the context of web, a meta-search engine is a system that forwards an incoming user query to all the component search engines (aka, resources); and then merges the retrieved results. Given that hundreds of such resources may exist, it is mandatory for a meta-search engine to avoid forwarding a query to all available resources, but rather focus on a subset of them. In this thesis, we first introduce a novel incremental query forwarding strategy for meta-search. More specifically, given a ranked list of N search engines, our strategy operates in rounds, such that in each round, we retrieve the results of the next k "unvisited" resources in the list (where $k<N$), asses the quality of the intermediate merged list, and stop if any further quality improvement seems unlikely. As our second contribution, we introduce a novel incremental query result merging strategy. In this strategy, we forward query to all search engines but we assess the quality of intermediate merged lists as early as we retrieve the results from an engine and stop if any further quality improvements are not likely. In order to assess the result quality, we utilize post-retrieval query performance prediction (QPP) techniques. Our experiments using the standard FedWeb 2013 dataset reveal that the proposed strategies can reduce the response

time and/or network bandwidth usage, while the quality of the result is comparable to, or sometimes, even better than the baseline strategy.

Keywords: Meta-search Engines, Query Performance Prediction, Evaluation

ÖZ

META-ARAMA İÇİN SORGU PERFORMANS TAHMİNİ YÖNTEMLERİYLE ERKEN SONUÇ OLUŞTURMA

Şener, Emre

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. İsmail Sengör Altıngövde

Ağustos 2016, 153 sayfa

Meta-arama motorları diğer kaynaklardan gelen sorgu sonuçlarını kullanarak sonuç listesi oluştururlar. Bu tez ile meta-arama motorlarındaki sorgu maliyetini düşürmeyi hedeflemekteyiz. Bu amaçla, ilk olarak seçilen arama kaynaklarının hepsine birden sorgunun yönlendirilmesi yerine, artırımlı olarak daha küçük *k* elemanlı alt kümelere sorgunun yönlendirilmesini öneriyoruz. Artırımlı olarak yönlendirilen alt kümedeki kaynaklardan gelen sonuçlara sorgu performans tahmini adımını ekleyerek, sorgu performans tahmini değerine göre cevap kalitesini artırmayacağına karar verdiğimiz durumlarda kalan kaynaklara sorgunun gönderilmemesi ile sorgu maliyetinin azaltılmasını hedefliyoruz. Önerdiğimiz ikinci yöntemde ise, sorgu seçilen kaynakların hepsine birden yönlendiriliyor ancak, herhangi bir kaynaktan sonuç gelir gelmez ara listeleri birleştirip, sorgu performans tahmini ile daha sonraki kaynaklardan gelecek cevapların sonuç kalitesini artırmayacağına karar verdiğimiz durumlarda, bu ara listeyi kullanıcıya dönerek sorgu zamanını azaltmayı hedefliyoruz. Yöntemlerimizi FedWeb 2013 verisi üstünde test ettik ve deney sonuçlarından artırımlı sorgu işleme yöntemlerimizin tüm seçilen kaynaklara tek seferde göndermeye göre sorgu maliyetini ve/veya ağ kaynakları kullanımını düşürdüğünü gördük.

Anahtar Kelimeler: Meta-arama Motorları, Sorgu Performans Ölçümü, Performans Değerlendirmesi

To My Precious Wife

ACKNOWLEDGEMENTS

I would like to enounce my deep gratitude to my supervisor Assoc. Prof Dr. İsmail Sengör Altıngövde for his valuable supervision, advice, useful critics and discussions throughout this study.

I am also grateful to my thesis committee members for their criticism and advices.

This research project is jointly supported by Ministry of Science, Industry and Technology of Turkey and Huawei Telekomünikasyon Ltd. Co. under SAN-TEZ funding program with the project number 0441.STZ.2013-2.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVATIONS

QBS	Query-based Sampling
FedWeb	Federated Web Search
SRS	Sample and Resample
CVV	Cue-validty Variance
ReDDE	Relevant Document Distribution Estimation
UUM	Unified Utility Maximization Framework
Rank-S	Rank SHIRE
SHIRE	Sampling Based Hierarchical Relevance Estimation
CSUMG	CombSUM with Global IDF
CSUML	CombSUM with Local IDF
RRF	Reciprocal Rank Fusion
ISR	Inverse Square Rank
SP	Stopping Policy

CHAPTER 1

INTRODUCTION

Keyword-based search is the most popular way of finding information on the internet. In 2014, Google alone has received two trillion queries which makes about six billion queries per day. When initial search engines emerged, the information need of the internet searchers was textual information and the results of these engines were mainly textual web pages. Along with the technology improvement and cheap hardware, the textual internet sites on the web are augmented with visual data such as images and videos. Nowadays, many commercial search engine companies such as Google, Bing, Yahoo and Yandex provide search interfaces for both textual and visual data on any digital device that has access to internet such as computers, cell phones, tablets and televisions. The most popular commercial search engines return different results for the same search query as the indexed content and coverage of the web search engines are quite diverse. The overlap of search indexes between Google and Yahoo is found to be less than 45% [1]. The diversity of search results of most used commercial search engines shows that the search result for a specific query might be still improved by using federated search techniques. In the context of web search, federated search systems, so-called meta-search engines, forward the query to other web search engines (web resources) and merge the retrieved results.

There are three challenges for a meta-search engine to form the final results. The challenges are the resource representation, resource selection and result merging. The resources are typically represented using the sample index downloaded from resources. Related work section reviews other resource representation solutions in the literature. Meta-search engines use the web resources that focus on special area and content, in addition to search engines that index general web content. Selecting appropriate web resources relevant to a query is important to increase the

performance of the results. Once a meta-search engine selects which resources to query, the query is sent to each of them and results are retrieved. Meta-search engines usually retrieve the result snippets rather than the real result documents, which makes the result merging step challenging because the information about a result page on snippets is much less compared to the information on the actual result page [2]. Other than the effectiveness issues, a meta-search engine encounters response time and network usage challenges. For a meta-search engine, the time for answering a query depends on response time of other web resources; and furthermore it takes more network transfer time (and bandwidth) to forward the query and retrieve the results compared to search engines that have a centralized index.

In this thesis, firstly we present a novel incremental query forwarding approach with query performance prediction at each iteration rather than forwarding the query to all selected web resources at once. By incremental query forwarding, we aim to reduce the query response time as well as the network bandwidth usage of a meta-search engine by stopping the query forwarding to more web resources if the query performance prediction on intermediate merge lists indicates that further results are not likely to increase the result quality. Secondly we introduce an incremental query result merging approach for meta-search engines. In this approach, similar to general meta-search engines, we forward query to all selected resources but terminate earlier than waiting the response of the all resources if query performance predictors applied on intermediate merge lists indicates that further resources will not bring any more quality improvements. In order to show that our proposed approaches reduce the query response time and/or network bandwidth usage, we evaluated the strategies using the Federated Web (FedWeb) Search Track 2013 Dataset [30] with various resource selection, result merging and query performance prediction methods from the literature. Our results reveal that the proposed strategies can reduce the query processing cost and/or bandwidth usage in comparison to the baseline strategy (i.e., forwarding the query to all the resources and waiting them all), while the quality of the result is comparable to, or sometimes, even better than that of the baseline.

The rest of this thesis is organized as follows. Chapter 2 reviews the related work on federated search engines. Chapter 3 presents the details of our incremental query forwarding and result merging approaches, as well as describing our meta-search engine architecture and components adopted from the literature. In Chapter 4, we describe the dataset and experimental setup, and report the results of the experiments. Finally, Chapter 5 concludes the thesis and suggests possible research directions for future work.

CHAPTER 2

RELATED WORK

Meta-search engines provide single search interface for users. The query received from users are forwarded to a set of available search engines and results of different search engines are merged into a single final list. The software architecture of a meta-search engine has been published on many previous works [3] [4] [5] [6] [7] [8] [9] [10]. Compared to general purpose web search engines, meta-search engines do not have an inverted index of vocabulary terms. Instead, a meta-search engine contains a broker which forwards the query to other web search engines in parallel and waits for the results from other search engines. When it receives the query results from other search engines, it merges the results and prepares a single final list and returns the final list to its users. Data flow in meta-search engines is shown in Figure 2.1. Broker receives the query from user and then it typically uses a *sample* index of web search engines to select relevant search engines. Broker forwards the query to selected web resources and gathers search results. Finally, the retrieved query result lists are merged into single result list and returned to the user. Activity diagram of the general meta-search engines is given in Figure 2.2. Meta-search engines encounter with three information retrieval problems, which are resource representation, resource selection and result merging [3].

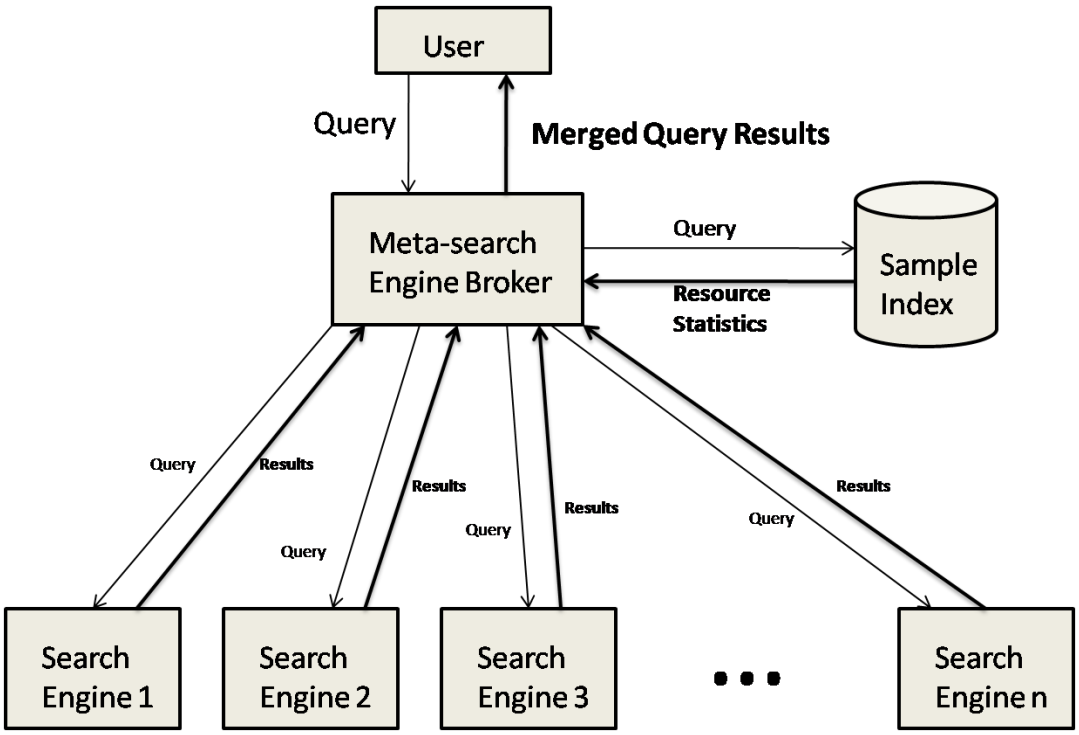

Figure 2.1 Meta-search Engine Architecture

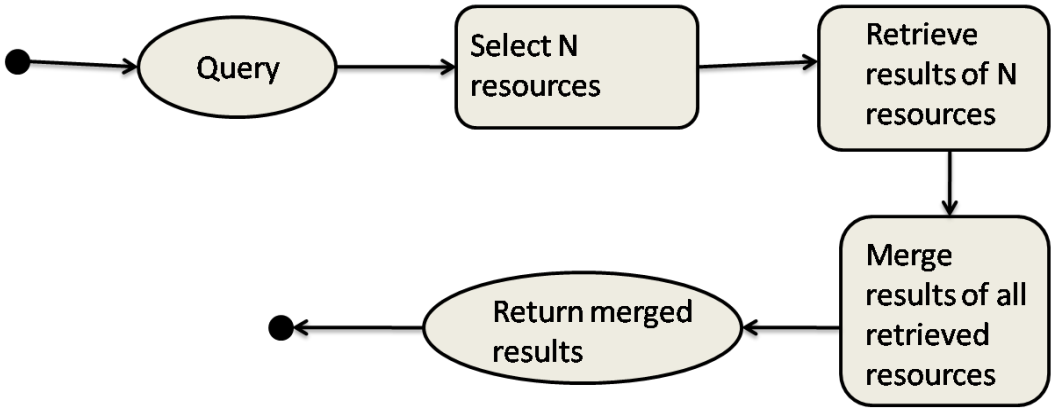

Figure 2.2 Activity Diagram of Meta-search Engines

Meta-search broker needs to select available web search engines. Selecting appropriate search engines for the query requires meta-search engine to have knowledge on available search engines. In cooperative federated search environment, the search engines might provide the collection statistics to the meta-search engine.

Federated search performance on a cooperative search environment has been studied in detail in previous works [11] [12] [13][14][15]. For meta-search engine case, it is uncooperative search environment as the component search engines do not provide information on their internal structure and statistic information on their central index. In uncooperative search environments, meta-search engines obtain approximate term statistics by gathering sample data of the component web resources. Callan and Connell proposed QBS (query-based sampling) method to download sample documents from such web resources [16]. QBS method starts with an initial single term query which would return many documents. Other queries are formed from words of the returned documents and querying is performed until enough number of documents are downloaded to form a sample data collection of the web resource. Shokouhi et al. showed that gathering more documents can increase the performance of resource selection for large collections and using the rate of unseen document terms, the authors chose the sample document size by adaptive selection [17]. Baillie et al. used the decrement of the number of unseen vocabulary terms as the condition for termination of query sampling [18]. The query sampling method usually misses the documents that have infrequent terms thus the sample document collection lacks the statistic for infrequent vocabulary terms. Using the idea of that the web resources within similar content category share similar vocabulary statistic, Ipeirotis and Gravano proposed the shrinkage method to improve the sample data with statics of other web resources in similar category [19]. For better resource selection and result merging, estimating the size of web resources is needed in addition to the term frequency information on sample document collections. To estimate collection size for federated search, Liu et al. proposed using a variant of capture history method which was commonly used method in ecology to estimate population of species in an area [20]. Capture-recapture method is based on the number of common documents on two different sample document sets which are sampled from the web resources. Alternative to variations of capture-recapture method, if web resources share term frequencies for query, sample-resample (SRS) method estimates web resource size by comparing document-term frequencies of sampled documents with the document-term frequencies of web resource [21]. The statistics on sample document collection

and estimated size of web resources can be used as resource representation for resource selection and result merging steps of federated search engines.

Meta-search engines typically process the sample data and prepare the statistics for resource representation offline. On the runtime environment of a meta-search engine, the first step is the resource selection according to the query. Using the statistics on sample data collection of web resources, meta-search engines compute the relevance between the web resources and search query and Rank-S the web resources according to relevance. Usually, only a subset of web resources are selected and query is only forwarded to these selected resources in order to decrease the network bandwidth usage [3]. In cooperative environments where web resources share their collection size and term frequencies , GIOSS algorithm is proposed for collection selection by estimating the number of documents related to the query using collection size and term frequency information [22]. For uncooperative environments, CORI resource selection algorithm Rank-S resources using Bayesian inference network and terms normalized by OKAPI term frequency normalization [23]. Document-surrogate methods performs better on uncooperative environments compared to lexicon based methods (such as GIOSS and CORI) that represent resources as a single bag of words and computes similarity between the query and the resulting big document model. Document-surrogate resource selection methods additionally use the ranking of sample documents in sample document collection. The relevant document distribution estimation (ReDDE) collection selection method estimates the distribution of relevant documents on all web resources and Rank-S the resources accordingly [24]. UUM, CRSC and SUSHI methods are proposed as variants of ReDDE algorithm [25] [26] [27]. Alternative to document-surrogate methods, the classification and clustering of web resources according to query can be used to select resources. Arguello et al. proposed classification based method for resource selection [28].

Meta-search engines receive query results as snippets and they need to merge them into a final list in relevance order. Result merging is the final step on general meta-search architectural design. In order to rank the results from different resources,

meta-search engine may assign scores to results. Such scores across different resources should be normalized to be comparable with each other. CORI result merging method uses the CORI collection selection scores in normalization of document scores [23]. Fox and Shaw proposed CombSUM, CombMax and CombMin methods to merge documents from different collections [31]. Document scores should be normalized in order to use one of these combination methods. MinMax, SUM and Virtual normalization techniques can be used to normalize document scores [33] [34]. Result merging algorithms implemented for this thesis are described in more detail in Chapter 3.

The closest work to ours in the literature is by Dreilenger et al. that proposes an incremental query forwarding approach to reduce query processing load of their meta-search engine, SavvySearch [48]. In their approach, the most relevant subset of web search engines is queried and the result page is formed by the merged results of them. The result page also contains search plan section where the rest of the engine subsets are listed. The decision on forwarding query to subsequent subset of web search engines is taken by the user instead of meta-search engine itself. One of the subsets in search plan list is selected by the user and user query is forwarded to selected subset and previous subsets of selected search plan, thus user also decides incremental step. In our approach, we automatically decide whether to continue using QPP scores of intermediate merge list with ad-hoc stop policies and ML models.

Query performance prediction can be done both before and after retrieving search results. In this thesis, we focus on query performance prediction after retrieving results. Cronen-Townsend and Croft propose clarity method to measure the coherence between query and collection [35]. More distinction of the terms in the retrieved document with the terms in the collection is accepted as better results. Clarity measures the KL divergence between the language model of retrieved documents and language model of collection. Variation of clarity method has been published as Divergence From Randomness by Amati et al. [36]. Alternatively, distribution of scores of retrieved documents can be used to predict query

performance [37] [38] [39] [40] [41] [42] [43] [44] [45] [46]. Raiber and Kurland used Markov Random Fields for learning to rank queries [47].

CHAPTER 3

UTILIZING QUERY PERFORMANCE PREDICTORS FOR EARLY TERMINATION IN META-SEARCH

A meta-search engine forwards the user query to the other search engines and gathers results from them. The processing time to gather these results and network bandwidth usage are main costs of meta-search engines. The limit of the access count on web resources is another cost and reaching the usage limit might make the web resource unavailable for following queries. Typically, when the final merged results are prepared, none of the results of some engines appear on the final list, so the time and bandwidth usage for forwarding query to those engines are wasted. Decreasing the number of search engines that the query is forwarded may decrease the response time and the network bandwidth spent.

3.1. QPP Based Adaptive Incremental Query Forwarding in Meta-Search

In a typical meta-search engine, a subset of N web resources that are decided to be more relevant to query are selected and the query is forwarded to all of the selected resources at once. We propose incremental query forwarding to reduce the meta-search engine costs. Query is forwarded to k number of resources which is a smaller subset of selected resources. The results of k number of resources are retrieved and merged into a single intermediate list. The query performance prediction method is applied on this merged list and if predicted performance of the query results shows that receiving more results from rest of the resources would not increase the quality of the final list, the query forwarding is terminated; otherwise the query is forwarded to the next k resources. Activity diagram of incremental query forwarding approach is shown in Figure 3.1 and we provide pseudocode for our approach in Table 3.1.

In the pseudocode of the algorithm rankResources() and mergeResults() methods can be implemented using the state-of-art methods described in Chapter 2. In Sections 3.3 and 3.4 we describe resource selection and result merging methods we employ in this thesis. The novelty of our approach is automatically deciding whether to stop or access more resources using current merged list of retrieved results. In order to decide stopping, we use the query performance prediction methods in StopCond() method. In Section 3.5, we describe SUM based and NDCG based query performance prediction methods that we implemented in this thesis. In Section 3.6, we present the stopping policies constructed in adhoc manner and using machine learning based on the QPPs. In the pseudocode, it can be seen that if StopCond() method returns true, than retrieveResultsInParallel() method is not called for the remaining resources, thus we do not forward query to the rest of the resources.

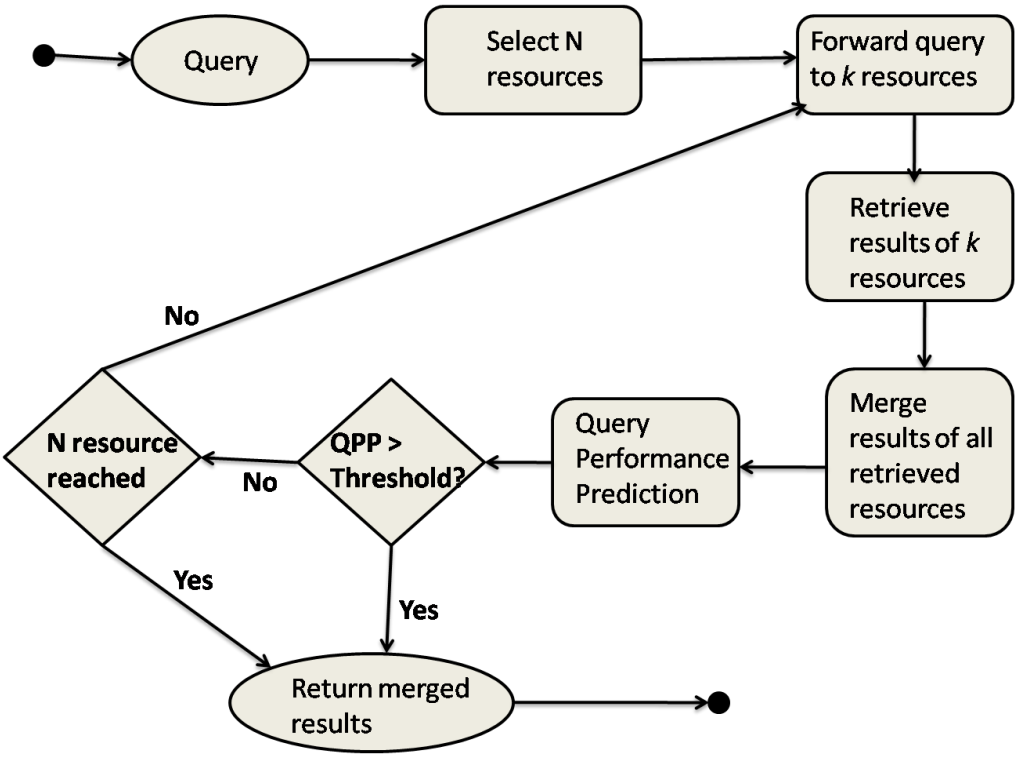

Figure 3.1 Activity Diagram of Incremental Query Forwarding

Table 3.1 Incremental Query Forwarding Algorithm

```
Algorithm Incremental Query Forwarding
Input:  q: query, k: step size
Output:  m: merged result
 1  start=1;
 2  end = k;
 3  stop ← false
 4  R[] ← rankResources(q) // R[]: a ranked list of resources in descending order of
    relevance to the query
 5  r[] ←emptyArray( |R|) // Empty array of result lists
    while (end <= |R| and not stop)
 6     // retrieve results from k resources
 7     r[start .. end] ← retrieveResultsInParallel(R, start, end)
 8     // merge & check for stopping condition
 9     prev ← emptyArray(end) // Empty array of size end
10     for  (i=1; i<=end; i++)
11     |    m ← mergeResults(r, 1, i);
12     |    prev[i] = m;
13     |    if StopCond(q, m, prev)
14     |    |   stop ← true;
15     |    |   break;
16     start ←end +1;
17     end ← end + k;
18  return m;
```

3.2. QPP Based Adaptive Incremental Query Result Merging in Meta-Search

In this approach similar to typical meta-search engine, we forward query to all resources at once. Meta-search engines generally wait for the results from all selected resources, and merges the results after receiving all results. In our approach, as early as we retrieve results from a resource, we calculate query performance prediction (QPP) value and decide whether to return the current merged list to the user before waiting the rest of the resources. In this approach, we aim to reduce the query response time compared to waiting all resources while preserving the result quality

of the final merged list. Activiy diagram of incremental result merging is given in Figure 3.2.

In the Table 3.2, the pseudocode of the incremental query result merging algorithm is given. The method forwardQueryToResources() forwards the query to selected resources and returns a handle to check whether a resource responsed. In the while loop getNextAvailableResultList() method blocks until next resource responses and returns the result list of the earliest responded resource. The previous result lists and last result list are merged and StopCond() is checked. If StopCond() method returns true indicating that further results will not improve the query result quality, the merged list is returned, thus we do not wait for other resources to respond.

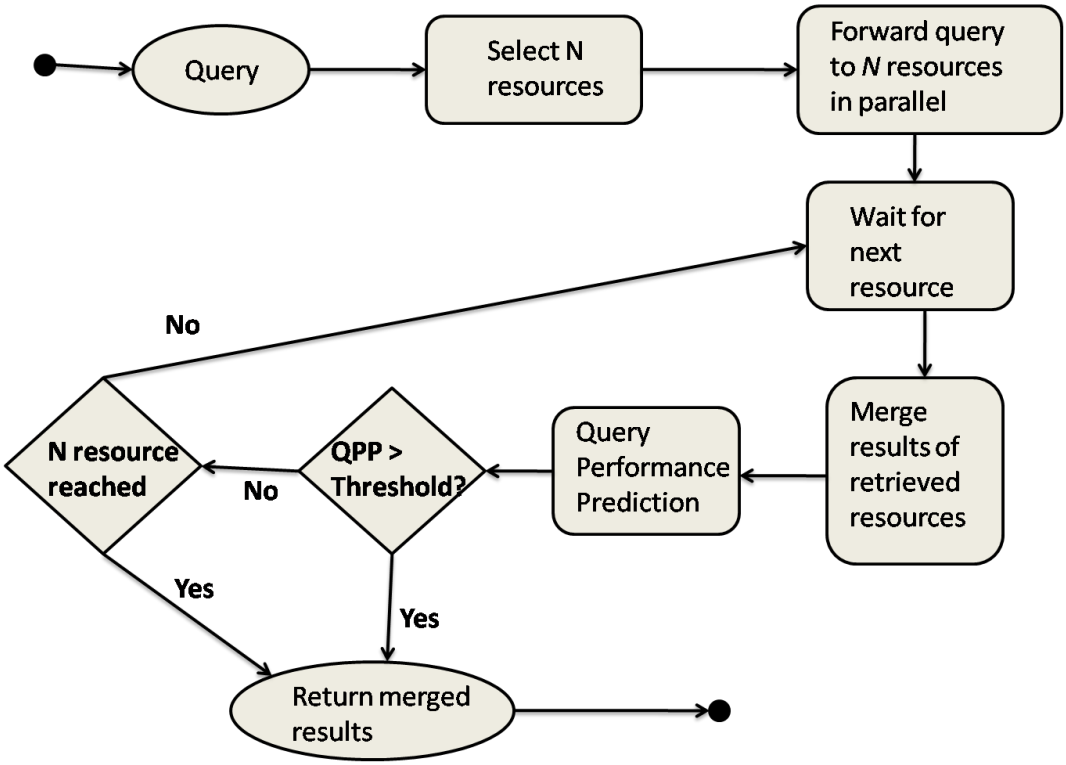

Figure 3.2 Activity Diagram of Incremental Query Result Merging

Table 3.2 Incremental Query Result Merging Algorithm

Algorithm Incremental Result Merging
Input: q: query
Output: m: merged result

```
1   stop ← false
2   counter=1;
3   R[] ← rankResources(q) // R[]: a ranked list of resources in descending order of
    relevance to the query
4   ResponseHandle = forwardQueryToResources(R) //Query is forwarded to all
    resources in parallel.
5   while (counter <= |R| and not stop)
      //Get next downloaded result list. Block until a resource responses.
6     r[counter] = getNextAvailableResultList(ResponseHandle);
7     // merge & check for stopping condition
8     prev ← emptyArray(end) // Empty array of size end
9     for  (i=1; i<=counter; i++)
10    |   m ← mergeResults(r, 1, i);
11    |   prev[i] = m;
12    |   if StopCond(q, m, prev)
13    |   |  stop ← true;
14    |   |  break;
15    counter ← counter + 1;
16  return m;
17
18
```

3.3. Resource Selection

The number of resources which will be accessed is restricted by the predefined number *N* for meta-search scenario, thus resource selection is an important step to reduce the meta-search engine costs. Resource selection includes resource ranking problem where resources are ranked according to relevance. Recently, in addition to resource ranking the cut-off prediction problem is explored in the literature. Cut-off prediction problem focuses on finding the exact number of resources that needs to be

accessed for a specific query. Our approach is orthogonal to cut-off prediction, but can be integrated with cut-off-prediction methods by incrementally accessing resources up to the predicted number. In this thesis we implemented 7 different resource selection algorithms from the literature to evaluate the performance of our proposed method. We briefly review these methods as follows:

3.3.1. TWF-IRF

TWF-IRF algorithm is the best performing algorithm in FedWeb2013 resource selection task. TWF score of a term is calculated as the sum of TF-IDF values in the sample documents for each resource, thus each engine gets different TWF score for single term [54]. IRF score is a variant of global IDF using a central sample index. The sum of TWFxIRF scores of query terms are calculated for each resource and the resources are ranked accordingly [54].

3.3.2. UISP

UiSP is the resource selection run submitted to FedWeb 2013 resource selection task by University of Stavenger. It uses central sample index to rank the sample documents and the relevance estimates of the sample documents are aggregated on resource level [30][55].

3.3.3. Oracle

For each query we calculated the average relevance score of the resources using the relevance judgement data. The resources are ranked according to the average relevance score in descending order. Obviously, the ranking of this algorithm is the ground truth for resource selection task.

3.3.4. FedWeb2013 Baseline

FedWeb 2013 Track provides baseline resource selection results to be used in result merging task [30]. The organizers manually evaluated top 3 snippets from each resource for each query and ranked resources according to partial evidence. We used the resource ranking data provided by the organizers.

3.3.5. ReDDE

Relevant Document Distribution Estimation uses a global index on sample data to rank the sample documents [24]. We calculated BM25 score using global IDF to rank sample documents. In the original ReDDE, the score contribution of a document *d* from a parent resource *R* is supposed to be Size(*R*) / SampleSize(*R*) values over the sample ranking obtained for the query. Thus, for a given *L* value, different queries may yield different number of resources and it is possible to use either all of these resources, or again select top-*N* among them. In this thesis, we report the experiments with the former, adaptive, strategy using a wide range of *L* values, namely, from 100 to 1 billion documents on a log scale. In [24], it is recommended to set *K* as the 0.3% of the sum of the collection sizes at each resource. In our setup, since the actual collection size is in the order of billions for certain resources (like Bing, Google, or YouTube), the sum of these yield a value of several billions, and hence, we experiment with a wide range of values, from 100 to 100 billion on a log scale. We see that the performance stabilizes after L = 10M and in this case ReDDE selects around 35 resources on the average for the dataset used in this thesis. (see Appendix B for the details of the threshold selection). The resources that do not bring any sample documents within the threshold are eliminated directly, which makes this algorithm an adaptive cut-off predictor according to the query.

3.3.6. Rank-S

Rank-S algorithm runs on the central sample index and computes *vote* of sample document d for its parent resource as vote(d) = score(d)*B^{-p} where B is the base of the exponential decay function and p is the rank position of the document [51]. The cut-off is determined as the rank position of d where the vote(d) gets less than 0.0001 as described in [51]. We trained B value between [1.1 , 50] range and determined 1.1 as the best performing B value to be used in our experiments (see Appendix B for the detailed experiments). Score of the resource is calculated as sum of the votes of the sample document related to query on that resource. The resources that have no voted document are eliminated.

3.3.7. Adaptive-k

Adaptive-k algorithm is a variant of ReDDE algorithm which restricts the sample documents to contain all query terms [52]. The threshold L used in ReDDE is not used any more since the documents are restricted to contain all query terms. The score of the resources are calculated as in the ReDDE.top algorithm [52][53]. The resources that do not contain a document with all query terms are eliminated.

3.4. Result Merging

The result merging step is crucial for our incremental query forwarding approach as result merging is performed again for any retrieved resource in addition to subsequent subset of retrieved resources. The result merging problem is widely explored in the literature and we briefly described the result merging methods used in meta-search setup in Chapter 2.

In this thesis, we implemented result merging methods from both rank-based and score-based categories. For rank-based category we implemented ISR [56] and RRF [29] algorithms which are reported to perform well on FedWeb 2013 dataset. For score-based category we implemented CombSUM algorithm [29][30][32]. We briefly review these methods as follows:

3.4.1. Oracle Merging

Using the relevance judgment data, the retrieved documents are sorted according to their relevance score in descending order. This algorithm is implemented to show the best possible performance for the result merging.

3.4.2. ISR

Inverse Square Rank (ISR) algorithm uses the rank of duplicated documents to calculate a score for merging. Duplicates are basically detected by having same page URL. The rank score is calculated as follows:

$$\mathrm{ISR}(d) = N(d) * \sum_{e=1}^{N(d)} \frac{1}{R(e,d)^2} \tag{3.1}$$

Where *N(d)* is the number of times a document appears on a results list (document frequency or number of duplicates for a document), and *R(e,d)* is the rank of document *d* on engine *e* [56]. The results are merged into final list in descending order of the calculated rank score ISR(*d*).

3.4.3. RRF

Similar to ISR algorithm, Reciprocal Rank Fusion (RRF) algorithm uses only the rank of the duplicated documents. The rank score is calculated as follows:

$$\mathrm{RRF}(d) = \sum_{e=1}^{N(d)} \frac{1}{k+R(e,d)} \quad (3.2)$$

Where *N(d)* is the number of times a document appears on a results list (document frequency or number of duplicates for a document), and *R(e,d)* is the rank of document *d* on engine *e* [29]. The results are merged into final list in descending order of the calculated rank score RRF(*d*).

3.4.4. CombSUM

We implemented well-known CombSUM algorithm that computes the document score of the results of retrieved resources and sums the document scores for the same document[31]. The CombSUM score is computed as:

$$\mathrm{CombSUM}(d) = \sum_{e=1}^{N(d)} \mathrm{score}(e,d) \quad (3.3)$$

Where score(*e,d*) is the document score calculated for engine *e* and *N(d)* is the number of duplicates of same document. Duplicates are detected by same page URL.

In the meta-search result merging case, only the snippets of the documents are available and the original result documents are inaccessible without downloading the document on the fly. Downloading the document requires both bandwidth and time resources thus terms on snippets are used as document, thus we only used snippets in score calculation. Snippet score is calculated as:

$$\mathrm{score}(d,Q) = \sum_{i=1}^{n} \mathrm{IDF}(q_i)\,.\,\mathrm{TF}(q_i,\mathrm{d}) \quad (3.4)$$

$$\mathrm{IDF}(q_i) = \log \frac{N-n(q_i)+0.5}{n(q_i)+0.5} \quad (3.5)$$

where *N* is the total number of documents on the downloaded sample index and $n(q_i)$ is the number of documents containing term q_i of query *Q*. $TF(q_i, D)$ is the number of occurrences of term q_i on the snippet *D*. We used sample indexes of all web resources as single sample index and calculated IDF value as the global IDF of all web resources, thus no normalization is needed after score calculation for this setup

(called CSUMG in the next chapter). We also calculated local IDF value for each resource using its own sample index and scores are normalized using MinMax normalization (called CSUML hereafter) [33] [34].

3.5. Query Performance Prediction (QPP)

Query performance prediction methods are widely used in the literature. For example; clarity-based predictors calculate the divergence between model induced from result list and resource models [35]. Some prediction methods analyze the retrieval scores of the result list as described in Chapter 2. Markovits et al. define QPP features using both the final result list and the internal result lists of resources. However, to the best of our knowledge, none of the earlier works utilize QPP techniques to assess the quality of the partially merged (i.e., intermediate) result lists.

StopCond() method in our algorithms (see Tables 3.1 and 3.2) takes three inputs; query, merged list, and previous merged lists. We calculate the quality for the final merged list and previous merged lists using SUM-based and NDCG-based approaches as described in the following.

3.5.1. SUM QPP

Using the TF-IDF equation (3.4) given in section, the score of the results in the merged list are calculated and they are summed to reflect an individual score for merged list. While calculating IDF value, we used local IDF value which is calculated on the sample index of the resource on which the snippet is retrieved.

3.5.2. NDCG QPP

To calculate the QPP of the merged result list, we used virtual NDCG@20 value of the merged result lists. If the document relevance scores are known, NDCG is calculated as follows:

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{2^{rel_i}-1}{\log_2(\mathrm{i}+1)} \tag{3.6}$$

$$\mathrm{nDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p} \tag{3.7}$$

where p is 20 for calculating NDCG@20 and rel_i is the relevance score assigned to the document [32] and IDCGp is the DCG score of the best ranking of the final results. Since the original documents and their relevance judgments are not available, we used TF-IDF score of the snippets as relevance scores in DCG formula (3.6). Murat Özdemiray defines the virtual document as that it is the document which contains only the query terms and its length is the average document length in the sample index [34]. He uses the virtual document for score normalization across different resources. Similar to his approach, we define the virtual snippet as that it is the snippet which contains only query terms and its length is the average snippet length. NDCG-based QPP score VNDCG is calculated as follows:

1. Value *v* is assigned to the TF-IDF value is calculated for the virtual snippet. (TF-IDF is calculated as equation 3.4 and local IDF is used for IDF value)

2. The TF-IDF value of the snippets in the result list is calculated.

3. Snippet score calculated in step 2 is normalized by dividing the score by *v* (virtual document score) thus each snippet acquires score between 0 and 1.

4. DCG_p is calculated as *reli* is the normalized score of the snippet in Step 3.

5. $IDCG_p$ is calculated as if there was an ideal result list contains all snippets as virtual snippet thus making *reli* = *v*/*v* = 1 for each snippet.

6. VNDCG is assigned to $DCG_p/IDCG_p$

3.6. Learning to Stop

After calculating the QPP scores described in previous section, we check the distribution of calculated QPP scores of final merged list and previous lists to see

whether the distribution matches to the stopping policy (discussed in section 3.6.1). If the distribution obeys the stopping policy conditions, then StopCond() method returns true indicating that query will not be forwarded to other resources and current final merged list will be returned to user.

In an incremental manner, all of the intermediate merged lists acquires a QPP score by SUM-based or NDCG-based query performance prediction. The question is how we can use these scores to predict whether accessing more resources will improve the quality of the final results. Basically a threshold score can be learnt over training set and used as a lower bound for QPP scores to stop, however, we believe that QPP scores are inconsistent among different queries. To resolve the inconsistency among different queries, we define adhoc stop policies and construct machine learning models based on the changes of QPP scores of intermediate merged lists.

3.6.1. Adhoc Stopping Policies

. We define three different type of adhoc stopping policies.

P1: Stop when the QPP score drops more than *t* times in *p* list-merges

P2: Stop when the QPP score drops more than *t* times in *p* consecutive list-merges

P3: Stop when the QPP score is under *s*% of the score in the previous list-merges

Table 3.3 Parameter Values

Parameter	Min	Max	Interval
t	1	3	1
p	2	20	1
s	-10	10	2

The range for *t, p, s* parameters is given in Table 3.3. All combinations of parameter values which are used P1, P2 and P3 stopping policies are trained on the training set. For each merging algorithm, we used P1, P2 and P3 policies with their range of parameters as a stopping condition and calculated NDCG@20 and P@20 score (using the relevance judgment data provided with the FedWeb 2013 dataset) on final merged list to decide when to stop. The stopping policy that leads to the highest NDCG@20 or P@20 score on the average on the training set is selected as the stopping policy for the test set. We used half of the queries with relevance judgment data to train and select the stopping policy and used other half to test the stopping policy.

3.6.2. Machine Learning Model

We build decision trees over the training query set to learn the rules for stopping. Assuming there is N resources selected for each query, we construct N-1 different models, M_i for i= 2 to N . Model M_i is trained with the features based on QPP(m_1) , QPP(m_2) , QPP(m_i) where m_i is the intermediate merged list using the resources 1 to i. We define 3 different classes to be learnt for each model M_i which are early (0), exact (1) and late (2) classes. For each query in training set, we determine the index t where the result quality is the highest. For the models M_i where $i < t$, we add a train data with QPP features calculated and assign class label as early (0). For the model M_i where $i = t$, we add a train data with class label as exact (1) and for the models M_i where $i > t$, we add a train data with class label as late (2), thus for each query in the

training set, all models acquire some training data. We used Weka J48 Decision Tree (C4.5 algorithm) implementation to train the ML models [57].

While testing a new query, assume that we received the result lists from *r* different resources. We test the extracted features of new query on the models starting from M_2 to M_r. If any model M_i where $i \leq r$ classifies new query as exact (1) or late (2), we decide to terminate and return the merged list of *r* resources, otherwise we wait for more resources and repeat testing the models.

The actual QPP values seems to be inconsistent among queries so rather than using QPP values directly as a feature, we define ML features as QPP Ratio and QPP difference.

3.6.2.1. QPP Ratio

The feature f_a value is calculated ass QPP(m_{a+1}) / QPP(m_a) for the QPP Ratio model M_i where a < i. There are *i*-1 features for model M_i. QPP(m_a) is the query performance prediction value for the merged list of *a* resources.

3.6.2.2. QPP Difference

The feature f_a value is calculated ass QPP(m_{a+1}) - QPP(m_a) for the QPP Ratio model M_i where a < i. There are *i*-1 features for model M_i. QPP(m_a) is the query performance prediction value for the merged list of *a* resources.

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1. Data Set

The training and test data used in our experiments is the FedWeb 2013 Data Collection [30]. The FedWeb 2013 Data Collection consists of both snippet and result page of search results from 157 different search engines [30]. Search engine names and URLs are given on Appendix A. Size of the dataset directories are given in Table 4.1. It includes approximately 1.9 million snippets of search snippets and documents which makes 12 thousand documents on the average for each search engine. To acquire sample documents, each search engine is queried with approximately 2000 queries and half of them is unique for all engines. Other half is formed using terms in the documents acquired in first half. Snippets are stored in XML format and sample snippet xml is given in Table 4.2.

Table 4.1 FedWeb 2013 Data Size

Directory	**Size**	**Data Type**
FW13-sample-search	952 MB	Snippets of search results in XML format
FW13-sample-docs	188 GB	Original documents that are linked from snippets
FW13-topics-search	162 MB	Snippets of topic results in XML format
FW13-topics-docs	18 GB	Original documents that are linked from snippets of topic results

Table 4.2 Snippet Xml Example

```
<samples>
<search_results>
<query id="5000">the</query>
<engine status="OK" timestamp="2013-04-04 11:23:15" name="arXiv.org"
id="FW13-e001"/>
<snippets>
<snippet>
<link cache="FW13-sample-docs/e001/5000_01.html">
    http://arxiv.org/abs/adap-org/9912005
</link>
<title>
    Title of the snippet
</title>
<description>
    Description of the snippet
</description>
<thumb cache="FW13-sample-docs/e001/5000_01_thumb.jpg">
http://uk.eonline.com/resize/175/100/thumbnails.eonline.com/p/
ice_310_highlight_b_200566_thumb.jpg
</thumb>
</snippet>
  ...
</snippets>
</search_results>
...
</samples>
```

Along with the search result data, the relevance judgment of 50 queries, manual evaluation of approximately 78 thousand of search result pages, is given [30]. Relevance judgment file format is shown in Table 4.3 First term is the query id, third term is the snippet id and last number is the relevance score of the document. The relevance scores are assigned one of the 0, 1, 3 and 7 values. In ascending order 0 is given for irrelevant documents and 7 is given for highly relevant documents. The queries with relevance judgment data are also listed in Appendix A.

Table 4.3 Relevance Judgment File Format

7001 0 FW13-e001-7001-01 3
7001 0 FW13-e001-7001-02 1
7001 0 FW13-e001-7001-03 3
7001 0 FW13-e001-7001-04 3
7001 0 FW13-e001-7001-05 3

In our experiments, we directly used the TWF-IRF and UiSP resource selection runs that are provided with the FedWeb 2013 dataset.

FedWeb 2013 Dataset does not contain information on the processing time or response time of web resources. Processing time of the web resources is estimated as the minimum length of the posting lists which are calculated on sample index of web resources. For instance; if the query "zimerman chopin ballade" has term frequencies 45, 79, 100 in resource A, the processing cost of the query for resource A is estimated as 45 in terms of posting list size.

We used Zettair software to build sample index for each web resource [58].

4.2. Experiment Setup

In the data section, the information about Fedweb 2013 data collection is given. There are relevance judgment of 50 queries for all 157 search engines. Initially, first 25 queries are used for training to learn stopping policies for merging algorithms and last 25 queries are used to evaluate the NDCG gain and estimated cost. Then training and test data are folded to use other 25 queries for training and the remaining for test. In order to test our QPP based approaches, we used one of the resource selection algorithms described in Section 3.3 to rank the resources. We used fix cut-off value as 20 engines for TWF-IRF, UiSP, Oracle and Baseline resource selection algorithms. For ReDDE, Rank-S and Adaptive-k resource ranking algorithms, we used exactly same cut-off point as an upper-bound in our approaches. The resource selection methods return the ids of the engines and we retrieve the snippets of these

resources from the dataset. We implemented Oracle, CombSUM , RRF and ISR result merging algorithms and they are used to merge the intermediate result lists as well. The merging algorithms takes the list of snippets from different resources as input and returns single merged list with 20 snippets. Using the text information on these snippets SUM QPP and NDCG QPP scores are calculated over the intermediate merge list. Finally using the QPP values of the latest merged list and previous merge lists, we apply either one of the adhoc stop policies or we extract QPP Ratio and QPP Difference features to feed into the machine-learnt models. The response time of a search engines is estimated as the minimum posting list length of the query terms in the sample index for this search engine (See Data Set section). For each of the proposed algorithm, we measure the overall cost of the query as described below.

4.2.1. QPP Based Adaptive Incremental Query Forwarding In Meta-Search Engines

The total cost of iterations is calculated as the sum of the maximum cost of the search engines in each iteration. Thus it depends on the *k* value which is the number of search engines queried at each iteration. Response time calculated as where *NI* stands for "number of iterations" in equation 4.1.

$$\text{Response Time} = \sum_{i=1}^{NI} Max(e_{1,} e_{2\,\ldots}, e_k) \qquad (4.1)$$

The network/ resource usage cost is calculated as the number of the resources to which the query is forwarded. Note that deciding an early termination will absolutely decrease the network/resource usage cost, since the rest of the resources would not be queried.

4.2.2. QPP Based Adaptive Incremental Query Result Merging In Meta-Search Engines

Meta-search engines forwards the query to resources at the same time, however the resources reply in different times. In order to simulate this setup, after resource selection algorithm returns *N* ranked resources, we sorted the calculated response time of them in ascending order and used new order as the resource ranking in our implementation. For instance, R_1, R_3, R_2 are returned as resource ranking from one of the implemented RS algorithms and according to the given test query, we sort them in terms of response time (Sample Index PLL) and form a new resource ranking as R_3, R_2, R_1. Using the cost-sorted list, we simulate the real environment of the meta-search engine. Note that if we do not use a cut-off value for resource selection, then resource selection method is unnecessary in this setup but we used 20 engines for TWF-IRF, UiSP, Oracle and Baseline RS algorithms and the cut-off value predicted by ReDDE, Rank-S and Adaptive-k algorithms. In this setup, the network cost is equal to sending the query to all selected resources since we also forward the query to all resources, however query response time might be lower. Therefore, for this method, we only report the response time that is calculated in Equation 4.1, where *k* is equal to the number of engines we decided for an early termination.

4.3. Results

In this section, the experiment results of QPP based adaptive incremental query forwarding and QPP based adaptive incremental query result merging approaches are given.

4.3.1. QPP Based Adaptive Incremental Query Forwarding Approach

This section contains the results of all possible cases of resource selection, result merging, QPP calculation and stop condition implementations to test our QPP based adaptive incremental query forwarding approach. We divided this section into two parts according to the cut-off used in resource selection. In Section 4.3.1.1, we give the results for RS algorithms, namely TWF-IRF, UiSP, Oracle and Baseline, that we used 20 engines as fixed cut-off value. In Section 4.3.1.2, we give the results for query adaptive resource selection algorithms ReDDE, Rank-S and Adaptive-k. For each case we compare the result quality of our approach with the result quality of sending to all selected resources, and we provide a table for comparing the response time in terms of PLL cost of our approach and response time of using all selected resources, and we provide a table for comparing the network/resource usage cost.

4.3.1.1. Fix Cut-Off Comparison

In this section we analyze the resource selection algorithms with 20 engines as fix cut-off value . Firstly, we present the results for the cases that we used NDCG@20 score for quality assessing and learning the index of the resources that brings the results with best NDCG@20 score. Next, we present the results for the cases that we used P@20 score for quality assessing and learning the best index of the resources that brings the results with best P@20 score.

4.3.1.1.1. Evaluation Results for Optimizing NDCG@20

In this section, the training of the queries is based on the best index of the merged lists with highest NDCG@20 score and the quality results are given in NDCG@20 metric. The results for applying our approach are grouped under TWF-IRF, UISP, Oracle and FedWeb2013 Baseline resource selection algorithms.

Results for TWF-IRF resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.4. We can see that using stopping policies and machine learning models, we cannot pass the quality results of sending to fix 20 engines, however "Best" column indicates that there were the indexes which performed better than fix 20 engines. At the best case the NDCG@20 score of our approach is 9% less than NDCG@20 score of using fix 20 engines, but we can decrease the network cost up to 36% if we sacrifice from the response time. At the worst case, the NDCG@20 score of our approach is 20% less than NDCG@20 score of using fix 20 engines, but we can decrease the network cost up to 50% if we sacrifice from the response time.

Table 4.4 TWF-IRF RS Algorithm NDCG QPP

	NDCG@ 20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0,719	0,758	0,769	0,829	0,660	0,384	0,829
CSUMG	0,358	0,313	0,300	0,361	0,326	0,223	0,415
CSUML	0,319	0,286	0,305	0,355	0,321	0,222	0,414
RRF	0,370	0,320	0,326	0,403	0,300	0,206	0,444
IRF	0,306	0,260	0,273	0,315	0,272	0,201	0,381

Table 4.5 and 4.6 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition

Table 4.5 TWF-IRF RS Algorithm NDCG QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3122	2567	2197	2004	1847	1807	1704	1650	1524	1230
CSUMG	3445	2969	2326	2034	2054	1945	1816	1744	1552	1230
CSUML	3231	2514	2005	1982	1973	1629	1481	1449	1286	1230
RRF	3773	2977	2345	2091	2077	1953	1822	1760	1606	1230
IRF	3545	2845	2274	2015	2004	1953	1822	1757	1606	1230

Table 4.6 TWF-IRF RS Algorithm NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	15.94	16.44	16.96	17.7	17.32	17.92	17.68	18.28	19.6	20
CSUMG	12.68	13.00	13.76	14.4	14.64	15.00	14.80	15.66	16.6	20
CSUML	12.12	12.52	13.20	13.8	14.08	14.36	14.40	15.34	16.2	20
RRF	13.22	13.64	14.40	14.9	15.20	15.62	15.52	15.92	17.0	20
IRF	12.32	12.68	13.44	13.7	14.40	14.88	14.88	15.48	16.2	20

Table 4.7 and 4.8 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.7 TWF-IRF RS Algorithm NDCG QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3599	2903	2280	2084	1998	1953	1823	1769	1621	1230
CSUMG	2834	2647	2198	2001	1944	1926	1787	1722	1437	1230
CSUML	2562	2077	1947	1854	1788	1759	1576	1431	1339	1230
RRF	2384	2044	1788	1701	1690	1656	1499	1453	1399	1230
IRF	2432	2054	1726	1903	1955	1797	1430	1128	1096	1230

Table 4.8 TWF-IRF RS Algorithm NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	15.14	15.64	16.88	17.4	17.2	17.48	17.44	18.04	20.0	20
CSUMG	11.56	11.88	12.80	13.5	13.84	14.28	14.64	16.24	16.4	20
CSUML	9.68	10.16	11.04	12.1	12.4	12.44	12.56	13.36	14.8	20
RRF	11.18	11.64	12.40	13.7	13.2	13.92	14.48	15.84	17.0	20
IRF	10.94	11.24	12.00	13.0	13.48	14.54	13.84	14.08	15.0	20

Table 4.9 and 4.10 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.9 TWF-IRF RS Algorithm NDCG QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3677	2949	2326	2065	2055	1953	1823	1769	1621	1230
CSUMG	2420	2123	1969	1800	1737	1630	1449	1398	1243	1230
CSUML	2475	2103	1973	1886	1724	1606	1465	1400	1113	1230
RRF	2728	2240	2113	1968	1949	1953	1795	1590	1206	1230
IRF	2292	2048	1810	1708	1714	1652	1481	1381	1114	1230

Table 4.10 TWF-IRF RS Algorithm NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	15.14	15.64	16.88	17.4	17.20	17.48	17.44	18.04	20.0	20
CSUMG	11.56	11.88	12.80	13.5	13.84	14.28	14.64	16.24	16.4	20
CSUML	9.68	10.16	11.04	12.1	12.40	12.44	12.56	13.36	14.8	20
RRF	11.18	11.64	12.40	13.7	13.20	13.92	14.48	15.84	17.0	20
IRF	10.94	11.24	12.00	13.3	13.48	14.54	13.84	14.08	15.0	20

Using SUM-QPP as the QPP method, we give the effectiveness results of TWF-IRF resource selection algorithm in Table 4.11. At the best case the NDCG@20 score of our approach is 4% less than NDCG@20 score of using fix 20 engines, but we can decrease the network cost up to 33% if we sacrifice from the response time. At the worst case, the NDCG@20 score of our approach is 21% less than NDCG@20 score of using fix 20 engines, but we can decrease the network cost up to 49% if we sacrifice from the response time.

Table 4.11 TWF-IRF RS Algorithm SUM QPP

	NDCG@ 20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.75336	0.75802	0.76872	0.82976	0.66032	0.38462	0.82976
CSUMG	0.34680	0.31342	0.30038	0.36156	0.32584	0.22352	0.41522
CSUML	0.34092	0.28634	0.30532	0.35552	0.32164	0.22200	0.41408
RRF	0.38888	0.32024	0.32662	0.40326	0.30028	0.20632	0.44382
IRF	0.31762	0.26086	0.27300	0.31504	0.27208	0.20174	0.38144

Table 4.12 and 4.13 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.12 TWF-IRF RS Algorithm SUM QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3391	2741	2290	2022	1953	1808	1704	1658	1537	1231
CSUMG	4022	3205	2380	2100	2090	1953	1823	1769	1621	1231
CSUML	4022	3205	2380	2100	2090	1953	1823	1769	1621	1231
RRF	4022	3205	2380	2100	2090	1953	1823	1769	1621	1231
IRF	4011	3198	2379	2099	2090	1953	1823	1769	1621	1231

Table 4.13 TWF-IRF RS Algorithm SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	15.94	16.44	16.96	17.7	17.32	17.92	17.68	18.28	19.6	20
CSUMG	12.68	13.00	13.76	14.4	14.64	15.00	14.80	15.66	16.6	20
CSUML	12.12	12.52	13.20	13.8	14.08	14.36	14.40	15.34	16.2	20
RRF	13.22	13.64	14.40	14.9	15.20	15.62	15.52	15.92	17.0	20
IRF	12.32	12.68	13.44	13.7	14.40	14.88	14.88	15.48	16.2	20

Table 4.14 and 4.15 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.14 TWF-IRF RS Algorithm SUM QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3325	2643	2269	2086	1994	1953	1823	1769	1621	1231
CSUMG	2743	2214	2055	1924	1818	1630	1469	1433	1335	1231
CSUML	2425	1980	1927	1712	1650	1556	1453	1339	1197	1231
RRF	2560	2082	1713	1778	1786	1532	1372	1345	1234	1231
IRF	2796	2219	1929	1932	1949	1604	1346	1304	1134	1231

Table 4.15 TWF-IRF RS Algorithm SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	15.02	15.56	16.16	17.2	17.08	18.32	16.80	18.04	20.0	20
CSUMG	9.84	10.12	10.88	12.5	11.96	12.28	12.16	13.40	14.6	20
CSUML	8.78	9.28	10.40	10.8	10.52	11.00	11.68	12.86	13.2	20
RRF	10.06	10.76	11.76	11.7	12.52	13.14	13.36	13.72	15.2	20
IRF	9.74	10.04	11.28	10.6	12.36	12.80	12.64	14.04	13.6	20

Table 4.16 and 4.17 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.16 TWF-IRF RS Algorithm SUM QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3315	2609	2262	2100	2003	1953	1823	1769	1621	1231
CSUMG	2786	2355	2127	1925	1914	1813	1702	1655	1231	1231
CSUML	2313	1981	1805	1817	1833	1731	1462	1259	1170	1231
RRF	3467	2723	2292	2096	2053	1953	1823	1767	1592	1231
IRF	2588	2130	1703	1702	1527	1497	1357	1240	1125	1231

Table 4.17 TWF-IRF RS Algorithm SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	14.96	15.48	16.48	16.8	17.52	17.72	16.80	18.00	20.0	20
CSUMG	11.42	11.72	12.80	13.0	13.96	13.72	14.64	16.24	16.0	20
CSUML	9.60	9.92	10.72	11.2	11.92	13.66	13.36	13.36	13.0	20
RRF	14.02	14.6	15.92	15.8	17.24	15.56	16.56	18.20	19.6	20
IRF	10.44	10.92	12.32	11.8	13.12	12.90	13.92	14.44	14.4	20

Results for UISP resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.18. At the best case the NDCG@20 score of our approach is 4% less than NDCG@20 score of using fix 20 engines, but we can decrease the network cost up to 45% if we sacrifice from the response time. At the worst case, the NDCG@20 score of our approach is 27% less than NDCG@20 score of using fix 20 engines, but we can decrease the network cost up to 45% while having better response time.

Table 4.18 UISP RS Algorithm NDCG QPP

	NDCG@ 20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.76164	0.78438	0.80204	0.85358	0.66254	0.41528	0.85358
CSUMG	0.33572	0.33408	0.32144	0.38584	0.33496	0.26022	0.46032
CSUML	0.35752	0.31298	0.31934	0.37720	0.32622	0.25082	0.46002
RRF	0.35366	0.29098	0.28564	0.39640	0.29316	0.22546	0.44722
IRF	0.29904	0.22982	0.25260	0.31060	0.26224	0.21764	0.38792

Table 4.19 and 4.20 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.19 UISP RS Algorithm NDCG QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3549	3371	2776	2592	2228	2441	2356	2325	2283	2123
CSUMG	4167	3509	2964	2929	2701	2605	2541	2509	2398	2123
CSUML	4299	3615	3170	3015	2922	2649	2580	2525	2407	2123
RRF	4420	3765	3189	3028	2934	2651	2584	2544	2423	2123
IRF	4517	3834	3222	3029	2943	2648	2584	2545	2425	2123

Table 4.20 UISP RS Algorithm NDCG QPP Stop Policy Network/Resource Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	15.72	16.20	17.28	17.6	17.4	17.66	17.60	17.76	19.0	20
CSUMG	12.66	13.16	14.08	14.3	14.4	14.84	15.44	15.42	16.0	20
CSUML	12.32	12.88	13.84	14.0	14.4	14.60	15.28	14.98	15.6	20
RRF	12.50	13.04	13.92	14.2	14.6	14.72	15.36	15.32	16.0	20
IRF	10.96	11.44	12.40	12.9	13.3	13.66	13.92	14.18	15.2	20

Table 4.21 and 4.22 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.21 UISP RS Algorithm NDCG QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4320	3607	3218	3030	2940	2651	2587	2547	2427	2123
CSUMG	3561	3255	2738	2649	2582	2323	2113	2065	1777	2123
CSUML	3414	2911	2677	2318	2231	2408	2310	2071	2025	2123
RRF	3691	3097	2842	2695	2598	2631	2547	1942	1805	2123
IRF	2892	2499	2276	2165	2131	2048	1945	1910	1882	2123

Table 4.22 UISP RS Algorithm NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	14.6	15.0	16.6	16.3	18.04	16.1	16.64	18.04	20.0	20
CSUMG	11.0	12.0	12.0	13.0	13.00	15.0	14.00	15.00	14.0	20
CSUML	10.1	10.7	12.0	11.6	12.76	12.9	14.08	13.72	13.2	20
RRF	11.0	11.6	13.2	12.2	14.16	14.0	15.52	14.94	14.0	20
IRF	7.1	7.6	8.4	8.6	9.24	10.3	10.88	10.98	11.6	20

Table 4.23 and 4.24 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.23 UISP RS Algorithm NDCG QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4456	3799	3223	3030	2942	2651	2587	2547	2427	2123
CSUMG	3659	3015	2909	2661	2431	2295	2215	2159	2037	2123
CSUML	3372	2890	2666	2322	2225	2211	2120	2050	2016	2123
RRF	3692	3102	2825	2832	2578	2601	2260	2080	1945	2123
IRF	2916	2500	2364	1998	1923	1913	1823	1782	1750	2123

Table 4.24 UISP RS Algorithm NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	15.58	16.00	17.28	17.6	18.04	17.96	17.28	18.04	20.0	20
CSUMG	10.96	11.44	12.72	13.2	13.20	13.30	14.48	14.58	15.4	20
CSUML	10.56	11.12	12.16	12.1	13.28	12.82	13.36	13.16	13.2	20
RRF	11.58	11.96	13.2	13.5	13.80	14.62	14.80	14.88	15.0	20
IRF	8.76	9.36	10.4	10.8	11.04	11.82	12.00	12.24	13.2	20

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.25. At the best case the NDCG@20 score of our approach is 2.5% less than NDCG@20 score of using fix 20 engines, but we can decrease the network cost up to 37% if we sacrifice from the response time. At the worst case, the NDCG@20 score of our approach is 25% less than NDCG@20 score of using fix 20 engines, but we can decrease the network cost up to 30%.

Table 4.25 UISP RS Algorithm SUM QPP

	NDCG@ 20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.7754	0.80348	0.76716	0.85358	0.66254	0.41528	0.85358
CSUMG	0.3769	0.34354	0.31930	0.38584	0.33496	0.26022	0.46032
CSUML	0.3635	0.35220	0.33270	0.37720	0.32622	0.25082	0.46002
RRF	0.3707	0.29584	0.29756	0.39640	0.29316	0.22546	0.44722
IRF	0.2994	0.25210	0.27320	0.31060	0.26224	0.21764	0.38792

Table 4.26 and 4.27 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.26 UISP RS Algorithm SUM QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3939	3447	2786	2642	2558	2501	2375	2333	2281	2123
CSUMG	4588	4056	3225	3030	2945	2651	2587	2547	2427	2123
CSUML	4588	4056	3225	3030	2945	2651	2587	2547	2427	2123
RRF	4563	3873	3224	3030	2945	2651	2587	2547	2427	2123
IRF	4563	3873	3224	3030	2945	2651	2587	2547	2427	2123

Table 4.27 UISP RS Algorithm SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	15.72	16.2	17.28	17.6	17.4	17.66	17.60	17.76	19.0	20
CSUMG	12.66	13.16	14.08	14.3	14.4	14.84	15.44	15.42	16.0	20
CSUML	12.32	12.88	13.84	14.0	14.4	14.60	15.28	14.98	15.6	20
RRF	12.50	13.04	13.92	14.2	14.6	14.72	15.36	15.32	16.0	20
IRF	10.96	11.44	12.40	12.9	13.2	13.66	13.92	14.18	15.2	20

Table 4.28 and 4.29 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.28 UISP RS Algorithm SUM QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4274	3741	3223	3030	2944	2651	2587	2547	2427	2123
CSUMG	3333	2864	2670	2344	2192	1941	1882	1830	1745	2123
CSUML	3808	3290	2737	2621	2584	2436	2270	2141	2034	2123
RRF	3650	3307	2776	2652	2606	2468	2354	2124	2035	2123
IRF	2962	2549	2301	2225	2145	2310	2230	1939	1915	2123

Table 4.29 UISP RS Algorithm SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	15.08	15.48	16.8	16.4	18.00	17.24	16.80	18.00	20.0	20
CSUMG	9.58	9.84	11.28	11.6	11.76	12.14	12.96	13.68	13.2	20
CSUML	11.64	12.00	13.12	13.3	14.24	15.02	15.12	15.92	14.8	20
RRF	11.22	11.92	13.04	13.0	14.08	14.90	15.20	14.08	14.6	20
IRF	8.36	8.88	9.68	10.1	11.28	11.02	12.32	11.88	12.2	20

Table 4.30 and 4.31 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.30 UISP RS Algorithm SUM QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4002	3563	3199	3008	2928	2651	2587	2540	2393	2123
CSUMG	2980	2603	2295	2208	2148	2191	1854	1790	1765	2123
CSUML	3029	2645	2577	2360	2312	2348	2303	2011	1967	2123
RRF	3734	3172	2749	2654	2510	2223	2121	2063	2045	2123
IRF	3723	3151	2738	2658	2481	2186	2094	2036	2011	2123

Table 4.31 UISP RS Algorithm SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	14.54	14.96	16.16	16.2	16.9	17.00	16.96	18.16	18.6	20
CSUMG	9.72	10.16	11.36	11.3	12.2	12.76	13.36	13.04	13.4	20
CSUML	10.5	11.16	12.72	12.2	12.6	13.24	14.24	14.40	14.8	20
RRF	10.72	11.36	11.92	13.1	13.6	13.54	13.60	13.30	14.4	20
IRF	10.86	11.56	12.32	12.8	14.6	13.06	12.88	13.04	14.2	20

Results for Oracle resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.32. At the best case the NDCG@20 score of our approach is 40% better than NDCG@20 score of using fix 20 engines, and we can decrease the network cost up to 75%. At the worst case, the NDCG@20 score of our approach is 20% better than NDCG@20 score of using fix 20 engines, and we can decrease the network cost up to 75%. In oracle resource selection, it can be seen that stopping earlier yields higher performance and our stopping policies and ML models learns to stop early.

Table 4.32 Oracle RS Algorithm NDCG QPP

	NDCG@ 20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.96380	0.95980	0.95706	0.99818	0.96958	0.86504	0.99818
CSUMG	0.60692	**0.63318**	0.62112	0.48576	0.55218	0.60850	0.67770
CSUML	0.61216	0.62940	**0.64444**	0.47240	0.54688	0.60622	0.68146
RRF	0.63108	**0.65372**	0.64804	0.53694	0.5848	0.63350	0.69564
IRF	0.62196	0.63216	**0.64626**	0.44336	0.5464	0.62444	0.69234

Tables 4.33 and 4.34 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.33 Oracle RS Algorithm NDCG QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3589	3651	3119	2971	3031	2835	2765	2726	2553	2538
CSUMG	1279	1141	1514	1027	1149	1237	1124	1187	1101	2538
CSUML	972	844	1252	867	943	992	990	1063	1065	2538
RRF	1451	1367	1522	1047	1069	978	948	1068	1070	2538
IRF	996	959	1438	948	981.9	938	948	1068	1070	2538

Table 4.34 Oracle RS Algorithm NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	11.1	11.68	12.64	13.1	13.36	13.76	14.32	14.74	15.6	20
CSUMG	4.44	4.88	5.60	6.5	7.08	7.56	8.32	9.18	10.2	20
CSUML	4.68	5.12	5.84	6.9	7.32	7.84	8.64	9.54	10.2	20
RRF	4.94	5.36	6.00	6.8	7.80	8.26	8.96	9.72	10.4	20
IRF	4.76	5.20	5.84	6.6	7.32	7.98	8.96	9.54	10.6	20

Tables 4.35 and 4.36 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.35 Oracle RS Algorithm NDCG QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	2591	2197	2507	2437	2557	2792	2224.0	2134	1851	2538
CSUMG	1276	1279	932	834	926	929	940.3	1063	1065	2538
CSUML	867	1204	1432	1375	1396	929	940.3	1063	1065	2538
RRF	1092	1247	944	810	926	929	940.3	1063	1065	2538
IRF	810	1196	1008	834	929	932	940.6	1063	1065	2538

Table 4.36 Oracle RS Algorithm NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	8.96	9.44	10.56	11.3	11.16	12.16	12.8	12.4	12.4	20
CSUMG	4.34	4.64	5.28	5.6	6.00	7.00	8.0	9.0	10.0	20
CSUML	4.98	5.60	6.24	7.2	7.44	7.14	8.0	9.0	10.0	20
RRF	3.98	4.48	4.96	5.0	6.00	7.00	8.0	9.0	10.0	20
IRF	4.48	4.96	5.84	5.4	6.12	7.14	8.1	9.0	10.0	20

Tables 4.37 and 4.38 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.37 Oracle RS Algorithm NDCG QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	2361	2177	2199	2633	2281	2619	2038	2081	1961	2538
CSUMG	1056	1202	1033	873	974	929	940.3	1063	1065	2538
CSUML	903	1214	1097	1005	1045	982	940.3	1063	1065	2538
RRF	1058	1238	908	877	952	935	940.3	1063	1065	2538
IRF	809	1196	975	961	930	930	940.3	1063	1065	2538

Table 4.38 Oracle RS Algorithm NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.18	9.56	10.56	11.4	11.64	12.46	12.48	12.06	12.8	20
CSUMG	4.54	5.20	6.00	5.5	6.60	7.00	8.00	9.00	10.0	20
CSUML	4.26	4.76	5.04	6.0	6.72	7.84	8.00	9.00	10.0	20
RRF	4.50	4.96	5.36	6.0	6.84	7.28	8.00	9.00	10.0	20
IRF	4.24	4.48	4.72	5.4	6.36	7.14	8.00	9.00	10.0	20

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.39. At the best case the NDCG@20 score of our approach is 36% better than NDCG@20 score of using fix 20 engines, and we can decrease the network cost up to 76%. At the worst case, the NDCG@20 score of our approach is 13% better than NDCG@20 score of using fix 20 engines, and we can decrease the network cost up to %78.

Table 4.39 Oracle RS Algorithm SUM QPP

	NDCG@ 20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.95294	0.96712	0.9569	0.99818	0.96958	0.86504	0.99818
CSUMG	0.60702	**0.63504**	0.63184	0.48576	0.55218	0.60850	0.67770
CSUML	0.60486	0.63198	**0.63448**	0.47240	0.54688	0.60622	0.68146
RRF	0.61146	**0.64586**	0.63564	0.53694	0.58480	0.63350	0.69564
IRF	0.60626	**0.64866**	0.64394	0.44336	0.54640	0.62444	0.69234

Tables 4.40 and 4.41 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.40 Oracle RS Algorithm SUM QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3483	3366	2856	2723	2779	2589	2513	2475	2316	2538
CSUMG	611	488	1189	850	949	929	940	1063	1065	2538
CSUML	611	488	1189	850	949	929	940	1063	1065	2538
RRF	1201	1247	1298	1283	1401	1347	1324	1395	1067	2538
IRF	611	488	1189	850.4	949	929	940	1063	1065	2538

Table 4.41 Oracle RS Algorithm SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	11.10	11.68	12.64	13.1	13.36	13.76	14.32	14.74	15.6	20
CSUMG	4.44	4.88	5.60	6.5	7.08	7.56	8.32	9.18	10.2	20
CSUML	4.68	5.12	5.84	6.9	7.32	7.84	8.64	9.54	10.2	20
RRF	4.94	5.36	6.00	6.8	7.80	8.26	8.96	9.72	10.4	20
IRF	4.76	5.20	5.84	6.6	7.32	7.98	8.96	9.54	10.6	20

Tables 4.42 and 4.43 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.42 Oracle RS Algorithm SUM QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	2406	2035	2419	2718	2448	2212	2165	2236	1996	2538
CSUMG	1056	1256	924	839	926	929	940.3	1063	1065	2538
CSUML	1007	1232	881	844	940	937	940.8	1063	1065	2538
RRF	1239	1265	839	814	927	930	940.4	1063	1065	2538
IRF	1261	1245	862	812	926	929	940.3	1063	1065	2538

Table 4.43 Oracle RS Algorithm SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.32	9.80	10.64	11.4	12.12	12.16	12.8	13.14	12.6	20
CSUMG	4.30	4.80	5.60	5.7	6.00	7.00	8.0	9.00	10.0	20
CSUML	4.28	4.80	5.44	6.1	6.24	7.28	8.1	9.18	10.0	20
RRF	4.18	4.36	4.64	5.3	6.12	7.14	8.1	9.00	10.0	20
IRF	4.02	4.44	4.88	5.1	6.00	7.00	8.0	9.00	10.0	20

Tables 4.44 and 4.45 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.44 Oracle RS Algorithm SUM QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1877	1665	2222	1590	1433	1771	1324	1354	1155	2538
CSUMG	1043	1261	1021	840	926	929	940	1063	1065	2538
CSUML	1013	1220	978	810	926	929	940	1063	1065	2538
RRF	893	1216	1506	1356	1411	967	969	1077	1065	2538
IRF	1293	1284	860	862	932	929	940	1063	1065	2538

Table 4.45 Oracle RS Algorithm SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	8.70	8.92	10.08	11.1	10.68	11.76	11.68	12.42	12.2	20
CSUMG	4.44	4.88	5.76	5.9	6.00	7.00	8.00	9.00	10.0	20
CSUML	4.04	4.84	5.68	5.0	6.00	7.00	8.00	9.00	10.0	20
RRF	4.90	5.48	6.32	6.6	6.96	7.42	8.32	9.18	10.0	20
IRF	4.26	4.32	4.40	5.5	6.36	7.00	8.00	9.00	10.0	20

Results for Baseline resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.46. At the best case the NDCG@20 score of our approach is 30% better than NDCG@20 score of using fix 20 engines, and we can decrease the network cost up to 70. At the worst case, the NDCG@20 score of our approach is 14% better than NDCG@20 score of using fix 20 engines, and we can decrease the network cost up to 69%.

Table 4.46 Baseline RS Algorithm NDCG QPP

	NDCG@ 20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.87716	0.91794	0.91376	0.96888	0.91242	0.80086	0.96888
CSUMG	0.49632	0.50726	**0.50846**	0.43400	0.48190	0.51268	0.57784
CSUML	0.47862	**0.50828**	0.49554	0.41202	0.46926	0.50846	0.57190
RRF	**0.55656**	0.54530	0.54504	0.49238	0.53222	0.55940	0.61980
IRF	0.53794	**0.53928**	0.53802	0.40742	0.49262	0.55086	0.60774

Tables 4.47 and 4.48 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.47 Baseline RS Algorithm NDCG QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	2170	1839	1456	1400	1377	1309	1248	1179	1138	1238
CSUMG	1019	792	895	773	741	763	763	750	740	1238
CSUML	893	757	605	684	668	640	549	545	569	1238
RRF	698	677	407	382	448	404	460	444	483	1238
IRF	804	778	667	818	670	668	611	447	484	1238

Table 4.48 Baseline RS Algorithm NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	12.92	13.4	14.32	14.7	14.72	15.28	15.52	15.98	17.0	20
CSUMG	6.34	6.88	7.76	8.1	8.76	9.48	10.08	10.44	11.2	20
CSUML	5.34	5.84	6.96	7.0	7.44	8.54	9.44	9.90	10.6	20
RRF	6.50	6.88	7.68	8.4	8.64	9.62	9.84	10.98	11.6	20
IRF	6.04	6.60	7.44	8.0	8.52	8.68	9.44	10.44	11.4	20

Tables 4.49 and 4.50 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.49 Baseline RS Algorithm NDCG QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1677	1426	1586	1583	1676	1703	1514	1184	1081	1238
CSUMG	1073	889	807	759	580	507	569	528	501	1238
CSUML	669	710	451	485	567	471	435	444	483	1238
RRF	922	854	571	486	505	479	437	444	483	1238
IRF	1067	854	679	606	504	448	435.3	444	483	1238

Table 4.50 Baseline RS Algorithm NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.98	10.44	11.44	11.8	12.84	14.24	13.68	13.14	13.2	20
CSUMG	5.56	6.24	7.52	7.0	7.32	7.98	8.96	9.36	10.2	20
CSUML	4.84	5.40	6.16	6.3	6.96	7.28	8.00	9.00	10.0	20
RRF	5.52	5.76	6.56	7.4	7.44	8.12	8.48	9.00	10.0	20
IRF	5.84	6.08	7.28	8.2	7.32	8.12	8.00	9.00	10.0	20

Tables 4.51 and 4.52 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.51 Baseline RS Algorithm NDCG QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1397	1097	1461	1487	1426	1324	1254	1034	790.5	1238
CSUMG	1104	1017	797.0	822.3	752.7	638.8	568	528	501.9	1238
CSUML	730.2	789.2	731.6	514.8	485.4	484.2	555	444	483.5	1238
RRF	965.9	832.7	667.5	488.7	470.3	477.5	499	444	483.5	1238
IRF	895.8	826.6	658.7	476.7	490	446.7	435	444	483.5	1238

Table 4.52 Baseline RS Algorithm NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.66	10.2	11.52	11.2	12.36	12.88	14.4	14.04	12.4	20
CSUMG	5.78	6.16	7.28	7.6	7.44	7.98	8.80	9.36	10.2	20
CSUML	4.60	5.28	6.48	6.0	6.12	7.14	8.16	9.00	10.0	20
RRF	5.38	5.64	6.48	7.0	7.20	8.12	8.32	9.00	10.0	20
IRF	5.44	5.76	6.64	6.9	7.32	8.26	8.00	9.00	10.0	20

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.53. At the best case the NDCG@20 score of our approach is 23% better than NDCG@20 score of using fix 20 engines, but we can decrease the network cost up to 72%. At the worst case, the NDCG@20 score of our approach is 3% better than NDCG@20 score of using fix 20 engines, and we can decrease the network cost up to 68%.

Table 4.53 Baseline RS Algorithm SUM QPP

	NDCG@ 20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.89852	0.91902	0.905	0.96888	0.91242	0.80086	0.96888
CSUMG	0.47346	0.50184	**0.50422**	0.43400	0.4819	0.51268	0.57784
CSUML	0.46278	**0.50566**	0.50464	0.41202	0.46926	0.50846	0.57190
RRF	0.51042	**0.55012**	0.54318	0.49238	0.53222	0.55940	0.61980
IRF	0.50144	0.54232	**0.54548**	0.40742	0.49262	0.55086	0.60774

Tables 4.54 and 4.55 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.54 Baseline RS Algorithm SUM QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	2278	1792	1310	1539	1224	1130	1093	1045	971.8	1238
CSUMG	502.3	496.9	480.2	400.5	513.9	485.1	553.8	549.9	559.2	1238
CSUML	502.3	496.9	480.2	400.5	513.9	485.1	553.8	549.9	559.2	1238
RRF	502.3	496.9	480.2	400.5	513.9	485.1	553.8	549.9	559.2	1238
IRF	466.2	460.5	418.8	347.1	459.3	438.2	492.3	482.9	512.7	1238

Table 4.55 Baseline RS Algorithm SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	12.92	13.4	14.32	14.7	14.72	15.28	15.52	15.98	17.0	20
CSUMG	6.34	6.88	7.76	8.1	8.76	9.48	10.08	10.44	11.2	20
CSUML	5.34	5.84	6.96	7.0	7.44	8.54	9.44	9.90	10.6	20
RRF	6.50	6.88	7.68	8.4	8.64	9.62	9.84	10.98	11.6	20
IRF	6.04	6.60	7.44	8.0	8.52	8.68	9.44	10.44	11.4	20

Tables 4.56 and 4.57 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.56 Baseline RS Algorithm SUM QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1718	1425	1393	1493	1582	1667	1478	981.6	838.6	1238
CSUMG	1040	918.8	865	865.8	782	450.7	442.9	445.6	483.5	1238
CSUML	977.5	870.6	785	612.8	533	420.4	475.6	480.3	483.5	1238
RRF	1258	1025	1086	919.3	814	696.2	730.1	707	679.5	1238
IRF	990.1	880.8	814	686.7	719	394.3	435.3	444.8	483.5	1238

Table 4.57 Baseline RS Algorithm SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	10.52	11.12	12.64	12.0	12.76	14.1	16.0	13.9	13.6	20
CSUMG	6.12	6.68	8.16	8.2	7.80	7.84	8.6	9.1	10.0	20
CSUML	5.68	6.40	7.76	6.9	7.56	7.56	8.3	9.1	10.0	20
RRF	6.24	6.96	8.48	7.7	8.16	7.98	8.6	9.7	10.4	20
IRF	5.62	6.44	8.00	6.7	7.32	7.42	8.0	9.0	10.0	20

Tables 4.58 and 4.59 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.58 Baseline RS Algorithm SUM QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1402	1176	1601	1503	1390	1294	1267	1219	970.9	1238
CSUMG	819.2	871.1	615.2	636.5	634.7	566.2	452.9	460	493.5	1238
CSUML	660	645.8	530.2	305.5	353	361.3	435.3	444.8	483.5	1238
RRF	910.9	852.7	704.5	461	734.5	486.9	464.2	444.8	483.5	1238
IRF	999.4	892.8	850.2	696.4	483.5	381.3	445.2	449.5	486.2	1238

Table 4.59 Baseline RS Algorithm SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.02	9.40	10.48	11.6	10.92	11.2	12.48	13.68	13.2	20
CSUMG	6.04	6.36	7.52	8.1	7.80	8.40	8.48	9.18	10.2	20
CSUML	4.90	5.72	7.20	6.1	6.36	7.14	8.00	9.00	10.0	20
RRF	5.86	6.44	7.60	7.0	7.92	8.68	9.28	9.00	10.0	20
IRF	5.78	6.0	7.04	8.2	7.32	7.84	8.16	9.18	10.2	20

4.3.1.1.2. Evaluation Results for Optimizing P@20

In this section, the training of the queries is based on the best index of the merged lists with highest P@20 score and the effectiveness results are again reported using the P@20 metric. The results for applying our approach are reported under TWF-IRF, UISP, Oracle and FedWeb2013 Baseline resource selection algorithms.

Results for TWF-IRF resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.60. At the best case the P@20 score of our approach is 2.5% less than P@20 score of using fix 20 engines, but we can decrease the network cost up to 45% if we sacrifice from the response time. At the worst case, the P@20 score of our approach is 33% less than P@20 score of using fix 20

engines, but we can decrease the network cost up to 60% if we sacrifice from the response time.

Table 4.60 TWF-IRF Algorithm NDCG QPP

	P@20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.837	0.7028	0.7330	0.918	0.764	0.4494	0.91800
CSUMG	0.520	0.4787	0.4670	0.531	0.495	0.3494	0.59768
CSUML	0.506	0.4380	0.4881	0.525	0.494	0.3474	0.60668
RRF	0.479	0.3766	0.3341	0.500	0.401	0.3184	0.55892
IRF	0.365	0.3606	0.3581	0.395	0.375	0.3164	0.50892

Tables 4.61 and 4.62 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.61 NDCG QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3123	2567	2198	2005	1848	1808	1704	1650	1524	1231
CSUMG	3445	2969	2326	2034	2054	1945	1816	1744	1552	1231
CSUML	3198	2503	2073	2012	2058	1710	1570	1509	1326	1231
RRF	3773	2977	2345	2091	2077	1953	1822	1760	1606	1231
IRF	2194	1927	1501	1516	1553	1484	1406	1361	1334	1231

Table 4.62 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	11.46	11.96	12.8	13.4	13.4	14.06	14.16	14.78	15.8	20
CSUMG	11.18	11.68	12.56	13.0	13.52	13.74	14.16	14.60	15.6	20
CSUML	10.80	11.20	12.16	12.5	13.08	13.22	13.76	14.46	15.2	20
RRF	11.76	12.20	13.28	13.6	14.16	14.32	14.56	15.00	16.4	20
IRF	9.96	10.32	11.36	11.7	12.28	12.76	13.04	13.48	14.4	20

Tables 4.63 and 4.64 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.63 NDCG QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	2522	2087	1996	1873	1908	1809	1539	1362	1223	1231
CSUMG	2363	1952	1864	1761	1750	1720	1480	1338	1279	1231
CSUML	2452	2008	1654	1839	1790	1514	1223	1078	1064	1231
RRF	1987	1688	1466	1265	1227	1203	1154	1134	1108	1231
IRF	2166	1833	1584	1710	1381	1238	1180	1116	1062	1231

Table 4.64 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.42	9.92	10.64	11.3	12.32	12.96	12.32	12.68	12.8	20
CSUMG	9.52	10.04	10.88	11.5	12.48	12.1	12.32	13.14	13.6	20
CSUML	9.32	9.72	10.56	11.6	11.40	12.36	12.00	12.18	13.0	20
RRF	8.90	9.44	10.48	10.8	11.04	11.48	12.00	13.32	14.8	20
IRF	8.60	9.20	10.48	10.4	11.40	11.58	12.72	11.88	12.4	20

Tables 4.65 and 4.66 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.65 NDCG QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	2358	1913	1765	1850	1868	1492	1312	1192	1105	1231
CSUMG	2461	2043	1968	1773	1793	1759	1570	1510	1391	1231
CSUML	2328	1947	1523	1681	1678	1656	1497	1142	1104	1231
RRF	1832	1498	1260	1221	1144	1138	1046	1008	990.3	1231
IRF	1976	1652	1356	1360	1243	1210	1186	1061	1043	1231

Table 4.66 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.26	10.0	10.48	11.9	12.04	12.12	12.08	12.64	14.0	20
CSUMG	10.34	11.04	12.16	12.2	12.96	13.42	13.84	13.68	14.6	20
CSUML	10.28	10.8	12.08	12.1	13.00	12.76	14.00	13.48	14.0	20
RRF	7.92	8.4	9.84	9.3	10.32	10.72	11.52	12.42	12.8	20
IRF	7.72	8.2	9.44	9.1	10.20	10.80	11.52	11.34	12.2	20

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.67. At the best case the P@20 score of our approach is 2% better than P@20 score of using fix 20 engines, and we can decrease the network cost up to 50% if we sacrifice from the response time. At the worst case, the P@20 score of our approach is 27% less than P@20 score of using fix 20 engines, but we can decrease the network cost up to 44% if we sacrifice from the response time.

Table 4.67 TWF-IRF algorithm SUM QPP

	P@20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.863	0.70200	0.65834	0.918	0.764	0.44944	0.9180
CSUMG	0.518	0.43834	0.47500	0.531	0.495	0.34944	0.59768
CSUML	0.508	0.44166	0.42178	0.525	0.494	0.34744	0.60668
RRF	0.498	0.36366	0.40478	0.500	0.401	0.31844	0.55892
IRF	**0.403**	0.35244	0.35500	0.395	0.375	0.31644	0.50892

Tables 4.68 and 4.69 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.68 SUM QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3391	2741	2290	2022	1953	1808	1704	1658	1537	1231
CSUMG	4022	3205	2380	2100	2090	1953	1823	1769	1621	1231
CSUML	3986	3181	2374	2094	2085	1953	1823	1768	1620	1231
RRF	4026	3208	2380	2100	2090	1953	1823	1769	1621	1231
IRF	4011	3198	2379	2099	2090	1953	1823	1769	1621	1231

Table 4.69 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	11.46	11.96	12.80	13.4	13.40	14.06	14.16	14.78	15.8	20
CSUMG	11.18	11.68	12.56	13.0	13.52	13.74	14.16	14.60	15.6	20
CSUML	10.80	11.20	12.16	12.5	13.08	13.22	13.76	14.46	15.2	20
RRF	11.76	12.20	13.28	13.6	14.16	14.32	14.56	15.00	16.4	20
IRF	9.96	10.32	11.36	11.7	12.28	12.76	13.04	13.48	14.4	20

Tables 4.70 and 4.71 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.70 SUM QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	2473	1986	1831	1617	1584	1489	1312	1193	987.6	1231
CSUMG	2431	2058	1963	1724	1653	1624	1500	1323	1172	1231
CSUML	2344	1966	1919	1661	1577	1547	1433	1317	1177	1231
RRF	2500	2180	1868	1721	1594	1446	1338	1295	1287	1231
IRF	2321	1918	1647	1766	1669	1316	1181	1145	1113	1231

Table 4.71 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.4	9.6	10.4	11.3	11.7	13.2	12.32	13.86	12.8	20
CSUMG	9.0	9.4	10.4	11.0	11.4	11.7	12.08	12.32	11.8	20
CSUML	8.4	8.8	10.1	10.5	10.6	10.7	11.44	12.24	12.2	20
RRF	9.1	9.7	10.9	11.5	11.4	11.2	11.76	12.60	14.0	20
IRF	8.7	9.3	10.4	10.8	10.8	11.2	12.24	12.24	13.4	20

Tables 4.72 and 4.73 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.72 SUM QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	2249	1844	1623	1834	1624	1353	1165	971	973	1231
CSUMG	2596	2186	2092	1885	1942	1915	1717	1558	1210	1231
CSUML	1868	1650	1383	1223	1071	1029	990	960	935	1231
RRF	2220	2061	1794	1733	1631	1528	1417	1236	1224	1231
IRF	2099	1770	1430	1335	1205	1155	1109	1071	1032	1231

Table 4.73 SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	8.74	9.32	10.16	10.9	12.0	11.54	11.44	10.98	12.0	20
CSUMG	10.08	10.52	11.44	11.6	12.7	14.04	13.84	14.04	13.6	20
CSUML	7.68	8.24	9.12	9.4	10.1	10.32	10.48	11.38	11.8	20
RRF	10.56	11.32	12.64	12.0	13.8	12.64	13.76	14.58	16.2	20
IRF	8.16	8.56	9.52	9.6	10.2	11.04	10.88	11.96	12.4	20

Results for UISP resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.74. At the best case the P@20 score of our approach is 4% less than P@20 score of using fix 20 engines, but we can decrease the network cost up to 68% if we sacrifice from the response time. At the worst case, the P@20 score of our approach is 25% less than P@20 score of using fix 20 engines, but we can decrease the network cost up to 55% if we sacrifice from the response time.

Table 4.74 UISP algorithm NDCG QPP

	P@20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.85500	0.73200	0.751	0.922	0.785	0.5119	0.92200
CSUMG	0.49356	0.51300	0.503	0.536	0.505	0.4079	0.65002
CSUML	0.51700	0.50100	0.448	0.537	0.498	0.4019	0.65734
RRF	0.44200	0.39700	0.370	0.490	0.419	0.3619	0.59816
IRF	0.37384	0.33476	0.331	0.385	0.387	0.3599	0.53644

Tables 4.75 and 4.76 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.75 NDCG QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3549	3371	2776	2592	2228	2441	2356	2325	2283	2123
CSUMG	4167	3509	2964	2929	2701	2605	2541	2509	2398	2123
CSUML	4299	3615	3170	3015	2922	2649	2580	2525	2407	2123
RRF	4420	3765	3189	3028	2934	2651	2584	2544	2423	2123
IRF	2343	2014	1686	1577	1523	1793	1757	1728	1727	2123

Table 4.76 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	10.72	11.24	12.24	12.6	13.12	13.22	13.68	14.20	15.2	20
CSUMG	9.90	10.32	11.44	11.7	12.64	12.78	13.52	13.66	14.6	20
CSUML	10.32	10.80	11.76	12.2	12.80	13.04	13.68	13.88	15.2	20
RRF	9.82	10.28	11.36	11.7	12.20	12.74	13.44	13.82	14.4	20
IRF	8.42	8.92	10.00	10.4	11.20	11.30	11.92	12.70	13.4	20

Tables 4.77 and 4.78 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.77 NDCG QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3500	2963	2709	2626	2536	2518	2217	1785	1710	2123
CSUMG	3946	3294	3073	2955	2883	2623	2260	2157	2048	2123
CSUML	3579	3052	2770	2605	2342	2217	2045	2000	1976	2123
RRF	3361	2827	2695	2724	2435	2331	2256	1945	1910	2123
IRF	2265	1971	1831	1800	1745	2177	2008	1738	1729	2123

Table 4.78 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	8.36	8.76	9.52	10.0	11.76	12.02	11.20	10.98	11.2	20
CSUMG	10.64	11.20	12.56	12.0	13.52	14.04	14.80	13.94	13.4	20
CSUML	10.20	10.56	11.68	12.3	13.44	12.40	12.16	13.14	14.6	20
RRF	8.76	9.52	10.72	10.5	11.52	11.42	12.64	11.88	12.0	20
IRF	6.06	6.84	7.60	7.6	8.64	9.52	10.72	10.80	11.0	20

Tables 4.79 and 4.80 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.79 NDCG QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3542	3215	2750	2577	2540	2527	2454	2101	1737	2123
CSUMG	3603	2985	2687	2706	2546	2579	2226	2134	2025	2123
CSUML	3575	3110	2628	2549	2480	2313	2279	2110	2001	2123
RRF	3883	3324	2790	2672	2563	2324	2249	2203	2117	2123
IRF	2428	2205	1807	1752	1720	1998	1980	1978	1962	2123

Table 4.80 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.30	9.72	11.04	10.3	12.12	13.1	12.64	12.78	12.0	20
CSUMG	9.70	10.24	11.04	11.3	12.48	13.34	13.44	12.78	14.0	20
CSUML	8.26	8.84	9.52	10.0	11.44	10.68	11.60	12.10	11.8	20
RRF	9.06	9.40	10.40	11.6	10.68	11.98	11.84	12.60	13.6	20
IRF	5.54	6.28	6.64	7.5	8.52	8.82	8.96	10.08	11.0	20

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.81. At the best case the P@20 score of our approach is 0,1% beter than P@20 score of using fix 20 engines, and we can decrease the network cost up to 50% if we sacrifice from the response time. At the worst case, the P@20 score of our approach is 30% less than P@20 score of using fix 20 engines, but we can decrease the network cost up to 75% if we sacrifice from the response time.

Table 4.81 UISP algorithm SUM QPP

	P@20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.889	0.7319	0.770	0.922	0.785	0.5119	0.92200
CSUMG	**0.537**	0.4814	0.499	0.536	0.505	0.4079	0.65002
CSUML	0.525	0.4489	0.497	0.537	0.498	0.4019	0.65734
RRF	0.465	0.4110	0.352	0.490	0.419	0.3619	0.59816
IRF	0.379	0.3607	0.342	0.385	0.387	0.3599	0.53644

Tables 4.82 and 4.83 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.82 SUM QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4025	3539	2847	2661	2579	2639	2574	2417	2300	2123
CSUMG	4571	3886	3224	3030	2945	2651	2587	2547	2427	2123
CSUML	4571	3886	3224	3030	2945	2651	2587	2547	2427	2123
RRF	4539	4027	3208	3030	2928	2634	2570	2530	2410	2123
IRF	4588	4056	3225	3030	2945	2651	2587	2547	2427	2123

Table 4.83 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	10.72	11.24	12.24	12.6	13.12	13.22	13.68	14.20	15.2	20
CSUMG	9.90	10.32	11.44	11.7	12.64	12.78	13.52	13.66	14.6	20
CSUML	10.32	10.8	11.76	12.2	12.8	13.04	13.68	13.88	15.2	20
RRF	9.82	10.28	11.36	11.7	12.2	12.74	13.44	13.82	14.4	20
IRF	8.42	8.92	10.00	10.4	11.2	11.30	11.92	12.70	13.4	20

Tables 4.84 and 4.85 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.84 SUM QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3507	3184	2712	2638	2511	2511	2162	2040	1750	2123
CSUMG	3349	3077	2667	2610	2446	2130	2078	2009	1716	2123
CSUML	3664	3211	2676	2611	2530	2350	2185	2065	2030	2123
RRF	3086	3150	2706	2524	2169	2369	2292	2085	1801	2123
IRF	2135	1932	1789	1796	1771	2010	1948	1692	1697	2123

Table 4.85 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	8.72	9.32	10.08	10.5	11.88	11.28	12.0	12.06	12.2	20
CSUMG	8.52	9.20	9.92	10.4	11.88	10.74	12.0	11.7	11.6	20
CSUML	9.36	10.04	10.72	11.6	12.48	11.60	12.8	13.32	14.6	20
RRF	9.54	10.12	11.6	11.2	11.88	12.32	14.0	14.4	13.6	20
IRF	5.68	6.28	6.72	7.6	8.76	10.22	9.28	9.0	10.0	20

Tables 4.86 and 4.87 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.86 SUM QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3532	3179	2732	2835	2482	2376	2291	2225	1941	2123
CSUMG	3522	2950	2362	2568	2247	2295	2235	2154	2042	2123
CSUML	3709	3123	2792	2674	2598	2470	2381	1932	1826	2123
RRF	3651	3125	2674	2567	2448	2177	2097	2062	2041	2123
IRF	2802	2615	2497	2043	1753	2024	1986	1947	1948	2123

Table 4.87 SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	9.7	10.48	11.52	11.5	12.72	12.08	13.12	12.78	13.8	20
CSUMG	8.5	9.00	10.16	9.8	11.16	11.76	13.44	12.42	12.8	20
CSUML	10.4	10.76	12.08	11.8	12.72	13.38	14.80	14.40	14.4	20
RRF	8.9	9.40	10.24	10.7	10.32	11.26	12.32	12.72	13.8	20
IRF	6.6	7.44	8.96	7.8	8.52	8.96	10.08	10.26	11.4	20

Results for Oracle resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.88. At the best case the P@20 score of our approach is 41% better than P@20 score of using fix 20 engines, but we can decrease the network cost up to 90. At the worst case, the P@20 score of our approach is 31% better than P@20 score of using fix 20 engines, and we can decrease the network cost up to 85%. Since oracle resource selection selects the best resources at top, it can be seen that we learn to stop early.

Table 4.88 Oracle RS algorithm NDCG QPP

	P@20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.99000	0.90424	0.90502	0.993	0.992	0.91386	0.99300
CSUMG	0.84610	**0.86706**	0.86830	0.655	0.726	0.77132	0.88508
CSUML	0.85032	**0.86830**	**0.86830**	0.651	0.729	0.76532	0.88820
RRF	0.85458	**0.86538**	**0.86538**	0.694	0.744	0.76632	0.88612
IRF	0.85674	**0.86538**	**0.86538**	0.606	0.711	0.76432	0.88720

Tables 4.89 and 4.90 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.89 NDCG QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3057	3358	3027	2898	2945	2980	2828	2749	2612	2538
CSUMG	546.4	435.4	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
CSUML	542.8	432.9	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
RRF	531.4	414.0	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
IRF	526.2	405.5	807.7	810.5	926.5	929.7	940.3	1063	1065	2538

Table 4.90 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.44	4.88	6.16	6.5	7.32	7.70	8.64	9.54	10.2	20
CSUMG	2.98	3.16	4.72	5.5	6.48	7.42	8.16	9.00	10.0	20
CSUML	3.22	3.40	4.88	5.9	6.72	7.56	8.32	9.36	10.4	20
RRF	3.16	3.44	5.04	5.7	6.72	7.42	8.48	9.18	10.2	20
IRF	3.14	3.40	4.96	5.6	6.60	7.56	8.64	9.36	10.2	20

Tables 4.91 and 4.92 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model.

Table 4.91 NDCG QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1426	1229	1159	912.7	926.5	929.7	940.3	1063	1065	2538
CSUMG	511.8	390.7	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
CSUML	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
RRF	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
IRF	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538

Table 4.92 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.50	5.24	6.4	5.5	6	7	8	9	10	20
CSUMG	2.04	2.04	4.0	5.0	6	7	8	9	10	20
CSUML	2.00	2.00	4.0	5.0	6	7	8	9	10	20
RRF	2.00	2.00	4.0	5.0	6	7	8	9	10	20
IRF	2.00	2.00	4.0	5.0	6	7	8	9	10	20

Tables 4.93 and 4.94 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.93 NDCG QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	778.6	1134	956.0	818.6	928.3	929.7	940.3	1063	1065	2538
CSUMG	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
CSUML	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
RRF	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
IRF	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538

Table 4.94 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.42	5.12	6.16	5.6	6.12	7	8	9	10	20
CSUMG	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20
CSUML	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20
RRF	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20
IRF	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.95. At the best case the P@20 score of our approach is 42% better than P@20 score of using fix 20 engines, and we can decrease the network cost up to 90%. At the worst case, the P@20 score of our approach is 15% better than P@20 score of using fix 20 engines, and we can decrease the network cost up to 85%.

Table 4.95 Oracle algorithm SUM QPP

	P@20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.98700	0.90646	0.91184	0.993	0.992	0.91386	0.99300
CSUMG	0.82082	**0.86830**	**0.86830**	0.655	0.726	0.77132	0.88508
CSUML	0.81382	**0.86830**	**0.86036**	0.651	0.729	0.76532	0.88820
RRF	0.80990	**0.86538**	**0.86538**	0.694	0.744	0.76632	0.88612
IRF	0.79290	**0.86538**	**0.86538**	0.606	0.711	0.76432	0.88720

Tables 4.96 and 4.97 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.96 SUM QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	3034	3351	3034	2892	2917	2748	2594	2540	2411	2538
CSUMG	611	488.9	1189	850.4	949.7	929.7	940.3	1063	1065	2538
CSUML	1190	914.7	1296	1282	1386	1320	983.6	1068	1067	2538
RRF	1273	974.6	1331	1313	1424	1564	1019	1085	1083	2538
IRF	1273	974.6	1331	1313	1424	1564	1019	1085	1083	2538

Table 4.97 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.44	4.88	6.16	6.5	7.32	7.70	8.64	9.54	10.2	20
CSUMG	2.98	3.16	4.72	5.5	6.48	7.42	8.16	9.00	10.0	20
CSUML	3.22	3.40	4.88	5.9	6.72	7.56	8.32	9.36	10.4	20
RRF	3.16	3.44	5.04	5.7	6.72	7.42	8.48	9.18	10.2	20
IRF	3.14	3.40	4.96	5.6	6.60	7.56	8.64	9.36	10.2	20

Tables 4.98 and 4.99 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.98 SUM QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1114	1211	1070	839.1	926.6	929.7	940.3	1063	1065	2538
CSUMG	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
CSUML	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
RRF	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
IRF	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538

Table 4.99 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.62	5.32	6.56	5.7	6.12	7	8	9	10	20
CSUMG	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20
CSUML	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20
RRF	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20
IRF	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20

Tables 4.100 and 4.101 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.100 SUM QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1243	1273	1437	1021	926.5	929.7	940.3	1063	1065	2538
CSUMG	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
CSUML	562.5	430.2	834.8	834.3	944.2	929.7	940.3	1063	1065	2538
RRF	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538
IRF	506.1	384.8	807.7	810.5	926.5	929.7	940.3	1063	1065	2538

Table 4.101 SUM QPP ML Difference Network/Resource Usage Cost

	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.62	5.12	6.24	6.4	6.00	7	8	9	10	20
CSUMG	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20
CSUML	2.36	2.40	4.16	5.2	6.24	7	8	9	10	20
RRF	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20
IRF	2.00	2.00	4.00	5.0	6.00	7	8	9	10	20

Results for Baseline resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.102. At the best case the P@20 score of our approach is 30% better than P@20 score of using fix 20 engines, and we can decrease the network cost up to 82%. At the worst case, the P@20 score of our approach is 8% better than P@20 score of using fix 20 engines, and we can decrease the network cost up to 81%.

Table 4.102 Baseline RS algorithm NDCG QPP

	P@20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.89946	0.87310	0.86656	0.988	0.973	0.88564	0.98800
CSUMG	**0.72254**	0.70764	0.71150	0.599	0.670	0.70976	0.81070
CSUML	**0.71766**	0.71498	0.70240	0.588	0.665	0.71076	0.81034
RRF	**0.74078**	0.72762	0.72678	0.669	0.700	0.72264	0.82864
IRF	**0.73694**	0.70972	0.72766	0.558	0.668	0.72164	0.82852

Tables 4.103 and 4.104 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.103 NDCG QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1407	1138	909.6	993.6	891.7	827.6	789.1	733.1	714.4	1238
CSUMG	413.4	286.8	237.6	269.3	370.5	380.3	444.2	444.8	483.5	1238
CSUML	508.9	400.2	251.3	256.5	362.0	380.1	463.8	471.1	509.0	1238
RRF	306.7	198.4	198.7	219.9	334.1	351.2	435.3	444.8	483.5	1238
IRF	316.8	212.8	203.9	223.4	334.1	351.2	435.3	444.8	483.5	1238

Table 4.104 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	5.86	6.40	7.36	7.6	8.28	8.54	9.12	10.08	11.0	20
CSUMG	4.60	4.92	6.16	6.8	7.68	8.08	8.96	9.90	11.0	20
CSUML	4.04	4.40	5.60	6.2	6.96	7.70	8.80	9.90	10.6	20
RRF	4.78	5.12	6.16	7.0	7.68	8.36	8.96	10.08	11.0	20
IRF	4.52	4.84	60	6.7	7.32	7.98	8.96	10.08	10.8	20

Tables 4.105 and 4.106 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.105 NDCG QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	726.1	686.1	409.7	518.5	346.4	351.2	435.3	444.8	483.5	1238
CSUMG	453.7	315.2	281.6	219.9	334.1	351.2	435.3	444.8	483.5	1238
CSUML	60.04	518.5	285.0	219.9	334.1	351.2	435.3	444.8	483.5	1238
RRF	543.2	458.6	237.6	219.9	334.1	351.2	435.3	444.8	483.5	1238
IRF	649.9	650.3	333.7	257.8	343.8	351.2	435.3	444.8	483.5	1238

Table 4.106 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.82	5.12	6.00	6.3	6.36	7	8	9	10	20
CSUMG	3.46	3.92	5.84	5.0	6.00	7	8	9	10	20
CSUML	3.80	4.24	4.48	5.0	6.00	7	8	9	10	20
RRF	3.76	4.64	5.28	5.0	6.00	7	8	9	10	20
IRF	4.26	4.68	5.04	5.6	6.48	7	8	9	10	20

Tables 4.108 and 4.109 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.107 NDCG QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	634.6	641.4	487.8	302.5	342.7	355.2	435.3	444.8	483.5	1238
CSUMG	416.8	315.2	224.7	219.9	334.1	351.2	435.3	444.8	483.5	1238
CSUML	731.6	590.6	428.9	506.9	577.4	489.5	435.3	444.8	483.5	1238
RRF	491.2	396.2	282.8	226.9	334.1	351.2	435.3	444.8	483.5	1238
IRF	530.8	429.3	235.7	219.9	334.1	351.2	435.3	444.8	483.5	1238

Table 4.108 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.66	5.08	5.92	6.0	6.36	7.14	8	9	10	20
CSUMG	3.12	3.24	4.48	5.0	6.00	7.00	8	9	10	20
CSUML	4.20	4.72	4.96	5.6	6.72	7.56	8	9	10	20
RRF	3.76	4.68	5.36	5.3	6.00	7.00	8	9	10	20
IRF	3.62	4.32	4.64	5.0	6.00	7.00	8	9	10	20

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.109. At the best case the P@20 score of our approach is 29% better than P@20 score of using fix 20 engines, and we can decrease the network cost up to 78%. At the worst case, the P@20 score of our approach is 6% better than P@20 score of using fix 20 engines, and we can decrease the network cost up to 80%.

Table 4.109 Baseline algorithm SUM QPP

	P@20 Score						
	SP	ML Rat	ML Diff	Fix 20	Fix 10	Fix 5	Best
Oracle	0.9168	0.88610	0.88522	0.988	0.973	0.88564	0.98800
CSUMG	0.7113	**0.71532**	0.70488	0.599	0.670	0.70976	0.81070
CSUML	0.7123	**0.71398**	0.70654	0.588	0.665	0.71076	0.81034
RRF	**0.7346**	0.72320	0.71112	0.669	0.700	0.72264	0.82864
IRF	**0.7242**	0.72196	0.71112	0.558	0.668	0.72164	0.82852

Tables 4.110 and 4.111 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.110 SUM QPP Stop Policy PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	1847	1456	1045	1153	1022	924.9	887.5	837.7	787.2	1238
CSUMG	585.1	552.7	539.7	415.9	523.8	523.9	592.9	590.4	598.8	1238
CSUML	585.1	552.7	539.7	415.9	523.8	523.9	592.9	590.4	598.8	1238
RRF	565.7	533.9	491.7	377.8	476.8	482.2	536.3	537.1	561.2	1238
IRF	413.8	412.3	375	313.8	417.9	396	449.1	458.6	492.5	1238

Table 4.111 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	5.86	6.40	7.36	7.6	8.28	8.54	9.12	10.08	11.0	20
CSUMG	4.60	4.92	6.16	6.8	7.68	8.08	8.96	9.900	11.0	20
CSUML	4.04	4.40	5.60	6.2	6.96	7.70	8.80	9.900	10.6	20
RRF	4.78	5.12	6.16	7.0	7.68	8.36	8.96	10.08	11.0	20
IRF	4.52	4.84	6.00	6.7	7.32	7.98	8.96	10.08	10.8	20

Tables 4.112 and 4.113 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.112 SUM QPP ML Ratio PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	755.4	685.7	491.9	545.4	369.6	351.2	435.3	444.8	483.5	1238
CSUMG	418.5	315.2	230.2	219.9	334.1	351.2	435.3	444.8	483.5	1238
CSUML	495.7	429.5	198.7	219.9	334.1	351.2	435.3	444.8	483.5	1238
RRF	475.2	374.5	205.4	219.9	334.1	351.2	435.3	444.8	483.5	1238
IRF	532.4	413.2	254.6	238.7	334.1	351.2	435.3	444.8	483.5	1238

Table 4.113 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.98	5.44	6.48	6.3	6	7	8	9	10	20
CSUMG	3.22	3.44	4.88	5.0	6	7	8	9	10	20
CSUML	3.50	4.00	4.00	5.0	6	7	8	9	10	20
RRF	3.42	4.20	4.40	5.1	6	7	8	9	10	20
IRF	3.54	4.24	4.48	5.3	6	7	8	9	10	20

Tables 4.114 and 4.115 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.114 SUM QPP ML Difference PLL Cost

	PLL Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	760.6	692.5	628.5	528.9	378.7	376.8	435.3	444.8	483.5	1238
CSUMG	424.7	324.2	231.1	234.6	337	351.2	435.3	444.8	483.5	1238
CSUML	563.4	454.5	266.5	249.4	359.3	379.8	463.9	464.6	496.4	1238
RRF	514.9	417.5	266.2	228.8	334.1	351.2	435.3	444.8	483.5	1238
IRF	515.6	417.5	269.9	229.8	334.1	351.2	435.3	444.8	483.5	1238

Table 4.115 SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost									
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	All
Oracle	4.90	5.52	6.72	5.8	6.48	7.28	8.00	9.00	10	20
CSUMG	3.34	3.56	4.96	5.3	6.24	7.00	8.00	9.00	10	20
CSUML	4.08	4.84	5.36	5.4	6.36	7.42	8.48	9.54	10	20
RRF	3.82	4.72	5.44	5.5	6.00	7.00	8.00	9.00	10	20
IRF	3.80	4.68	5.36	5.6	6.00	7.00	8.00	9.00	10	20

4.3.1.2. Adaptive RS Comparison

In this section we analyze the resource selection algorithms with query adaptive cut-off values. Firstly, we present the results for the cases that we used NDCG@20 score for quality assessing and learning the index of the resources that brings the results with best NDCG@20 score. Then, we present the results for the cases that we used P@20 score for quality assessing and learning the best index of the resources that brings the results with the best P@20 score.

4.3.1.2.1. Evaluation Results for Optimizing NDCG@20

In this section, the training of the queries are based on the best index of the merged lists with highest NDCG@20 score and the quality results are given in NDCG@20 metric. . The results for applying our approach are grouped under ReDDE, Rank-S and Adaptive-k resource selection algorithms.

Results for ReDDE resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.116. At the best case the NDCG@20 score of our approach is 16% better than NDCG@20 score of sending to all selected resources, and we can decrease the network cost up to 72%. At the worst case, the NDCG@20 score of our approach is 7% less than NDCG@20 score of sending to all selected resources, but we can decrease the network cost up to 75%.

Table 4.116 ReDDE RS algorithm NDCG QPP

	NDCG@20 Score				
	SP	ML Rat	ML Diff	All	Best
Oracle	0.77294	0.81834	0.82832	0.916327	0.91632
CSUMG	0.38150	**0.41772**	0.41248	0.411694	0.50586
CSUML	0.39140	**0.41798**	0.40480	0.408429	0.50494
RRF	0.39248	0.37552	0.38826	0.436020	0.50150
IRF	0.36554	**0.38832**	0.38734	0.316306	0.48286

Tables 4.117 and 4.118 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.117 NDCG QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	4928	4138	3290	3165	2829	2610	2627	2637	2495	2293	2055
CSUMG	3202	2949	2709	2580	2118	1954	2059	2106	1981	2084	2055
CSUML	3485	3094	2695	2401	2158	1994	2101	2133	2004	2105	2055
RRF	1671	1715	1495	1423	1210	933	1159	1181	1157	2070	2055
IRF	1243	1326	1231	1169	1021	702	940	940	906	1962	2055

Table 4.118 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	24.3	24.8	25.6	26.1	26.5	26.4	26.9	27.0	27.2	30.5	34.6
CSUMG	8.44	8.92	9.92	10.2	10.7	11.1	12.0	12.8	13.6	20.5	34.6
CSUML	10.2	10.6	11.6	12.1	12.2	12.9	13.5	14.3	14.9	22.2	34.6
RRF	8.50	9.00	9.96	10.3	11.2	11.6	12.4	13.0	13.6	20.5	34.6
IRF	7.98	8.46	9.30	9.74	10.8	10.9	11.6	11.9	12.8	20.8	34.6

Tables 4.119 and 4.120 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.119 NDCG QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	2790	2467	2173	2765	2424	2400	2397	1834	1455	2007	2055
CSUMG	1087	963	1343	909	860	587	831	907	905	1963	2055
CSUML	1263	1315	1328	1488	1053	672	868	943	937	1974	2055
RRF	1237	1352	1497	2068	1565	940	952	1010	1004	1963	2055
IRF	745	1156	1235	936	530	557	819	896	896	1963	2055

Table 4.120 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	12.8	13.2	14.2	13.9	15.2	15.6	16.7	16.4	14.9	20.1	34.6
CSUMG	6.66	7.14	8.14	8.22	8.26	8.70	9.10	10.1	11.0	19.4	34.6
CSUML	7.48	8.02	8.74	8.8	9.70	9.08	9.86	10.8	11.7	19.7	34.6
RRF	6.64	7.22	7.58	8.32	9.46	8.98	9.26	10.2	11.2	19.4	34.6
IRF	5.72	5.88	6.68	7.82	6.94	7.86	8.78	9.70	10.6	19.4	34.6

Tables 4.121 and 4.122 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.121 NDCG QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	2576	2361	2318	2794	2630	2688	2544	1577	1466	2020	2055
CSUMG	1028	994	701	729	614	611	818	895	895	1963	2055
CSUML	1186	1103	1377	1578	1551	1229	1389	1232	931	1963	2055
RRF	1127	986	1592	1419	1393	1113	1303	1171	1171	1963	2055
IRF	1052	1222	1137	1172	857	813	827	896	896	1963	2055

Table 4.122 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	14.2	14.7	16.0	15.3	16.1	17.3	18.9	16.6	16.1	21.5	34.6
CSUMG	6.06	6.32	6.80	7.70	8.08	8.82	8.8	9.76	10.7	19.4	34.6
CSUML	7.14	7.62	8.42	9.00	9.34	9.84	10.4	10.3	10.9	19.4	34.6
RRF	6.86	7.50	8.46	7.82	8.98	9.26	10.1	10.2	11.2	19.4	34.6
IRF	6.34	6.72	7.32	7.82	8.50	8.70	9.1	9.7	10.6	19.4	34.6

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.123. At the best case the NDCG@20 score of our approach is 25% better than NDCG@20 score of sending to all selected resources, and we can decrease the network cost up to 84%. At the worst case, the NDCG@20 score of our approach is 14% less than NDCG@20 score of sending to all selected resources, but we can decrease the network cost up to 83%.

Table 4.123 ReDDE algorithm SUM QPP

	NDCG@20 Score				
	SP	ML Rat	ML Diff	All	Best
Oracle	0.82738	0.82306	0.80752	0.916327	0.91632
CSUMG	0.37266	0.40920	**0.41362**	0.411694	0.50586
CSUML	0.37454	**0.41890**	0.41768	0.408429	0.50494
RRF	0.42482	0.39146	0.37568	0.436020	0.50150
IRF	0.35144	**0.39430**	0.38106	0.316306	0.48286

Tables 4.124 and 4.125 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.124 SUM QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	5519	4656	3708	3563	3194	2998	2976	2989	2829	2363	2055
CSUMG	4479	3835	3107	3156	2825	2755	2751	2775	2671	2256	2055
CSUML	4479	3835	3107	3156	2825	2755	2751	2775	2671	2256	2055
RRF	5688	4790	3784	3625	3250	3087	3052	3068	2896	2381	2055
IRF	783	639	570	1026	769	749	970	1027	1027	1962	2055

Table 4.125 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	24.3	24.8	25.6	26.1	26.5	26.4	26.9	27.0	27.2	30.5	34.6
CSUMG	8.4	8.9	9.92	10.2	10.7	11.1	12.0	12.8	13.6	20.5	34.6
CSUML	10.2	10.6	11.60	12.1	12.2	12.9	13.5	14.3	14.9	22.2	34.6
RRF	8.5	9.0	9.96	10.3	11.2	11.6	12.4	13.0	13.6	20.5	34.6
IRF	7.9	8.4	9.30	9.74	10.8	10.9	11.6	11.9	12.8	20.8	34.6

Tables 4.126 and 4.127 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.126 SUM QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3296	2772	2734	2842	2683	2681	2677	2365	1691	2023	2055
CSUMG	982	893	652	603	572	603	865	942	930	1963	2055
CSUML	1919	1659	1583	1714	1153	1139	1388	1440	1420	2014	2055
RRF	1665	1853	1739	1691	1561	1618	1775	1620	1309	1964	2055
IRF	998	875	670	533	530	557	819	896	896	1963	2055

Table 4.127 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	15.5	16.0	17.0	16.8	17.5	17.8	19.0	18.1	17.2	21.3	34.6
CSUMG	5.76	6.14	6.7	7.12	7.5	8.1	9.1	10.1	11.0	19.4	34.6
CSUML	8.88	9.66	11.0	9.78	10.	10.3	11.4	12.3	13.4	20.4	34.6
RRF	7.96	8.42	9.3	9.04	9.7	10.5	11.4	11.4	12.0	19.9	34.6
IRF	5.44	6.10	7.1	6.22	6.9	7.8	8.7	9.7	10.6	19.4	34.6

Tables 4.128 and 4.129 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.128 SUM QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	2490	2253	2439	2745	2600	2303	2054	1826	1184	2008	2055
CSUMG	1156	1030	1543	1282	876	861	819	896	896	1963	2055
CSUML	1149	1503	1674	2086	1312	884	1144	1203	898	1963	2055
RRF	789	676	1134	1522	1195	857	1102	900	895	1963	2055
IRF	1019	835	984	526	525	552	817	894	895	1963	2055

Table 4.129 SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	12.4	12.9	13.7	13.8	14.8	15.2	15.8	15.2	14.6	20.6	34.6
CSUMG	6.58	7.18	8.78	7.80	7.66	8.42	8.78	9.70	10.6	19.4	34.6
CSUML	7.12	7.86	8.78	9.02	9.46	8.42	9.42	10.4	11.0	19.4	34.6
RRF	6.00	6.48	7.04	8.02	8.32	8.26	9.12	9.94	10.5	19.4	34.6
IRF	5.54	6.32	7.76	6.22	6.76	7.70	8.64	9.58	10.5	19.4	34.6

Results for Rank-S resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Tables 4.130. At the best case the NDCG@20 score of our approach is 3% less than NDCG@20 score of sending to all selected resources, and we can decrease the network cost up to 30% if we sacrifice from the response time. At the worst case, the NDCG@20 score of our approach is 21% less than NDCG@20 score of sending to all selected resources, but we can decrease the network cost up to 64% if we sacrifice from the response time.

Table 4.130 Rank-S RS algorithm NDCG QPP

	NDCG@20 Score			
	SP	ML Rat	ML Diff	All
Oracle	0.70186	0.78816	0.74114	0.83622
CSUMG	0.35078	0.28340	0.30752	0.35818
CSUML	0.34760	0.29552	0.29328	0.35462
RRF	0.35962	0.32284	0.31022	0.38618
IRF	0.28042	0.23904	0.25042	0.29614

Tables 4.131 and 4.132 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.131 NDCG QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3311	2798	2690	2530	2486	2385	2257	2100	2040	1850	1788
CSUMG	3776	3224	2961	2623	2582	2429	2286	2240	2121	1836	1788
CSUML	3788	3230	2952	2591	2577	2450	2302	2248	2125	1844	1788
RRF	4103	3370	2965	2695	2712	2531	2422	2246	2113	1832	1788
IRF	4284	3561	3077	2863	2802	2590	2428	2247	2128	1864	1788

Table 4.132 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	19.7	20.2	21.0	21.2	21.5	21.9	21.7	22	22.3	23.7	26.2
CSUMG	18.5	19.0	20.1	20.4	20.7	21.1	21.7	22.1	22.2	23.5	26.2
CSUML	18.4	18.9	19.9	20.4	20.5	21.1	21.2	21.8	22.2	23.4	26.2
RRF	20.3	20.8	21.7	22.2	22.5	22.8	22.9	23.5	23.6	25.1	26.2
IRF	21.9	22.4	23.4	23.6	24.1	24.0	24.4	24.6	24.8	25.6	26.2

Tables 4.133 and 4.134 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.133 NDCG QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	4727	3783	3027	2813	2766	2584	2439	2388	2240	1817	1788
CSUMG	3152	2833	2615	2373	2415	2232	2044	1973	1848	1743	1788
CSUML	3170	2791	2616	2418	2404	2164	2001	1971	1832	1749	1788
RRF	3885	3353	2893	2693	2651	2510	2370	2305	2016	1792	1788
IRF	3436	2926	2671	2389	2425	2269	2169	1946	1780	1752	1788

Table 4.134 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	21.9	22.1	22.4	22.4	22.9	22.5	23.1	23.1	22.8	22.8	26.2
CSUMG	9.66	10.2	11.0	11.2	11.6	12.2	12	12.7	13.0	18.5	26.2
CSUML	10.2	10.8	11.3	12.3	12.5	12.1	12.8	12.6	13.4	18.4	26.2
RRF	14.0	14.2	14.9	15.5	15.4	16.3	16.3	17.0	16.7	19.3	26.2
IRF	11.3	11.7	12.9	12.3	12.9	14.1	15.1	14.2	13.4	18.7	26.2

Tables 4.135 and 4.136 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.135 NDCG QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	4556	3698	2976	2790	2757	2596	2440	2352	2207	1806	1788
CSUMG	3364	2935	2646	2408	2535	2377	2175	2037	1898	1746	1788
CSUML	3118	2802	2483	2292	2332	2268	2110	1878	1677	1744	1788
RRF	3555	3177	2830	2607	2488	2381	2241	2166	1919	1754	1788
IRF	3561	3097	2692	2452	2436	2264	2148	2039	1987	1745	1788

Table 4.136 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	18.6	18.9	19.5	19.5	19.7	20.3	20.4	19.2	19.9	21.4	26.2
CSUMG	10.7	11.0	11.8	12.1	12.6	13.8	13.6	14.0	13.4	18.6	26.2
CSUML	9.4	9.9	11.3	10.5	11.0	12.4	13.4	12.0	11.6	18.4	26.2
RRF	13.2	13.4	14.3	14.9	14.2	15.2	15.8	16.4	15.9	19.6	26.2
IRF	10.9	11.3	12.0	12.5	12.9	13.9	13.7	13.2	13.8	18.4	26.2

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.137. At the best case the NDCG@20 score of our approach is 1% less than NDCG@20 score of sending to all selected resources, but we can decrease the network cost up to 9% if we sacrifice from the response time. At the worst case, the NDCG@20 score of our approach is 26% less than NDCG@20 score of sending to all selected resources, but we can decrease the network cost up to 53% if we sacrifice from the response time.

Table 4.137 Rank-S algorithm SUM QPP

	NDCG@20 Score			
	SP	ML Rat	ML Diff	All
Oracle	0.73172	0.79222	0.79102	0.83622
CSUMG	0.35182	0.30508	0.32134	0.35818
CSUML	0.35002	0.28726	0.26936	0.35462
RRF	0.37138	0.32228	0.28436	0.38618
IRF	0.28740	0.24826	0.24198	0.29614

Tables 4.138 and 4.139 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.138 SUM QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3386	2924	2810	2596	2642	2485	2255	2102	2038	1847	1788
CSUMG	4505	3745	3093	2869	2824	2608	2432	2252	2131	1868	1788
CSUML	4505	3745	3093	2869	2824	2608	2432	2252	2131	1868	1788
RRF	4456	3684	3092	2869	2824	2608	2432	2252	2131	1868	1788
IRF	4411	3647	3090	2868	2823	2607	2445	2251	2131	1868	1788

Table 4.139 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	19.3	19.9	20.7	20.9	21.1	21.5	21.3	21.5	22.0	23.1	26.2
CSUMG	24.0	24.6	25.2	25.3	25.8	25.7	25.5	25.9	25.8	26.0	26.2
CSUML	24.0	24.6	25.2	25.3	25.8	25.7	25.5	25.9	25.8	26.0	26.2
RRF	23.8	24.3	25.0	25.2	25.7	25.6	25.4	25.9	25.8	26.0	26.2
IRF	23.3	23.8	24.5	24.6	25.0	25.2	24.9	25.4	25.4	25.7	26.2

Tables 4.140 and 4.141 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.140 SUM QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	4677	3744	2994	2790	2746	2573	2427	2354	2232	1810	1788
CSUMG	3193	2825	2511	2423	2473	2278	2057	1886	1857	1745	1788
CSUML	3088	2736	2289	2227	2214	2162	2067	1890	1848	1744	1788
RRF	3427	2916	2731	2563	2467	2408	2253	2072	1962	1758	1788
IRF	3087	2794	2614	2441	2320	2283	2096	1823	1786	1751	1788

Table 4.141 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	21.7	21.9	22.2	22.5	22.2	22.5	22.8	22.6	23.0	23.0	26.2
CSUMG	11.4	12.0	12.7	13.0	13.7	14.8	14.6	13.2	14.0	18.5	26.2
CSUML	10.1	10.5	11.3	11.5	12.2	13.5	13.3	12.6	12.7	18.4	26.2
RRF	16.3	16.5	17.1	17.8	17.2	18.0	18.8	18.7	19.2	20.7	26.2
IRF	11.1	11.5	12.5	12.8	12.9	14.1	14.5	13.1	13.5	19.0	26.2

Tables 4.142 and 4.143 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.142 SUM QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	4576	3663	2971	2761	2721	2572	2429	2354	2229	1818	1788
CSUMG	2976	2621	2450	2288	2314	2162	1994	1842	1785	1752	1788
CSUML	2658	2435	2255	2148	2190	2145	1938	1693	1659	1741	1788
RRF	3132	2801	2665	2487	2348	2353	2058	1989	1959	1746	1788
IRF	2980	2857	2615	2344	2333	2275	2040	1953	1801	1744	1788

Table 4.143 SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	22.7	22.9	23.1	23.4	23	23.4	23.6	23.4	23.8	23.8	26.2
CSUMG	11.4	11.7	12.3	13.1	13.4	14.6	13.5	13.6	14.5	18.7	26.2
CSUML	7.8	8.1	9.1	9.4	9.3	10.5	10.7	11.2	11.4	18.3	26.2
RRF	12.4	12.8	13.4	14.3	13.8	15.2	15.4	15.3	15.9	19.1	26.2
IRF	10.1	10.4	11.3	12.0	11.9	13.3	13.2	12.6	12.6	18.4	26.2

Results for Adaptive-k resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.144. At the best case the NDCG@20 score of our approach is 45% better than NDCG@20 score of sending to all selected resources, and we can decrease the network cost up to 79% if we sacrifice from the response time. At the worst case, the NDCG@20 score of our approach is 7% less than NDCG@20 score of sending to all selected resources, but we can decrease the network cost up to 89% if we sacrifice from the response time.

Table 4.144 Adaptive-k RS algorithm NDCG QPP

	NDCG@20 Score				
	SP	ML Rat	ML Diff	All	Best
Oracle	0.71986	0.67626	0.67372	0.790265	0.790265
CSUMG	0.33012	0.33434	**0.33862**	0.330857	0.42076
CSUML	0.3364	**0.34284**	0.33716	0.306102	0.42228
RRF	**0.35164**	0.32746	0.32218	0.34451	0.42872
IRF	**0.33598**	0.31138	0.30452	0.231224	0.40242

Tables 4.145 and 4.146 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.145 NDCG QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	21947	19290	16078	15407	14933	14035	13403	13758	12775	9688	6097
CSUMG	12710	11858	10709	10869	9716	9038	8609	9287	8479	7947	6097
CSUML	7244	7361	6249	6115	6032	5982	5711	6459	6205	6323	6097
RRF	24719	21442	17386	16576	15986	15516	14470	14700	13683	10979	6097
IRF	4169	4184	4929	4397	4503	4637	4522	5469	4877	5732	6097

Table 4.146 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	42.8	43.2	44.1	44.8	45.0	46.0	45.7	45.8	46.8	51.0	72
CSUMG	13.0	13.4	14.3	14.8	15.7	16.1	16.5	17.0	17.8	25.1	72
CSUML	12.9	13.3	14.1	14.7	15.5	16.0	16.1	16.6	17.4	24.6	72
RRF	15.5	15.8	16.7	17.7	18.0	19.2	19.2	19.8	20.8	28.5	72
IRF	8.92	9.32	10.4	11.1	11.4	12.5	12.8	13.2	14.0	22.6	72

Tables 4.147 and 4.148 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.147 NDCG QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	11508	11031	10032	9949	9513	9115	9012	10037	9189	7930	6097
CSUMG	5189	5068	5656	6366	6941	6829	5555	4183	3888	5733	6097
CSUML	4852	5142	5456	6452	6691	5012	3688	4671	4653	5733	6097
RRF	4195	4077	3767	4279	4134	4290	4275	4927	4539	5733	6097
IRF	3856	3600	3917	3416	3181	3359	3357	4105	3901	5733	6097

Table 4.148 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	16.7	17.0	18.3	19.1	18.6	18.7	19.8	21.0	20.8	24.9	72
CSUMG	7.9	8.5	9.76	9.3	10.6	11.4	12.2	10.6	10.5	19.9	72
CSUML	8.0	8.5	9.04	10.1	11.3	11.0	10.2	11.0	11.8	19.9	72
RRF	6.3	6.6	7.36	7.6	8.9	9.2	10.5	11.5	11.8	19.9	72
IRF	5.8	6.4	8.00	6.8	7.8	8.4	9.3	10.4	10.8	19.9	72

Tables 4.149 and 4.150 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.149 NDCG QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	11944	11267	9932	9785	10031	9597	9127	10018	8803	7466	6097
CSUMG	4839	5195	6371	7034	7529	5770	4116	4328	4005	5733	6097
CSUML	5214	5158	5707	6376	6737	5422	4658	4781	4653	5733	6097
RRF	5361	5469	5825	6013	5796	5621	5089	5518	4858	5733	6097
IRF	4132	3986	5027	4536	3636	3588	3371	4070	3888	5733	6097

Table 4.150 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	19.3	19.7	20.4	21.3	21.7	21.8	22.1	23.1	22.9	28.0	72
CSUMG	8.4	8.8	9.6	10.6	11.4	11.4	10.6	10.6	11.1	19.9	72
CSUML	8.6	9.2	9.9	10.9	11.2	11.1	11.1	11.7	12.5	19.9	72
RRF	7.3	7.7	8.4	8.9	9.8	10.4	11.0	11.5	11.1	19.9	72
IRF	6.7	7.2	8.5	8.4	8.5	8.72	9.36	9.70	10.5	19.9	72

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.151. At the best case the NDCG@20 score of our approach is 38% better than NDCG@20 score of sending to all selected resources, and we can decrease the network cost up to 92% if we sacrifice from the response time. At the worst case, the NDCG@20 score of our approach is 11% less than NDCG@20 score of sending to all selected resources, but we can decrease the network cost up to 79% if we sacrifice from the response time.

Table 4.151 Adaptive-k algorithm SUM QPP

	NDCG@20 Score				
	SP	ML Rat	ML Diff	All	Best
Oracle	0.74346	0.64552	0.6873	0.790265	0.79026
CSUMG	0.30016	**0.34246**	0.32864	0.330857	0.42076
CSUML	0.28816	**0.34268**	0.33494	0.306102	0.42228
RRF	0.30982	**0.33486**	0.31666	0.344510	0.42872
IRF	0.3097	**0.31922**	0.30314	0.231224	0.40242

Tables 4.152 and 4.153 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.152 SUM QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	22328	19712	16345	15682	14972	14345	13527	14049	12598	9663	6097
CSUMG	21846	19052	15789	14972	14276	13794	13000	13672	12519	10238	6097
CSUML	21893	19174	15739	14923	14241	13758	12969	13639	12493	10210	6097
RRF	21811	19016	15784	14964	14270	13788	12994	13665	12512	10238	6097
IRF	2535	2540	2894	2990	3050	3208	3208	4127	4088	5732	6097

Table 4.153 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	42.8	43.2	44.1	44.8	45.0	46.0	45.7	45.8	46.8	51.0	72
CSUMG	13.0	13.4	14.3	14.8	15.7	16.1	16.5	17.0	17.8	25.1	72
CSUML	12.9	13.3	14.1	14.7	15.5	16.0	16.1	16.6	17.4	24.6	72
RRF	15.5	15.8	16.7	17.7	18.0	19.2	19.2	19.8	20.8	28.5	72
IRF	8.9	9.3	10.4	11.1	11.4	12.5	12.8	13.2	14.0	22.6	72

Tables 4.154 and 4.155 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.154 SUM QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	9838	9635	8559	8913	8717	8012	7481	8334	7077	6645	6097
CSUMG	4379	4431	5228	5269	4465	3717	3477	4046	3888	5733	6097
CSUML	4683	4922	5726	6851	7364	5257	3494	3877	3888	5733	6097
RRF	4811	5234	5122	5017	5205	4995	4957	5768	4693	5749	6097
IRF	3619	3614	4185	3965	3421	2987	2978	3869	3888	5733	6097

Table 4.155 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	
Oracle	13.5	14.1	14.5	16.1	16.3	16.1	16.0	17.1	17.8	23.1	72
CSUMG	6.6	7.4	8.4	8.4	9.2	8.3	9.2	10.1	10.5	19.9	72
CSUML	7.5	8.0	8.4	9.5	11.2	10.5	9.2	9.7	10.5	19.9	72
RRF	7.2	7.8	8.5	8.6	9.6	9.9	10.8	12.0	12.2	20.3	72
IRF	5.5	6.1	6.9	7.6	7.4	7.8	8.7	9.5	10.5	19.9	72

Tables 4.156 and 4.157 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.156 SUM QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	12581	11619	10006	10293	10147	9599	9236	9843	8781	7333	6097
CSUMG	4547	4423	5392	6024	5293	4440	3879	4119	3888	5733	6097
CSUML	4869	4652	5106	5447	5796	5166	3628	3869	3888	5733	6097
RRF	4684	4523	4564	5051	4936	4812	4245	5197	5024	5733	6097
IRF	3458	3455	4046	3835	3589	3486	3262	4086	3888	5733	6097

Table 4.157 SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	21.1	21.4	22.3	23.2	22.6	23.5	24.2	24.9	25.7	30.8	72
CSUMG	7.0	7.7	8.8	8.7	9.4	9.4	9.6	9.7	10.5	19.9	72
CSUML	7.1	7.5	8.5	8.6	9.4	10.7	9.2	9.5	10.5	19.9	72
RRF	7.2	7.6	7.9	8.7	9.4	10.1	10.5	11.3	12.4	19.9	72
IRF	6.1	6.4	7.6	8.3	7.3	8.0	8.8	9.8	10.5	19.9	72

4.3.1.2.2. Evaluation Results for Optimizing P@20

In this section, the training of the queries are based on the best index of the merged lists with highest P@20 score and the quality results are given in P@20 metric. . The results for applying our approach are grouped under ReDDE, Rank-S and Adaptive-k resource selection algorithms.

Results for ReDDE resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.158. At the best case the P@20 score of our approach is 56% better than P@20 score of sending to all selected resources, and we can decrease the network cost up to 87%. At the worst case, the P@20 score of our approach is 3% less than P@20 score of sending to all selected resources, but we can decrease the network cost up to 75% if we sacrifice from the response time.

Table 4.158 ReDDE RS algorithm NDCG QPP

	P@20 Score				
	SP	ML Rat	ML Diff	All	Best
Oracle	0.87672	0.82150	0.81662	0.964286	0.96428
CSUMG	0.55796	0.61054	**0.61110**	0.556122	0.72080
CSUML	0.54402	**0.59706**	0.57676	0.545918	0.71864
RRF	0.55336	0.56700	**0.57226**	0.566327	0.70600
IRF	0.53082	0.56722	**0.57922**	0.370408	0.69404

Tables 4.159 and 4.160 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.159 NDCG QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	4255	3786	3061	3230	2665	2509	2560	2578	2381	2199	2055
CSUMG	2340	2218	2105	2203	1762	1706	1818	1858	1793	2036	2055
CSUML	2629	2364	2124	2041	1788	1760	1846	1872	1793	2093	2055
RRF	625	754	664	902	782	555	815	892	893	1962	2055
IRF	707	824	798	962	854	621	851	899	893	1962	2055

Table 4.160 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	24.3	24.8	25.6	26.1	26.5	26.4	26.9	27.0	27.2	30.5	34.6
CSUMG	8.4	8.9	9.9	10.2	10.7	11.1	12.0	12.8	13.6	20.5	34.6
CSUML	10.2	10.6	11.6	12.1	12.2	12.9	13.5	14.3	14.9	22.2	34.6
RRF	8.5	9.0	9.9	10.3	11.2	11.6	12.4	13.0	13.6	20.5	34.6
IRF	7.9	8.4	9.3	9.7	10.8	10.9	11.6	11.9	12.8	20.8	34.6

Tables 4.161 and 4.162 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.161 NDCG QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	1235	1359	1796	2010	979	558	820	896	896	1963	2055
CSUMG	1208	1029	811	709	680	679	906	978	970	2007	2055
CSUML	635	565	724	889	816	557	819	896	896	1963	2055
RRF	652	794	532	571	573	596	817	894	895	1963	2055
IRF	610	496	591	524	526	552	817	894	895	1963	2055

Table 4.162 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	7.04	7.46	8.70	9.90	8.62	8.00	8.94	9.70	10.6	19.4	34.6
CSUMG	6.38	6.82	7.46	7.56	8.14	9.12	9.62	10.50	11.2	19.8	34.6
CSUML	5.04	5.74	6.22	6.92	7.18	7.86	8.78	9.70	10.6	19.4	34.6
RRF	4.72	5.26	5.42	6.12	7.12	7.98	8.64	9.58	10.5	19.4	34.6
IRF	4.68	5.38	5.74	6.22	7.00	7.84	8.80	9.76	10.7	19.4	34.6

Tables 4.163 and 4.164 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.163 NDCG QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	1384	1454	2003	1637	874	908	1170	1227	1170	1963	2055
CSUMG	938	805	959	673	586	552	817	894	895	1963	2055
CSUML	694	877	358	601	528	552	817	894	895	1963	2055
RRF	685	829	628	604	578	603	821	896	896	1963	2055
IRF	621	500	331	529	525	552	817	894	895	1963	2055

Table 4.164 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	7.06	7.50	9.02	9.42	8.02	8.28	9.26	10.20	11.2	19.4	34.6
CSUMG	5.40	6.12	6.56	7.42	7.96	7.70	8.64	9.58	10.5	19.4	34.6
CSUML	5.28	5.72	6.24	6.82	7.24	7.70	8.64	9.58	10.5	19.4	34.6
RRF	5.48	5.86	6.30	7.12	7.42	8.42	9.10	9.70	10.6	19.4	34.6
IRF	4.52	5.18	5.50	6.12	6.76	7.70	8.64	9.58	10.5	19.4	34.6

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.156. At the best case the P@20 score of our approach is 52% better than P@20 score of sending to all selected resources, and we can decrease the network cost up to 89% if we sacrifice from the response time. At the worst case, the P@20 score of our approach is 7% less than P@20 score of sending to all selected resources, but we can decrease the network cost up to 83% if we sacrifice from the response time.

Table 4.165 ReDDE algorithm SUM QPP

	P@20 Score				
	SP	ML Rat	ML Diff	All	Best
Oracle	0.88100	0.8235	0.7693	0.964286	0.96428
CSUMG	0.52098	**0.5872**	0.5735	0.556122	0.72080
CSUML	0.51998	**0.5903**	0.5876	0.545918	0.71864
RRF	0.53198	**0.5694**	0.5736	0.566327	0.70600
IRF	0.56642	**0.5637**	0.5736	0.370408	0.69404

Tables 4.166 and 4.167 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.166 SUM QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	4761	4028	3517	3372	3050	2913	2914	2625	2396	2258	2055
CSUMG	4352	3714	3001	3067	2743	2688	2690	2714	2607	2256	2055
CSUML	4360	3710	2990	3057	2732	2678	2681	2703	2598	2250	2055
RRF	4375	3724	3001	3067	2743	2688	2690	2714	2607	2257	2055
IRF	738	598	497	946	690	686	914	971	971	1962	2055

Table 4.167 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	8.82	9.18	10.3	11.0	11.0	11.3	11.9	12.7	13.5	20.9	34.6
CSUMG	5.92	6.32	7.6	8.0	8.7	9.2	10.3	10.9	11.9	19.4	34.6
CSUML	5.00	5.32	6.6	7.2	7.6	8.2	9.3	10.1	11.1	19.2	34.6
RRF	6.40	6.72	8.0	8.5	9.4	10.1	11.0	11.4	12.1	19.7	34.6
IRF	4.08	4.40	5.8	6.3	7.1	7.8	8.9	9.3	10.1	18.8	34.6

Tables 4.168 and 4.169 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.168 SUM QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	1248	1383	1813	1490	695	558	820	897	896	1963	2055
CSUMG	686	765	671	911	894	861	820	900	900	1963	2055
CSUML	829	681	473	689	625	645	904	976	970	2007	2055
RRF	750	844	698	901	898	925	887	930	914	1963	2055
IRF	667	569	654	626	615	647	852	896	896	1963	2055

Table 4.169 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	7.10	7.34	8.70	10.4	8.26	8.00	8.94	9.88	10.6	19.4	34.6
CSUMG	5.24	5.94	6.14	7.2	7.90	8.28	8.94	9.88	10.8	19.4	34.6
CSUML	5.48	6.28	6.64	7.1	7.78	8.56	9.46	10.40	11.2	19.8	34.6
RRF	5.58	5.98	6.22	7.2	7.78	8.70	9.10	9.88	10.8	19.4	34.6
IRF	5.28	5.7	6.06	6.8	7.30	8.28	8.94	9.70	10.6	19.4	34.6

Tables 4.170 and 4.171 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.170 SUM QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	1152	974	1505	950	569	552	817	894	895	1963	2055
CSUMG	1219	1164	881	943	845	879	1142	1221	1216	1963	2055
CSUML	679	572	393	584	554	570	817	894	895	1963	2055
RRF	711	586	677	876	877	887	855	935	930	1963	2055
IRF	659	551	659	580	556	559	821	900	901	1963	2055

Table 4.171 SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	6.02	6.60	7.92	7.92	7.36	7.70	8.64	9.5	10.5	19.4	34.6
CSUMG	5.32	6.06	6.46	7.30	7.96	8.26	9.28	10.3	11.3	19.4	34.6
CSUML	4.88	5.46	5.82	6.62	7.12	8.12	8.64	9.5	10.5	19.4	34.6
RRF	5.48	6.20	6.52	7.02	8.14	8.84	9.26	10.2	11.2	19.4	34.6
IRF	5.16	5.88	6.12	7.12	7.78	8.14	9.10	10.1	11.0	19.4	34.6

Results for Rank-S resource selection: Using NDCG-QPP as the QPP method, we provide the effectiveness results in Table 4.172. At the best case the P@20 score of our approach is 0,8% better than P@20 score of sending to all selected resources, and we can decrease the network cost up to 72% if we sacrifice from the response time. At the worst case, the P@20 score of our approach is 26% less than P@20 score of sending to all selected resources, but we can decrease the network cost up to 68% if we sacrifice from the response time.

Table 4.172 Rank-S RS algorithm NDCG QPP

	P@20 Score			
	SP	ML Rat	ML Diff	All
Oracle	0.80842	0.71800	0.69400	0.903
CSUMG	**0.51676**	0.37918	0.44176	0.512
CSUML	0.49842	0.40434	0.41576	0.507
RRF	0.42875	0.38462	0.39236	0.491
IRF	**0.37152**	0.35722	0.33700	0.368

Tables 4.173 and 4.174 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.173 NDCG QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3599	3131	2805	2545	2600	2506	2370	2163	2030	1846	1788
CSUMG	3776	3224	2961	2623	2582	2429	2286	2240	2121	1836	1788
CSUML	3788	3230	2952	2591	2577	2450	2302	2248	2125	1844	1788
RRF	3379	2782	2655	2514	2523	2412	2311	2128	2017	1835	1788
IRF	2130	1940	2111	2091	2048	2037	1896	1774	1789	1723	1788

Table 4.174 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	18.0	18.5	19.4	19.4	19.9	20.0	20.2	20.9	21.1	22.9	26.2
CSUMG	18.5	19.0	20.1	20.4	20.7	21.1	21.7	22.1	22.2	23.5	26.2
CSUML	18.4	18.9	19.9	20.4	20.5	21.1	21.2	21.8	22.2	23.4	26.2
RRF	13.2	13.6	14.6	15.4	15.7	16.4	16.6	17.4	18.1	21.8	26.2
IRF	7.3	7.7	8.6	9.7	9.8	10.2	10.7	11.5	12.0	17.9	26.2

Tables 4.175 and 4.176 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.175 NDCG QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3455	2900	2572	2371	2302	2223	2078	1958	1918	1755	1788
CSUMG	2868	2529	2292	2073	2010	1954	1911	1825	1820	1744	1788
CSUML	3196	2788	2472	2378	2381	2259	2052	1992	1943	1744	1788
RRF	3186	2708	2514	2316	2358	2321	2132	1895	1873	1744	1788
IRF	3307	2874	2559	2415	2354	2256	2103	1993	1919	1745	1788

Table 4.176 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	13.2	13.4	13.9	14.9	14.0	15.2	15.0	15.3	16.1	19.3	26.2
CSUMG	8.4	8.9	9.6	10.4	10.1	11.3	11.7	11.6	12.1	18.4	26.2
CSUML	9.2	9.4	10	10.6	11.2	11.9	11.8	12.7	13.0	18.4	26.2
RRF	9.4	9.8	10.9	11.0	10.8	11.5	12.6	12.6	13.3	18.4	26.2
IRF	9.5	9.9	11.4	11.3	10.7	11.4	12.1	11.9	12.8	18.9	26.2

Tables 4.177 and 4.178 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.177 NDCG QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3185	2792	2530	2308	2166	2052	2006	1921	1805	1751	1788
CSUMG	3070	2771	2485	2277	2392	2314	2150	2020	1775	1747	1788
CSUML	3252	2859	2539	2344	2377	2386	2212	2072	1936	1744	1788
RRF	3489	3004	2641	2493	2433	2300	2175	2118	2097	1752	1788
IRF	3036	2647	2406	2278	2379	2267	2093	1762	1707	1745	1788

Table 4.178 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	11.4	11.7	12.4	13.0	12.9	13.6	14	14.3	14.5	19.0	26.2
CSUMG	9.9	10.1	10.8	11.0	11.5	12.7	13.4	13.5	12.9	18.9	26.2
CSUML	9.2	9.3	10.3	10.4	10.8	11.8	12.8	13.4	12.0	18.4	26.2
RRF	11.3	11.7	12.7	13.4	12.8	13.0	14.2	14.0	15.1	18.7	26.2
IRF	10	10.5	11.6	11.4	11.7	12.8	13.3	12.3	12.7	18.5	26.2

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.179. At the best case the P@20 score of our approach is 0,2% less than P@20 score of sending to all selected resources, but we can decrease the network cost up to 13% if we sacrifice from the response time. At the worst case, the P@20 score of our approach is 25% less than P@20 score of sending to all selected resources, but we can decrease the network cost up to 58% if we sacrifice from the response time.

Table 4.179 Rank-S algorithm SUM QPP

	P@20 Score			
	SP	ML Rat	ML Diff	All
Oracle	0.83642	0.71600	0.75300	0.903
CSUMG	0.50842	0.42598	0.43176	0.512
CSUML	0.50242	0.42276	0.41898	0.507
RRF	0.48042	0.37100	0.39312	0.491
IRF	0.36742	0.30764	0.32466	0.368

Tables 4.180 and 4.181 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.180 SUM QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3641	3166	2812	2602	2644	2484	2255	2102	2038	1847	1788
CSUMG	4461	3686	3092	2869	2824	2608	2446	2252	2132	1868	1788
CSUML	4424	3659	3076	2857	2812	2596	2435	2242	2122	1860	1788
RRF	4510	3747	3093	2869	2824	2608	2446	2252	2132	1868	1788
IRF	4459	3708	3091	2868	2823	2607	2445	2251	2131	1868	1788

Table 4.181 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	18.6	19.1	19.9	20.1	20.4	20.7	20.6	20.7	21.4	22.8	26.2
CSUMG	24.0	24.6	25.2	25.3	25.7	25.7	25.6	25.9	25.9	26.0	26.2
CSUML	22.6	23.1	23.8	23.8	24.5	24.3	24.4	24.6	24.4	24.9	26.2
RRF	24.3	24.8	25.4	25.5	25.9	25.8	25.7	25.9	25.9	26.0	26.2
IRF	23.6	24.1	24.7	24.8	25.2	25.3	25.1	25.4	25.4	25.7	26.2

Tables 4.182 and 4.183 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.182 SUM QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3297	2831	2601	2390	2346	2196	2080	1997	1879	1752	1788
CSUMG	3409	2964	2501	2366	2360	2264	2103	2024	1970	1753	1788
CSUML	3149	2713	2508	2360	2480	2377	2162	1756	1721	1751	1788
RRF	3117	2759	2536	2436	2561	2239	1846	1784	1745	1749	1788
IRF	2485	2268	1812	1780	1775	1729	1712	1656	1664	1744	1788

Table 4.183 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	11.4	11.7	12.5	12.8	12.9	13.5	14.3	14.7	13.7	18.6	26.2
CSUMG	10.3	10.6	11.4	11.4	11.7	12.9	13.8	12.8	13.6	18.8	26.2
CSUML	9.4	9.8	10.7	10.4	11.4	12.3	12.7	12.0	12.3	18.6	26.2
RRF	11	11.3	12	12.4	13.5	14.6	13.2	13.4	13.4	18.8	26.2
IRF	8.1	8.6	9.2	9.3	10.2	11.1	11.6	11.5	12.0	18.9	26.2

Tables 4.184 and 4.185 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference.

Table 4.184 SUM QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3407	2872	2611	2417	2358	2221	2103	2054	2024	1752	1788
CSUMG	3067	2765	2515	2400	2345	2261	2127	1910	1869	1746	1788
CSUML	3303	3021	2677	2502	2447	2322	2180	2066	1860	1753	1788
RRF	3741	3163	2830	2701	2620	2534	2396	2221	2040	1789	1788
IRF	2994	2597	2299	2039	2112	2021	1903	1828	1804	1751	1788

Table 4.185 SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	13.4	13.7	14.5	15.0	15.0	15.2	15.5	15.9	16.8	19.1	26.2
CSUMG	9.6	10.1	10.7	11.3	11.8	11.5	12.5	12.5	13.5	18.6	26.2
CSUML	9.8	10.3	10.9	11.5	11.9	11.3	12.2	12.5	13.2	18.8	26.2
RRF	13.1	13.6	13.9	14.7	14.5	15.7	16.1	16.5	17.0	19.5	26.2
IRF	8.7	9.2	10.2	10.5	10.0	11.0	11.8	12.2	12.9	18.6	26.2

Results for Adaptive-k resource selection: Using NDCG-QPP as the QPP method, we give the effectiveness results in Table 4.186. At the best case the P@20 score of our approach is 99% better than P@20 score of sending to all selected resources, and we can decrease the network cost up to 94% if we sacrifice from the response time. At the worst case, the P@20 score of our approach is 0,2% less than P@20 score of sending to all selected resources, but we can decrease the network cost up to 87% if we sacrifice from the response time.

Table 4.186 Adaptive-k RS algorithm NDCG QPP

	P@20 Score				
	SP	ML Rat	ML Diff	All	Best
Oracle	0.81700	0.64278	0.65267	0.860204	0.86700
CSUMG	0.48894	**0.51192**	0.51544	0.489796	0.65920
CSUML	**0.52228**	0.51826	0.51438	0.455102	0.66046
RRF	**0.53244**	0.51079	0.49216	0.443878	0.64658
IRF	**0.53244**	0.48685	0.50300	0.267347	0.62834

Tables 4.187 and 4.188 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.187 NDCG QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	24026	21158	17413	16657	16068	15127	14360	14749	13542	10189	6097.4
CSUMG	12710	11858	10709	10869	9716	9038	8609	9287	8479	7947	6097.4
CSUML	6661	6140	5548	5442	5320	5069	5022	5793	5556	6298	6097.4
RRF	2922	3219	3692	3770	3715	3490	3216	4123	4136	5732	6097.4
IRF	2922	3219	3692	3770	3715	3490	3216	4123	4136	5732	6097.4

Table 4.188 NDCG QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	14.3	14.7	15.7	16.6	17.0	17.2	17.5	18.4	19.4	26.6	72
CSUMG	9.4	9.8	11.0	11.4	12.5	13.3	13.6	14.3	15.0	23.9	72
CSUML	6.2	6.5	7.8	8.3	9.4	10.1	10.4	10.8	11.6	20.5	72
RRF	10.8	11.2	12.4	13.3	13.7	14.7	15.2	16.2	17.4	25.8	72
IRF	4.3	4.6	5.9	6.7	7.36	8.3	8.8	9.7	10.4	19.8	72

Tables 4.189 and 4.190 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.189 NDCG QPP ML Ratio PLL Cost

	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	4344	4760	5732	6149	6338	5186	3947	4280	3957	5733	6097
CSUMG	3380	3601	3036	2748	2796	2958	2963	3869	3888	5733	6097
CSUML	3604	3390	4702	5147	4345	3945	3083	3869	3888	5733	6097
RRF	2614	2932	2661	2777	2824	2985	2981	3869	3888	5733	6097
IRF	3109	3642	3206	3380	3404	3579	3593	4346	3888	5733	6097

Table 4.190 NDCG QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	7.44	7.72	8.88	9.68	9.76	10.8	10.8	10.4	11.1	19.9	72
CSUMG	5.16	5.84	6.96	5.68	6.64	7.6	8.5	9.5	10.5	19.9	72
CSUML	5.86	6.12	7.28	8.28	8.32	9.0	8.7	9.5	10.5	19.9	72
RRF	4.02	4.56	5.20	5.98	7.00	8.0	9.0	9.5	10.5	19.9	72
IRF	5.06	5.28	5.92	6.58	7.72	8.8	9.5	10.6	10.5	19.9	72

Tables 4.191 and 4.192 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.191 NDCG QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	4966	4991	5869	6206	6138	5434	4310	4714	4726	5733	6097
CSUMG	3608	3952	3565	3123	2796	2958	2963	3869	3888	5733	6097
CSUML	3304	3194	4707	4921	3958	3462	3127	4004	3888	5733	6097
RRF	3534	3462	3967	3798	3334	3543	3167	3869	3888	5733	6097
IRF	2822	2973	3161	2902	2966	3128	3133	4007	3888	5733	6097

Table 4.192 NDCG QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	7.74	8.20	9.36	9.68	10.5	10.5	10.2	10.6	11.7	19.9	72
CSUMG	5.46	6.16	7.60	6.38	6.6	7.6	8.5	9.5	10.5	19.9	72
CSUML	5.50	5.68	6.96	8.38	7.9	8.1	8.7	9.7	10.5	19.9	72
RRF	4.90	5.60	6.16	6.68	7.1	8.1	9.0	9.5	10.5	19.9	72
IRF	4.40	4.92	6.56	5.88	6.8	7.8	8.8	9.8	10.5	19.9	72

Using SUM-QPP as the QPP method, we give the effectiveness results in Table 4.193. At the best case the P@20 score of our approach is 93% better than P@20 score of sending to all selected resources, and we can decrease the network cost up to 94% if we sacrifice from the response time. At the worst case, the P@20 score of our approach is 2% less than P@20 score of sending to all selected resources, but we can decrease the network cost up to 92% if we sacrifice from the response time.

Table 4.193 Adaptive-k algorithm SUM QPP

	P@20 Score				
	SP	ML Rat	ML Diff	All	Best
Oracle	0.827	0.66480	0.62800	0.860204	0.8670
CSUMG	0.482	0.51344	**0.52716**	0.489796	0.6592
CSUML	0.448	**0.52418**	0.51924	0.455102	0.6604
RRF	0.439	**0.52534**	0.51677	0.443878	0.6465
IRF	0.266	0.49685	**0.51728**	0.267347	0.6283

Tables 4.194 and 4.195 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use SP as the stopping condition.

Table 4.194 SUM QPP Stop Policy PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	22328	19712	16345	15682	14972	14345	13527	14049	12598	9663	6097.4
CSUMG	28731	25157	20373	19371	18338	17366	16287	16762	15304	11681	6097.4
CSUML	28731	25157	20373	19371	18338	17366	16287	16762	15304	11681	6097.4
RRF	28731	25157	20373	19371	18338	17366	16287	16762	15304	11681	6097.4
IRF	28535	24985	20217	19232	18205	17314	16235	16716	15261	11671	6097.4

Table 4.195 SUM QPP Stop Policy Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	14.3	14.7	15.7	16.6	17	17.2	17.5	18.4	19.4	26.6	72
CSUMG	9.4	9.8	11	11.4	12.5	13.3	13.6	14.3	15.0	23.9	72
CSUML	6.2	6.5	7.8	8.3	9.4	10.1	10.4	10.8	11.6	20.5	72
RRF	10.8	11.2	12.4	13.3	13.7	14.7	15.2	16.2	17.4	25.8	72
IRF	4.3	4.6	5.9	6.7	7.3	8.3	8.8	9.7	10.4	19.8	72

Tables 4.196 and 4.197 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Ratio Model as the stopping condition.

Table 4.196 SUM QPP ML Ratio PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	5270	5628	6345	6803	5911	5973	5816	6026	5405	5733	6097
CSUMG	3397	3605	3038	2748	2796	2958	2963	3869	3888	5733	6097
CSUML	3515	3283	4201	3515	3155	3308	3293	3953	3967	5775	6097
RRF	2605	2910	2908	3015	3080	3236	2997	3889	3888	5733	6097
IRF	2784	3528	3198	3366	3398	3580	3626	4359	3888	5733	6097

Table 4.197 SUM QPP ML Ratio Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	7.80	8.12	8.96	10.2	9.64	9.84	10.8	11.3	12	19.9	72
CSUMG	5.12	5.80	6.88	5.68	6.64	7.60	8.5	9.5	10.5	19.9	72
CSUML	6.66	7.10	8.70	8.68	8.54	9.50	10.5	11.2	12.2	21.4	72
RRF	4.04	4.52	5.12	6.08	7.12	8.16	8.7	9.7	10.5	19.9	72
IRF	4.82	5.24	6.16	6.58	7.72	8.86	10	10.8	10.5	19.9	72

Tables 4.198 and 4.199 show the trade-off between the loss in response time and the gain in network for different iteration sizes of *k* if we use ML Difference as the stopping condition.

Table 4.198 SUM QPP ML Difference PLL Cost

	PLL Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	3823	4123	5350	5217	5047	4629	3645	4237	4229	5733	6097
CSUMG	3725	4076	4988	4397	3102	3231	2963	3869	3888	5733	6097
CSUML	3386	3490	4103	4006	3512	3189	2963	3869	3888	5733	6097
RRF	3294	3400	2640	2748	2796	2958	2963	3869	3888	5733	6097
IRF	3156	3370	2640	2748	2796	2958	2963	3869	3888	5733	6097

Table 4.199 SUM QPP ML Difference Network/Resource Usage Cost

	Network Resource Cost										
	k=1	k=2	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=20	All
Oracle	6.5	7.04	8.16	8.88	9.40	9.28	9.36	9.88	10.9	19.9	72
CSUMG	5.8	6.28	7.44	8.38	7.24	8.30	8.56	9.52	10.5	19.9	72
CSUML	5.2	6.04	6.80	7.28	7.48	7.74	8.56	9.52	10.5	19.9	72
RRF	4.4	4.72	4.72	5.68	6.64	7.60	8.56	9.52	10.5	19.9	72
IRF	4.2	4.72	4.72	5.68	6.64	7.60	8.56	9.52	10.5	19.9	72

4.3.1.3. Summary of Findings

The incremental query forwarding approach for early termination in meta-search:

- Can decrease the network cost between 30% and 78% while the result quality is between -27% and 40% in NDCG@20 metric compared to fix cut-off (20 engines) RS algorithms (TWF-IRF, UISP, Oracle, FW13 Baseline).

- Can decrease the network cost between 44% and 90% while the result quality is between -33% and 41% in P@20 metric compared to fix cut-off (20 engines) RS algorithms (TWF-IRF, UISP, Oracle, FW13 Baseline).

- Can decrease the network cost between 92% and 9% while the result quality is between -26% and 45% in NDCG@20 metric compared adaptive cut-off RS algorithms (ReDDE, Rank-S, Adaptive-k).

- Can decrease the network cost between 94% and 13% while the result quality is between -26% and 99% in P@20 metric compared to fix cut-off (20 engines) RS algorithms (ReDDE, Rank-S, Adaptive-k).

- On the average adhoc stop policies predicts the termination more precisely than the ML Models, thus using Stop Policies yields better NDCG@20 score.

- Without applying our approach , for all resource selection algorithms RRF merging algorithm performs the best in NDCG@20 metric. Our approach using adhoc stop policies and RRF merging algorithm yields NDCG@20 scores between -11% and 17% compared to merging the results of all resources.

- Without applying our approach , for 4 out of 7 resource selection algorithms CombSUM (with global idf in TF-IDF calculation) merging algorithm performs best in P@20 metric. Using CombSUM result merging algorithm, our approach yields NDCG@20 score between -28% and 32% compared to merging the results of all resources.

- Using larger values of iteration size *k* may decrease the response time of our approach but also reduces the gains in network cost.

4.3.2. QPP Based Adaptive Incremental Result Merging Approach

In this section, we give the experiment results of our incremental result merging approach. Because we also forward query to all selected resources in this approach, there is no gain or loss in terms of the network/resource usage cost. Instead, when we decide to stop early, we decrease the response time which is calculated in terms of posting list length (PLL) in comparison to waiting all resources. Therefore, in the following sections, we only report the response time.

4.3.2.1. Fix Cut-Off Comparison

In this section we analyze the resource selection algorithms with 20 engines as fix cut-off value . Firstly, we present the results for the cases that we used NDCG@20 score for quality assessing and learning the index of the resources that brings the results with best NDCG@20 score. Next, we present the results for the cases that we used P@20 score for quality assessing and learning the best index of the resources that brings the results with best P@20 score.

4.3.2.1.1. Evaluation Results for Optimizing NDCG@20

In this section, we give the effectiveness results using NDCG@20 metric. The training of the machine learning models and stop policies is based on the stop index with best NDCG@20 score. The results for applying our approach are grouped under TWF-IRF, UISP, Oracle and FedWeb2013 Baseline resource selection algorithms.

Results for TWF-IRF resource selection: Using stop policies among NDCG-QPP scores performs better than machine learning models. At the best case, the quality results of our approach is 0,9% less than merging the results for selected 20 resources, but in terms of response time (estimated as PLL cost) our incremental query merging approach is 82% lower than the response time of waiting all 20 resources. In Table 4.200, for all merging algorithms it can be seen that the gain in response time is more than 75%.

Table 4.200 TWF-IRF Resource Selection NDCG QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.733	0.712	0.716	0.829	270.92	251.7	**208.26**	1230.74
CSUMG	0.332	0.264	0.286	0.359	246.74	**151.6**	160.76	1230.74
CSUML	0.301	0.259	0.252	0.359	279.56	186.4	**184.44**	1230.74
RRF	0.350	0.358	0.318	0.354	228.22	223.9	**190.06**	1230.74
IRF	0.298	0.229	0.253	0.404	247.04	**166.3**	190.96	1230.74

Using stop policies among SUM-QPP scores also performs better than machine learning models which can be seen in Table 4.201. At the best case, the quality results of our approach is 11% better than merging the results for selected 20 resources, and also our approach is 68% lower than the response time of waiting all 20 resources in terms of response time. For TWF-IRF resource selection, stop policies among SUM-QPP scores performs best.

Table 4.201 TWF-IRF Resource Selection SUM QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.730	0.721	0.719	0.829	259.7	263.00	**221.40**	1230.7
CSUMG	0.345	0.283	0.279	0.359	375.18	192.08	**141.04**	1230.7
CSUML	0.342	0.250	0.267	0.359	375.18	**141.00**	155.58	1230.7
RRF	**0.396**	0.297	0.320	0.354	397.26	**171.26**	230.34	1230.7
IRF	0.315	0.237	0.259	0.404	397.26	**161.48**	210.04	1230.7

Results for UISP resource selection: Looking at Tables 4.202 and 4.203, using stop policies among SUM-QPP scores seems to perform best for UISP resource selection algorithm. At the best case the quality of our approach is 4% lower than the quality of using the results of all 20 resources, however the response time of our approach is 82% lower.

Table 4.202 UiSP Resource Selection NDCG QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.733	0.721	0.726	0.853	278.18	234.08	**232.52**	2122.68
CSUMG	0.306	0.277	0.313	0.389	217.86	**165.48**	196.04	2122.68
CSUML	0.299	0.252	0.258	0.379	179.16	116.50	**108.56**	2122.68
RRF	0.353	0.313	0.265	0.397	248.44	**196.52**	218.90	2122.68
IRF	0.283	0.250	0.239	0.310	276.46	**173.04**	177.48	2122.68

Table 4.203 UiSP Resource Selection SUM QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.749	0.759	0.758	0.853	291.98	245.84	**242.96**	2122.68
CSUMG	0.377	0.301	0.303	0.389	385.28	**135.74**	253.90	2122.68
CSUML	0.368	0.255	0.278	0.379	399.42	**113.26**	153.56	2122.68
RRF	0.384	0.276	0.300	0.397	399.42	**198.96**	231.36	2122.68
IRF	0.299	0.236	0.223	0.310	385.28	167.28	**159.92**	2122.68

Results for Oracle resource selection: Stop policies among SUM-QPP scores performs best for oracle resource selection as the results of previous algorithms. At the best case, the NDCG@20 quality result of our approach is 1% better than the result quality of 20 resources and the response time is 84% is lower than the resource with the highest response time.

Table 4.204 Oracle Resource Selection NDCG QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.814	0.938	0.929	0.998	160.38	**205.66**	244.72	2538.26
CSUMG	0.437	0.370	0.340	0.475	280.28	124.08	**88.22**	2538.26
CSUML	0.402	0.346	0.334	0.464	232.02	106.18	**77.48**	2538.26
RRF	0.454	0.408	0.417	0.525	232.40	**89.90**	168.70	2538.26
IRF	0.419	0.366	0.373	0.436	229.48	**84.38**	157.68	2538.26

Table 4.205 Oracle Resource Selection SUM QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.811	0.951	0.944	0.998	129.18	356.08	**326.42**	2538.26
CSUMG	0.468	0.376	0.411	0.475	364.12	**126.48**	321.26	2538.26
CSUML	0.461	0.366	0.396	0.464	360.94	**110.60**	216.08	2538.26
RRF	0.506	0.426	0.415	0.525	379.80	149.78	**122.72**	2538.26
IRF	**0.442**	0.365	0.393	0.436	382.98	288.18	**168.20**	2538.26

Results for Baseline resource selection: Stop policies among SUM-QPP scores also performs best for baseline resource selection. At the best case, the NDCG@20 quality result of our approach is 1% better than the result quality of 20 resources and the response time is 75% is lower than the resource with the highest response time in 20 resources. Table 4.206 and 4.207 contains the result quality and response time costs.

Table 4.206 Baseline Resource Selection NDCG QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.648	0.878	0.902	0.968	135.74	242.68	**229.14**	1238.06
CSUMG	0.396	0.339	0.356	0.440	198.82	**114.14**	158.46	1238.06
CSUML	0.360	0.300	0.315	0.417	180.84	137.92	**122.96**	1238.06
RRF	0.431	0.419	0.425	0.498	205.32	270.94	**185.16**	1238.06
IRF	0.398	0.370	0.365	0.411	227.32	164.18	**123.44**	1238.06

Table 4.207 Baseline Resource Selection SUM QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.695	0.906	0.874	0.968	141.92	389.10	**201.68**	1238.06
CSUMG	0.433	0.345	0.353	0.440	318.04	123.58	**113.58**	1238.06
CSUML	**0.418**	0.361	0.339	0.417	318.04	**129.70**	136.10	1238.06
RRF	0.488	0.414	0.416	0.498	317.88	**150.76**	213.88	1238.06
IRF	**0.416**	0.371	0.358	0.411	317.88	**135.62**	152.34	1238.06

4.3.2.1.2. Evaluation Results for Optimizing P@20

In this section, we give the effectiveness results using P@20 metric. The training of the machine learning models and the training of stop policies are based on the stop index with best P@20 score. The results for applying our approach are grouped under TWF-IRF, UISP, Oracle and FedWeb2013 Baseline resource selection algorithms.

Results for TWF-IRF resource selection: Looking at Tables 4.208 and 4.209, we can see that using the stopping policies among SUM-QPP values performs best on TWF-IRF resource selection. At the best case, our approach performs 1% better in P@20 metric, and lowers the response time 69% compared to forwarding query to 20 resources and waiting for their responses.

Table 4.208 TWF-IRF Resource Selection NDCG QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.854	0.644	0.675	0.918	270.92	117.26	**113.02**	1230.74
CSUMG	0.482	0.400	0.400	0.532	220.74	**68.14**	128.60	1230.74
CSUML	0.45	0.380	0.405	0.524	279.56	**64.42**	111.62	1230.74
RRF	0.433	0.349	0.357	0.501	228.22	**135.00**	159.98	1230.74
IRF	0.393	0.309	0.310	0.397	247.04	**83.26**	100.64	1230.74

Table 4.209 TWF-IRF Resource Selection SUM QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.818	0.718	0.677	0.918	254.72	139.96	**113.34**	1230.74
CSUMG	0.514	0.379	0.334	0.532	397.26	**90.08**	112.28	1230.74
CSUML	0.512	0.375	0.335	0.524	397.26	**73.18**	112.68	1230.74
RRF	0.489	0.329	0.378	0.501	380.56	**136.10**	152.62	1230.74
IRF	**0.403**	0.301	0.320	0.397	380.56	**114.88**	123.46	1230.74

Results for UISP resource selection: Looking at Tables 4.210 and 4.211, we can see that using the stopping policies among SUM-QPP values performs best on UISP resource selection. At the best case, the quality result of our approach is 5% lower compared to "All", but our approach also lowers the response time 82% compared to forwarding query to 20 resources and waiting for their responses.

Table 4.210 UiSP Resource Selection NDCG QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.855	0.736	0.673	0.922	278.18	123.32	**115.96**	2122.68
CSUMG	0.492	0.417	0.421	0.539	240.22	123.16	**113.10**	2122.68
CSUML	0.446	0.370	0.374	0.537	179.16	**81.70**	111.80	2122.68
RRF	0.448	0.354	0.332	0.487	248.44	145.12	**125.**10	2122.68
IRF	0.369	0.300	0.305	0.387	276.46	**94.34**	100.36	2122.68

Table 4.211 UiSP Resource Selection SUM QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.866	0.703	0.695	0.922	288.84	150.94	**122.56**	2122.68
CSUMG	0.532	0.372	0.411	0.539	399.42	113.92	**79.50**	2122.68
CSUML	0.527	0.375	0.389	0.537	399.42	**84.00**	131.12	2122.68
RRF	0.477	0.308	0.398	0.487	399.42	**104.38**	166.58	2122.68
IRF	0.375	0.333	0.304	0.387	395.56	123.62	**97.00**	2122.68

Results for Oracle resource selection: Looking at Tables 4.212 and 4.213, we can see that using the stopping policies among SUM-QPP values performs best on oracle resource selection. At the best case, the quality result of our approach is 2% better than the results of 20 resources, and it also lowers the response time 85% compared to forwarding query to 20 resources and waiting for their responses.

Table 4.212 Oracle Resource Selection NDCG QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.952	0.707	0.737	0.993	197.72	**29.66**	32.02	2538.26
CSUMG	0.629	0.564	0.552	0.648	280.28	56.16	**49.74**	2538.26
CSUML	0.595	0.545	0.554	0.644	211.56	56.54	**29.06**	2538.26
RRF	0.628	0.581	0.547	0.687	232.40	**98.44**	106.54	2538.26
IRF	0.612	0.506	0.554	0.604	229.48	**35.02**	72.34	2538.26

Table 4.213 Oracle Resource Selection SUM QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.928	0.732	0.707	0.993	191.74	33.62	**26.82**	2538.26
CSUMG	0.647	0.527	0.579	0.648	364.12	61.98	**47.16**	2538.26
CSUML	0.643	0.529	0.568	0.644	381.16	**33.72**	57.98	2538.26
RRF	0.673	0.559	0.566	0.687	382.98	**79.76**	91.18	2538.26
IRF	**0.618**	0.5239	0.538	0.604	364.12	76.52	**72.94**	2538.26

Results for Baseline resource selection: Looking at Tables 4.214 and 4.215, we can see that using the stopping policies among SUM-QPP values performs best on oracle resource selection. At the best case, the quality result of our approach is 1% better than the results of 20 resources, and it also lowers the response time 74% compared to forwarding query to 20 resources and waiting for their responses.

Table 4.214 Baseline Resource Selection NDCG QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.786	0.784	0.717	0.988	135.74	69.06	**35.38**	1238.06
CSUMG	0.570	0.488	0.493	0.597	198.82	**71.62**	97.04	1238.06
CSUML	0.552	0.480	0.481	0.587	180.84	98.16	**88.72**	1238.06
RRF	0.595	0.545	0.532	0.668	205.32	119.28	**113.14**	1238.06
IRF	0.557	0.484	0.491	0.559	227.32	100.00	**64.70**	1238.06

Table 4.215 Baseline Resource Selection SUM QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.858	0.776	0.763	0.988	143.92	43.96	**35.48**	1238.06
CSUMG	0.592	0.486	0.518	0.597	318.04	**68.36**	90.96	1238.06
CSUML	0.584	0.486	0.486	0.587	318.04	**77.22**	97.20	1238.06
RRF	0.646	0.494	0.533	0.668	317.22	**101.78**	121.48	1238.06
IRF	**0.568**	0.469	0.514	0.559	317.02	**100.44**	125.34	1238.06

4.3.2.2. Adaptive RS Comparison

In this section we analyze the resource selection algorithms with query adaptive cut-off values. Firstly, the results for the cases that we used NDCG@20 score for quality assessing and learning the index of the resources that brings the results with best NDCG@20 score. Secondly, the results for the cases that we used P@20 score for quality assessing and learning the best index of the resources that brings the results with best P@20 score.

4.3.2.2.1. Evaluation Results for Optimizing NDCG@20

In this section, we give the effectiveness results using NDCG@20 metric. The training of the machine learning models and stop policies is based on the stop index with best NDCG@20 score. The results for applying our approach are grouped under ReDDE, Rank-S and Adaptive-k resource selection algorithms.

Results for ReDDE resource selection: ReDDE selects 34.6 engines on the average for 50 queries. All column in Table 4.216 and 4.217 represents the results of the resources selected by ReDDE. Stopping policies among SUM-QPP values performs best for ReDDE resource selection algorithm. At the best case, the result quality of our approach is %5 lower than the result quality of the all resources selected by

ReDDE. However our approach also lowers the response time %74 compared to waiting all resources.

Table 4.216 ReDDE Resource Selection NDCG QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.703	0.789	0.780	0.916	**190.82**	604.10	547.68	2054.58
CSUMG	0.359	0.259	0.258	0.411	229.84	**302.08**	355.40	2054.58
CSUML	0.339	0.277	0.267	0.408	179.08	**299.12**	315.56	2054.58
RRF	0.360	0.256	0.260	0.436	238.52	**228.86**	256.84	2054.58
IRF	0.278	0.213	0.206	0.316	256.72	**258.62**	359.68	2054.58

Table 4.217 ReDDE Resource Selection SUM QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.762	0.762	0.759	0.916	**224.72**	534.4	465.64	2054.58
CSUMG	0.391	0.261	0.300	0.411	336.06	**257.3**	303.80	2054.58
CSUML	0.381	0.258	0.271	0.408	334.02	**193.4**	308.96	2054.58
RRF	0.421	0.260	0.273	0.436	336.06	313.7	**254.12**	2054.58
IRF	0.300	0.219	0.200	0.316	331.08	271.8	**258.48**	2054.58

Results for Rank-S resource selection: Rank-S selects 26,2 engines on the average for 50 queries. All column in Tables 4.218 and 4.219 represents the results of the resources selected by Rank-S. Stopping policies among SUM-QPP values performs best for Rank-S resource selection algorithm. At the best case, the result quality of our approach is %1 better than the result quality of the all resources selected by Rank-S. Our approach also lowers the response time %80 compared to waiting all resources.

Table 4.218 Rank-S Resource Selection NDCG QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.666	0.736	0.807	0.836	**298.86**	1617.38	1671.4	1787.88
CSUMG	0.336	0.286	0.241	0.357	**243.48**	1126.38	966.0	1787.88
CSUML	0.327	0.274	0.257	0.353	**224.20**	1273.90	778.7	1787.88
RRF	0.359	0.316	0.283	0.388	**278.86**	1309.72	1029.9	1787.88
IRF	0.277	0.240	0.223	0.295	**281.92**	1028.92	835.0	1787.88

Table 4.219 Rank-S Resource Selection SUM QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.728	0.797	0.804	0.836	**275.22**	1645.7	1668.7	1787.88
CSUMG	0.362	0.252	0.259	0.357	**353.12**	1104.5	580.2	1787.88
CSUML	0.354	0.296	0.255	0.353	**353.12**	1164.2	636.1	1787.88
RRF	0.391	0.296	0.299	0.388	**353.12**	1303.5	1315.1	1787.88
IRF	**0.300**	0.240	0.199	0.295	**353.12**	1364.9	822.2	1787.88

Results for Adaptive-k resource selection: Adaptive-k selects 72 engines on the average for 50 queries. All column in Tables 4.220 and 4.221 represents the results of the resources selected by Adaptive-k. Stopping policies among SUM-QPP values performs best for Adaptive-k resource selection algorithm. At the best case, the result quality of our approach is %4 lower than the result quality of the all resources selected by Adaptive-k. Our approach also lowers the response time %79 compared to waiting all resources.

Table 4.220 Adaptive-k Resource Selection NDCG QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.694	0.495	0.492	0.790	1628.46	**732.22**	808.10	6097.46
CSUMG	0.297	0.196	0.205	0.330	268.88	**199.52**	357.84	6097.46
CSUML	0.260	0.203	0.190	0.306	208.14	384.18	**197.32**	6097.46
RRF	0.295	0.143	0.153	0.344	301.86	**138.48**	668.58	6097.46
IRF	0.191	0.100	0.097	0.231	493.50	**249.08**	378.66	6097.46

Table 4.221 Adaptive-k Resource Selection SUM QPP Results

	NDCG@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.711	0.443	0.482	0.790	1631.80	**792.26**	850.82	6097.46
CSUMG	0.321	0.185	0.167	0.330	1088.56	295.16	**176.98**	6097.46
CSUML	0.296	0.166	0.162	0.306	1096.88	146.62	**142.72**	6097.46
RRF	0.339	0.158	0.136	0.344	1484.22	**152.92**	349.18	6097.46
IRF	0.223	0.113	0.109	0.231	1320.80	277.64	**185.50**	6097.46

4.3.2.2.2. Evaluation Results for Optimizing P@20

In this section, we give the effectiveness results using P@20 metric. The training of the machine learning models and the training of stop policies are based on the stop index with best P@20 score. The results for applying our approach are grouped under ReDDE, Rank-S and Adaptive-k resource selection algorithms.

Results for ReDDE resource selection: ReDDE selects 34.6 engines on the average for 50 queries. All column in Tables 4.222 and 4.223 represents the results of the resources selected by ReDDE. Stopping policies among SUM-QPP values performs best for ReDDE resource selection algorithm. At the best case, the result quality of

our approach is %3 lower than the result quality of the all resources selected by ReDDE. However our approach also lowers the response time %74 compared to waiting all resources.

Table 4.222 ReDDE Resource Selection NDCG QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.849	0.732	0.700	0.964	190.82	361.50	**192.2**	2054.58
CSUMG	0.508	0.383	0.378	0.556	229.84	**270.42**	343.16	2054.58
CSUML	0.485	0.354	0.346	0.545	179.08	241.52	**213.98**	2054.58
RRF	0.472	0.346	0.302	0.566	238.52	189.62	**174.66**	2054.58
IRF	0.353	0.305	0.278	0.370	256.72	**186.30**	266.52	2054.58

Table 4.223 ReDDE Resource Selection SUM QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.884	0.715	0.691	0.964	**224.72**	237.04	258.32	2054.58
CSUMG	0.540	0.384	0.339	0.556	336.06	**232.44**	289.72	2054.58
CSUML	0.530	0.382	0.350	0.545	336.06	**233.72**	288.16	2054.58
RRF	0.532	0.344	0.362	0.566	331.08	**237.32**	291.04	2054.58
IRF	0.353	0.292	0.287	0.370	331.08	**195.90**	227.38	2054.58

Results for Rank-S resource selection: Rank-S selects 26,2 engines on the average for 50 queries. All column in Tables 4.224 and 4.225 represents the results of the resources selected by Rank-S. Stopping policies among SUM-QPP values performs best for Rank-S resource selection algorithm. At the best case, the result quality of our approach is equal to the result quality of the all resources selected by Rank-S. Our approach also lowers the response time %79 compared to waiting all resources.

Table 4.224 Rank-S Resource Selection NDCG QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.756	0.713	0.743	0.903	**245.12**	877.88	1112.18	1787.88
CSUMG	0.400	0.384	0.360	0.510	**125.46**	1271.64	750.88	1787.88
CSUML	0.469	0.403	0.337	0.507	**224.20**	1026.78	657.70	1787.88
RRF	0.460	0.405	0.334	0.492	**278.86**	1345.82	817.34	1787.88
IRF	0.325	0.314	0.303	0.370	**171.12**	609.28	572.34	1787.88

Table 4.225 Rank-S Resource Selection SUM QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.853	0.642	0.722	0.903	**275.2**	808.3	1087.1	1787.88
CSUMG	0.511	0.391	0.368	0.510	**367.9**	606.4	709.6	1787.88
CSUML	0.504	0.382	0.358	0.507	**367.9**	549.4	711.2	1787.88
RRF	0.493	0.380	0.361	0.492	**367.9**	1108.7	1031.8	1787.88
IRF	0.371	0.295	0.286	0.370	**363.5**	529.2	645.5	1787.88

Results for Adaptive-k resource selection: Adaptive-k selects 72 engines on the average for 50 queries. All column in Tables 4.226 and 4.227 represents the results of the resources selected by Adaptive-k. Stopping policies among SUM-QPP values performs best for Adaptive-k resource selection algorithm. At the best case, the result quality of our approach is %2 lower than the result quality of the all resources selected by Adaptive-k. Our approach also lowers the response time %72 compared to waiting all resources.

Table 4.226 Adaptive-k Resource Selection NDCG QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.808	0.489	0.477	0.860	1628.4	**281.40**	557.7	6097.46
CSUMG	0.449	0.220	0.242	0.489	268.8	**141.88**	144.8	6097.46
CSUML	0.412	0.273	0.264	0.455	208.1	350.18	**148.4**	6097.46
RRF	0.309	0.181	0.204	0.443	**252.8**	322.72	424.9	6097.46
IRF	0.185	0.153	0.158	0.267	**48.98**	182.34	68.88	6097.46

Table 4.227 Adaptive-k Resource Selection SUM QPP Results

	P@20 Score				PLL Cost			
	SP	ML Rat	ML Diff	All	SP	ML Rat	ML Diff	All
Oracle	0.823	0.497	0.516	0.860	1631.8	**149.46**	430.2	6097.46
CSUMG	0.478	0.231	0.243	0.489	1716.4	**115.58**	140.2	6097.46
CSUML	0.442	0.233	0.238	0.455	1329.1	246.18	**111.8**	6097.46
RRF	0.435	0.161	0.177	0.443	1716.4	**47.16**	136.2	6097.46
IRF	0.259	0.133	0.153	0.267	1320.8	**29.24**	65.2	6097.46

4.3.2.3. Summary of Findings

Our key findings in this section can be summarized as follows:

- Machine Learning Models performs worse in all cases compared to using adhoc Stop Policies. The reason behind this might be the distribution of the quality of the resources varies for each query since we access them in a cost-sorted way, and the QPP difference and the QPP ratio features might not be enough to distinguish between the good and bad resource. In our first incremental query forwarding approach, the trend in QPP values between intermediate merge lists is usually decreasing as the number of resources

increases, however in the query result merging approach, there is no trend among QP values.

- Using adhoc stop policies in query result merging approach yields between 27% decrement and 11% increment in NDCG@20 score compared to merging the result of all selected engines.

- Using adhoc stop policies in query result merging approach yields between 17% decrement and %1 increment in P@20 score compared to merging the all result of all selected engines

- Gain in response time using our approach is up to 90%. The reason behind this might be the distribution of the estimated response times using posting list lengths of the search engines in the sample index. Minimum posting list lengths of engines in sample index is usually less than 500 postings but for some queries there are some engines with PLL more than 10000 postings in the sample index. This increases our gain in PLL cost with early termination by not accessing the engines with long response time.

RRF merging algorithm performs best for most cases in our experiments. In Table 4.228, the summary of the results of the RRF algorithm and NDCG-QPP score is given for both of our methods. In both methods, adhoc stop policies seems to perform better than machine learning models and at the worst case using stop policies decreases the quality of the results %18 in NDCG@20 metric compared to the results of all engines. ReDDE seems to perform better than other RS algorithms excluding Oracle and Baseline RS which use relevance judgment data to rank resources. The incremental query forwarding approach seems to perform better for good resource selection algorithms such as Oracle and Baseline RS.

Table 4.228 NDCG@20 Score Comparison of Our Methods for RRF Merging NDCG-QPP

	Incremental Query Forwarding			Incremental Result Merging			
	SP	ML Ratio	ML Diff	SP	ML Ratio	ML Diff	All
TWF-IRF	0.370	0.320	0.326	0.350	0.358	0.318	**0.403**
UISP	0.353	0.290	0.285	0.353	0.313	0.265	**0.396**
Oracle	0.631	0.653	**0.648**	0.454	0.408	0.417	0.536
Baseline	**0.556**	0.545	0.545	0.431	0.419	0.425	0.492
ReDDE	0.392	0.375	0.388	0.360	0.256	0.260	**0.436**
Rank-S	0.359	0.322	0.310	0.359	0.316	0.283	**0.386**
Adaptive-k	**0.351**	0.327	0.322	0.295	0.143	0.153	0.344

CHAPTER 5

CONCLUSION AND FUTURE WORK

Resource selection and result merging are key steps that are employed in meta-search engines. We proposed two novel methods, namely, incremental query forwarding and incremental query result merging methods, that utilize the query performance predictors to improve performance in meta-search. Using the QPP performance values on the training query set, we constructed ML models and defined ad-hoc stop policies to decide on the early termination for an unseen query. We evaluated our approaches using the combination of 7 different resource selection algorithms and 5 different result merging methods on FedWeb 2013 dataset, which contains the results of 50 queries from 157 different search engines.

In our first approach, we proposed an incremental query forwarding rather than forwarding the query to all selected resources. We forwarded the query to a subset of selected resources of size *k,* and wait for the response from them. When the results of *k* resources are retrieved, we prepared the intermediate merged lists and calculated QPP scores for the intermediate merged lists. Using the QPP scores, we decided on the early termination to stop query forwarding to rest of the resources. Our experiments revealed that adhoc stop policies performs better than ML models in FedWeb 2013 dataset. Using the stop policies, we could decrease the network resource usage up to 90% in comparison to forwarding query to all selected resources, with acceptable reductions in the effectiveness.

In our second approach, we proposed an incremental query result merging of the retrieved resources. In this approach, similar to a typical meta-search engine, we forwarded the query to all selected resources, however we started merging the results as soon as we receive the results from the resources and obtained the intermediate merge lists. By calculating the QPP score of the intermediate merge lists, we decided

whether we should wait for the results from other resources or terminate early to decrease the response time of a meta-search engine. Experiments showed that our approach can decrease the response time up to 92%, again with reasonable or sometimes no reduction in effectiveness, in comparison to waiting the response from all resources.

As a future work, we plan to apply and evaluate our incremental approaches in a cooperative setup where query processing costs can be obtained from the actual resources.

REFERENCES

[1] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. In L. Carr, D. Roure, A. Iyengar, C. Goble, and M. Dahlin, editors, *Proceedings of the 15th International Conference on World Wide Web*, pages 367-376, Edinburgh, UK, 2006. ACM.ISBN 1-59593-323-9

[2] Thomas Demeester1, Dong Nguyen2,Dolf Trieschnigg2, Chris Develder1, and Djoerd Hiemstra2. What Snippets Say About Pages in Federated Web Search , *AIRS 12*

[3] Milad Shokouhi and Luo Si. Federated search. *Foundations and Trends in Information Retrieval*, 5(1):1–102, January 2011.

[4] E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the web. *IEEE Expert*, 12(1):8-14, 1997a. ISSN 0885-9000.

[5] K. Liu, W. Meng, J. Qiu, C. Yu, V. Raghavan, Z. Wu, Y. Lu, H. He, and H. Zhao. Allinonenews: development and evaluation of a largescale news metasearch engine. In Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou, editors, In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1017-1028, Beijing, China, 2007. ISBN 978-1-59593-686-8.

[6] S. Gauch, G. Wang, and M. Gomez. ProFusion: Intelligent fusion from multiple, distributed search engines. In *Journal of Universal Computer Science*, 2(9):637-649, 1996b. ISSN 1041-4347.

[7] D. Dreilinger and A. Howe. Experiences with selecting search engines using metasearch. *ACM Transaction on Information Systems*, 15(3):195-222, 1997. ISSN 1046-8188.

[8] E. Han, G. Karypis, D. Mewhort, and K. Hatchard. Intelligent metasearch engine for knowledge management. In Kraft et al. [2003], pages 492-495. ISBN 1-58113-723-0.

[9] A. Smeaton and F. Crimmins. Using a data fusion agent for searching the WWW. In P. Enslow, M. Genesereth, and A. Patterson, editors, Selected papers from the *Sixth International Conference on World Wide Web*, Santa Clara, CA, 1997. Elsevier. Poster Session.

[10] E. Glover, S. Lawrence, W. Birmingham, and C. Giles. Architecture of a metasearch engine that supports user information needs. In *Gauch* [1999], pages 210-216. ISBN 1-58113-1461.

[11] L. Gravano, C. Chang, H. Garcia-Molina, and A. Paepcke. STARTS: Stanford proposal for internet meta-searching. In J. Peckham, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 207-218, Tucson, AZ, 1997. ISBN 0-89791-911-4.

[12] D. D'Souza, J. Thom, and J. Zobel. Collection selection for managed distributed document databases. *Information Processing and Management*, 40(3):527-546, 2004a. ISSN 0306-4573.

[13] J. Xu and J. Callan. Effective retrieval with distributed collections. In Croft et al. [1998], pages 112-120. ISBN 1-58113-015-5.

[14] J. Zobel. Collection selection via lexicon inspection. In P. Bruza, editor, *Proceedings of the Australian Document Computing Symposium*, pages 74-80, Melbourne, Australia, 1997.

[15] B. Yuwono and D. Lee. Server ranking for distributed text retrieval systems on the internet. In R. Topor and K. Tanaka, editors, *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications, volume 6 of Advanced Database Research and Development Series*, pages 41-50, Melbourne, Australia, 1997. World Scientic. ISBN 981-02-3107-5.

[16] J. Callan and M. Connell. Query-based sampling of text databases. ACM *Transactions on Information Systems*, 19(2):97-130, 2001. ISSN 1046-8188.

[17] M. Shokouhi, F. Scholer, and J. Zobel. Sample sizes for query probing in uncooperative distributed information retrieval. In X. Zhou, J. Li, H. Shen, M. Kitsuregawa, and Y. Zhang, editors, *Proceedings of Eighth Asia Pacific Web Conference*, pages 63-75, Harbin, China, 2006a. ISBN 3-540-31142-4.

[18] M. Baillie, L. Azzopardi, and F. Crestani. Adaptive query-based sampling of distributed collections. In Crestani et al. [2006], pages 316-328. ISBN 3-540-45774-7.

[19] P. Ipeirotis and L. Gravano. When one sample is not enough: improving text database selection using shrinkage. In G.Weikum, A. Konig, and S. Deiloch, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 767-778, Paris, France,2004. ISBN 1-58113-859-8.

[20] K. Liu, C. Yu, and W. Meng. Discovering the representative of a search engine. In Paques et al. [2001], pages 652-654. ISBN 1-58113-436-3.

[21] J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97-130, 2001. ISSN 1046-8188.

[22] L. Gravano, H. Garcia-Molina, and A. Tomasic. The efectiveness of GlOSS for the text database discovery problem. In R. Snodgrass and M. Winslett, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 126-137, Minneapolis, MN, 1994a. ISBN 0-89791-639-5.

[23] J. Callan. Distributed information retrieval. In B. Croft, editor, *Advances in information retrieval, Chapter 5, volume 7 of The Information Retrieval Series*, pages 127-150. Kluwer Academic Publishers, 2000. ISBN 978-0-7923-7812-9.

[24] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In Clarke et al. [2003], pages 298-305. ISBN 1-58113-646-3.

[25] L. Si and J. Callan. Unified utility maximization framework for resource selection. In D. Grossman, L. Gravano, C. Zhai, O. Herzog, and D. Evans, editors, *Proceedings of the ACM CIKM International Conference on Information and Knowledge Management*, pages 32-41, Washington, DC, 2004b. ISBN 1-58113-874-1.

[26] P. Thomas and M. Shokouhi. SUSHI: scoring scaled samples for server selection. In Allan et al. [2009], pages 419-426. ISBN 978-1-60558-483-6.

[27] M. Shokouhi. Central-rank-based collection selection in uncooperative distributed information retrieval. In G. Amati, C. Carpineto, and G. Romano, editors, *Proceedings of the 29th European Conference on Information Retrieval Research, volume 4425 of Lecture Notes in Computer Science*, pages 160-172, Rome, Italy, 2007a. Springer.

[28] J. Arguello, J. Callan, and F. Diaz. Classification-based resource selection. In Cheung et al. [2009], pages 1277-1286. ISBN 978-1-60558-512-3.

[29] Cormack, G.V., Clarke, C.L.A., Buettcher, S.: Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In: *SIGIR '09*

[30] Demeester, T., Trieschnigg, D., Nguyen, D., Hiemstra, D.: Overview of the trec 2013 federated web search track. In: *TREC*

[31] E. Fox and J. Shaw. Combination of multiple searches. In D. Harman,editor, *Proceedings of the Second Text REtrieval Conference*, pages 243-252, Gaithersburg, MD, 1993. NIST Special Publication.

[32] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning* (ICML '05). ACM, New York, NY, USA, 89-96. DOI=10.1145/1102351.1102363

[33] Fernandez, M., Vallet, D., & Castells, P. (2006). Probabilistic score normalization for rank aggregation. In *Proceedings of the 28th European Conf. on IR Research* (ECIR'06) (p. 553-556).

[34] A. M. Ozdemiray and I. S. Altingovde. Explicit search result diversification using score and rank aggregation methods. JASIST. In press.

[35] S. Cronen-Townsend and W. Bruce Croft. Quantifying Query Ambiguity. HLT'02 San Diego, CA

[36] Amati, G.; van Rijsbergen, C. Probabilistic models of Information Retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems* 20(4), pages 357–389. 2002

[37] S. Tomlinson. Robust, Web and Terabyte Retrieval with Hummingbird Search Server at TREC 2004. In *Proc. of TREC-13*, 2004.

[38] Y. Bernstein, B. Billerbeck, S. Garcia, N. Lester, F. Scholer, and J. Zobel. RMIT university at trec 2005: Terabyte and robust track. In *Proc. of TREC-14*, 2005.

[39] Y. Bernstein, B. Billerbeck, S. Garcia, N. Lester, F. Scholer, and J. Zobel. RMIT university at trec 2005: Terabyte and robust track. In *Proc. of TREC-14*, 2005.

[40] F. Diaz. Performance prediction using spatial autocorrelation. In *Proc. of SIGIR*, pages 583–590, 2007.

[41] Y. Zhou and B. Croft. Query performance prediction in web search environments. In *Proc. of SIGIR*, pages 543–550, 2007.

[42] N. Balasubramanian, G. Kumaran, and V. R. Carvalho. Predicting query performance on the web. In *Proc. of SIGIR*, pages 785–786, 2010.

[43] J. P´erez-Iglesias and L. Araujo. Standard deviation as a query hardness estimator. In *Proc. of SPIRE*, pages 207–212, 2010.

[44] R. Cummins. Predicting query performance directly from score distributions. In *Proc. of AIRS*, pages 315–326, 2011.

[45] R. Cummins, J. M. Jose, and C. O'Riordan. Improved query performance prediction using standard deviation. In *Proc. of SIGIR*, pages 1089–1090, 2011.

[46] A. Shtok, O. Kurland, D. Carmel, F. Raiber, and G. Markovits. Predicting query performance by query-drift estimation. *ACM Transactions on Information Systems*, 30(2):11, 2012.

[47] F. Raiber, O. Kurland. Query-Performance Prediction: Setting the Expectations Straight. *SIGIR'14*, July 6–11, 2014, Gold Coast, Queensland, Australia

[48] D. Dreilinger and A. Howe. Experiences with selecting search engines using metasearch. *ACM Transaction on Information Systems*, 15(3):195-222, 1997. ISSN 1046-8188.

[49] G. Markovits, A. Shtok, O. Kurland, and D. Carmel. Predicting query performance for fusion-based retrieval. In *CIKM '12*, pages 813–822. ACM, 2012.

[50] A. Broder, M. Fontura, V. Josifovski, R. Kumar, R. Motwani, S. Nabar, R. Panigrahy, A. Tomkins, and Y. Xu. Estimating corpus size via queries. In *CIKM 2006*, pages 594-603, 2006.

[51] A. Kulkarni. Shrkc: Shard rank cutoff prediction for selective search. In *SPIRE*, 2015.

[52] I. Markov and F. Crestani. Theoretical, qualitative, and quantitative analyses of small-document approaches to resource selection. *ACM TOIS*, 32(2):9, 2014.

[53] J.Arguello, J. Callan, and F. Diaz. 2009. Classification-based resource selection. In *Proceedings of the 18th ACM conference on Information and knowledge management* (CIKM '09).

[54] E. D. Buccio, I. Masiero, and M. Melucci. University of Padua at TREC 2013: federated web search track. In *Proceedings of the 22nd Text REtrieval Conference Proceedings* (TREC), 2014.

[55] K. Balog. Collection and document language models for resource selection. In *Proceedings of the 22nd Text REtrieval Conference Proceedings* (TREC), 2014.

[56] A. Mourao, F. Martins, and J. Magalhaes. Novasearch at TREC 2013 federated web search track: Experiments with rank fusion. In *TREC '13*, 2013.

[57] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); *The WEKA Data Mining Software: An Update; SIGKDD Explorations*, Volume 11, Issue 1.

[58] Bodo Billerbeck Adam Cannane Abhijit Chattaraj Nicholas Lester William Webber Hugh E. Williams John Yiannis Justin Zobel, RMIT University at TREC 2004:. In *TREC 13*, 2004 .

APPENDIX A

FEDWEB 2013 DATASET INFORMATION

Table A.1 FedWeb 2013 Data Engine Links

Engine ID	Searh Engine Name	Search Engine URL
e001	arXiv.org	http://arxiv.org
e002	CCSB	http://liinwww.ira.uka.de
e003	CERN Documents	http://cdsweb.cern.ch
e004	CiteSeerX	http://citeseerx.ist.psu.edu
e005	CiteULike	http://www.citeulike.org
e006	Economists Online	http://www.economistsonline.org
e007	eScholarship	http://escholarship.org
e008	KFUPM ePrints	http://eprints.kfupm.edu.sa
e009	MPRA	http://mpra.ub.uni-muenchen.de
e010	MS Academic	http://academic.research.microsoft.com
e011	Nature	http://www.nature.com
e012	Organic Eprints	http://orgprints.org
e013	SpringerLink	http://www.springerlink.com
e014	U. Twente	http://doc.utwente.nl
e015	UAB Digital	http://ddd.uab.cat
e016	UQ eSpace	http://espace.library.uq.edu.au
e017	PubMed	http://pubmed.gov
e018	LastFM	http://www.last.fm/
e019	LYRICSnMUSIC	http://www.lyricsnmusic.com/
e020	Comedy Central	http://www.comedycentral.com/
e021	Dailymotion	http://www.dailymotion.com/
e022	YouTube	http://www.youtube.com
e023	Google Blogs	http://www.google.com/blogsearch
e024	LinkedIn Blog	http://blog.linkedin.com
e025	Tumblr	http://www.tumblr.com
e026	WordPress	http://en.search.wordpress.com
e027	Columbus Library	http://www.columbuslibrary.org/
e028	Goodreads	http://www.goodreads.com/
e029	Google Books	http://books.google.com

Table A.1 FedWeb 2013 Data Engine Links (continued)

e030	NCSU Library	http://www.lib.ncsu.edu
e032	IMDb	http://www.imdb.com
e033	Wikibooks	http://en.wikibooks.org
e034	Wikipedia	http://en.wikipedia.org
e036	Wikispecies	http://species.wikimedia.org
e037	Wiktionary	http://en.wiktionary.org
e038	E! Online	http://www.eonline.com
e039	Entertainment Weekly	http://www.ew.com/
e041	TMZ	http://www.tmz.com/
e042	The Sun	http://www.thesun.co.uk/
e043	Addicting games	http://www.addictinggames.com/
e044	Amorgames	http://armorgames.com/
e045	Crazy monkey games	http://www.crazymonkeygames.com/
e047	GameNode	http://www.gamenode.com
e048	Games.com	http://games.com
e049	Miniclip	http://www.miniclip.com/games/en/
e050	About.com	http://www.about.com/
e052	Ask	http://www.ask.com
e055	CMU ClueWeb	http://lemurproject.org/clueweb09/
e057	Gigablast	http://www.gigablast.com
e062	Baidu	http://www.baidu.com
e063	Centers for Disease Control and Prevention	http://www.cdc.gov/
e064	Family Practice notebook	http://www.fpnotebook.com/
e065	Health Finder	http://www.healthfinder.gov/
e066	HealthCentral	http://www.healthcentral.com/
e067	HealthLine	http://www.healthline.com/
e068	Healthlinks.net	http://www.healthlinks.net/
e070	Mayo Clinic	http://www.mayoclinic.org/
e071	MedicineNet	http://www.medicinenet.com/
e072	MedlinePlus	http://www.nlm.nih.gov/medlineplus/
e075	University of Iowa hospitals and clinics	http://www.uihealthcare.org/
e076	WebMD	http://www.webmd.com/
e077	Glassdoor	http://www.glassdoor.com/
e078	Jobsite	http://www.jobsite.co.uk
e079	LinkedIn Jobs	http://www.linkedin.com
e080	Simply Hired	http://www.simplyhired.com
e081	USAJobs	http://www.usajobs.gov
e082	Comedy Central Jokes.com	http://www.jokes.com/
e083	Kickass jokes	http://kickasshumor.com/search/

Table A.1 FedWeb 2013 Data Engine Links (continued)

e085	Cartoon Network	http://www.cartoonnetwork.com/
e086	Disney Family	http://family.go.com
e087	Factmonster	http://www.factmonster.com/
e088	Kidrex	http://www.kidrex.org/
e089	KidsClicks!	http://kidsclick.org/
e090	Nick jr	http://www.nickjr.com/
e091	Nickelodeon	http://www.nick.com/
e092	OER Commons	http://www.oercommons.org
e093	Quintura Kids	http://affiliates.quintura.com/mainkids/
e095	Foursquare	https://foursquare.com/
e098	BBC	http://www.bbc.co.uk
e099	Bing News	http://www.bing.com
e100	Chronicling America	http://chroniclingamerica.loc.gov/
e101	CNN	http://www.cnn.com
e102	Forbes	http://www.forbes.com/search/
e103	Google News	http://news.google.com
e104	JSOnline	http://www.jsonline.com/
e106	Slate	http://www.slate.com
e107	The Guardian	http://www.guardian.co.uk/
e108	The Street	http://www.thestreet.com/
e109	Washington post	http://www.washingtonpost.com/
e110	HNSearch	http://www.hnsearch.com/
e111	Slashdot	http://slashdot.org
e112	The Register	http://search.theregister.co.uk
e113	DeviantArt	http://browse.deviantart.com
e114	Flickr	http://www.flickr.com
e115	Fotolia	http://www.fotolia.com
e117	Getty Images	http://www.gettyimages.com
e118	IconFinder	http://www.iconfinder.com/
e119	NYPL Gallery	http://digitalgallery.nypl.org
e120	OpenClipArt	http://openclipart.org
e121	Photobucket	http://photobucket.com
e122	Picasa	http://picasaweb.google.com/lh/explore
e123	Picsearch	http://www.picsearch.com/
e124	Wikimedia	http://commons.wikimedia.org
e126	Funny or Die	http://www.funnyordie.com
e127	4Shared	http://search.4shared.com
e128	AllExperts	http://www.allexperts.com/
e129	Answers.com	http://wiki.answers.com

Table A.1 FedWeb 2013 Data Engine Links (continued)

e130	Chacha	http://www.chacha.com/
e131	StackOverflow	http://stackoverflow.com
e132	Yahoo Answers	http://answers.yahoo.com
e133	MetaOptimize	http://metaoptimize.com/qa/
e134	HowStuffWorks	http://www.howstuffworks.com/
e135	AllRecipes	http://allrecipes.com
e136	Cooking.com	http://www.cooking.com/recipes-and-more
e137	Food Network	http://www.foodnetwork.com/
e138	Food.com	http://www.food.com/
e139	Meals.com	http://www.meals.com/
e140	Amazon	http://www.amazon.com
e141	ASOS	http://www.asos.com/
e142	Craigslist	http://www.craigslist.org/
e143	eBay	http://www.ebay.com
e144	Overstock	http://www.overstock.com/
e145	Powell's	http://www.powells.com
e146	Pronto	http://www.pronto.com/
e147	Target	http://www.target.com/
e148	Yahoo! Shopping	http://shopping.yahoo.com/
e152	Myspace	http://api.myspace.com
e153	Reddit	http://www.reddit.com/
e154	Tweepz	http://tweepz.com
e156	Cnet	http://download.cnet.com
e157	GitHub	http://github.com/search
e158	SourceForge	http://sourceforge.net
e159	bleacher report	http://bleacherreport.com/
e160	ESPN	http://search.espn.go.com
e161	Fox Sports	http://multimedia.foxsports.com
e162	NBA	http://www.nba.com/
e163	NHL	http://www.nhl.com/
e164	SB nation	http://www.sbnation.com/
e165	Sporting news	http://aol.sportingnews.com/
e166	WWE	http://www.wwe.com/
e167	Ars Technica	http://arstechnica.com/
e168	CNET	http://www.cnet.com/
e169	Technet	http://social.technet.microsoft.com
e170	Technorati	http://www.technorati.com
e171	TechRepublic	http://www.techrepublic.com
e172	TripAdvisor	http://www.tripadvisor.com/

Table A.1 FedWeb 2013 Data Engine Links (continued)

e173	Wiki Travel	http://wikitravel.org/en/Main_Page
e174	5min.com	http://www.5min.com
e175	AOL Video	http://www.aol.com
e176	Google Videos	http://www.google.com/videohp
e178	MeFeedia	http://www.mefeedia.com
e179	Metacafe	http://www.metacafe.com
e181	National geographic	http://www.nationalgeographic.com/
e182	Veoh	http://www.veoh.com/
e184	Vimeo	http://vimeo.com
e185	Yahoo Screen	http://screen.yahoo.com
e200	BigWeb	no url

Table A.2 Queries with Relevance Judgment Data

Query ID	Query	Query ID	Query
7001	LHC collision publications	7090	eurovision 2012
7003	Male circumcision	7094	calculate inertia sphere
7004	z-machine	7096	touchpad scroll dell latitude
7007	Allen Ginsberg Howl review	7097	best dum blonds
7009	linkedin engineering	7099	lecture manova
7018	audiobook Raymond e feist	7103	cystic fibrosis treatment
7025	M/G/1 queue	7109	best place to eat pho in new york
7030	Lyrics Bangarang	7115	pittsburgh steelers news
7033	Porto	7124	yves saint laurent boots
7034	sony vaio laptop	7127	which cities surround long beach ca
7039	import .csv excel	7129	avg home edition
7040	vom fass gent	7132	massachusetts general hospital jobs
7042	bmw c1	7145	why do cats purr
7046	tuning fork	7209	crab dip appetizer
7047	Dewar flask	7258	swahili dishes
7056	ROADM	7348	map of the united states
7067	used kindle	7404	kobe bryant
7068	Speech and Language Processing	7406	does my child have adhd
7069	Eames chair	7407	kim kardashian pregnant
7075	zimerman chopin ballade	7415	most anticipated games of 2013
7076	Bouguereau	7465	xman sequel
7080	lord of the rings hobbits theme	7485	bachelor party jokes
7084	Burn after reading review	7504	leiden schools
7087	Jonathan Kreisberg discography	7505	ethnic myanmar
7089	varese ionisation	7506	I touch myself singer dead

APPENDIX B

RESOURCE SELECTION PARAMETER ANALYSIS

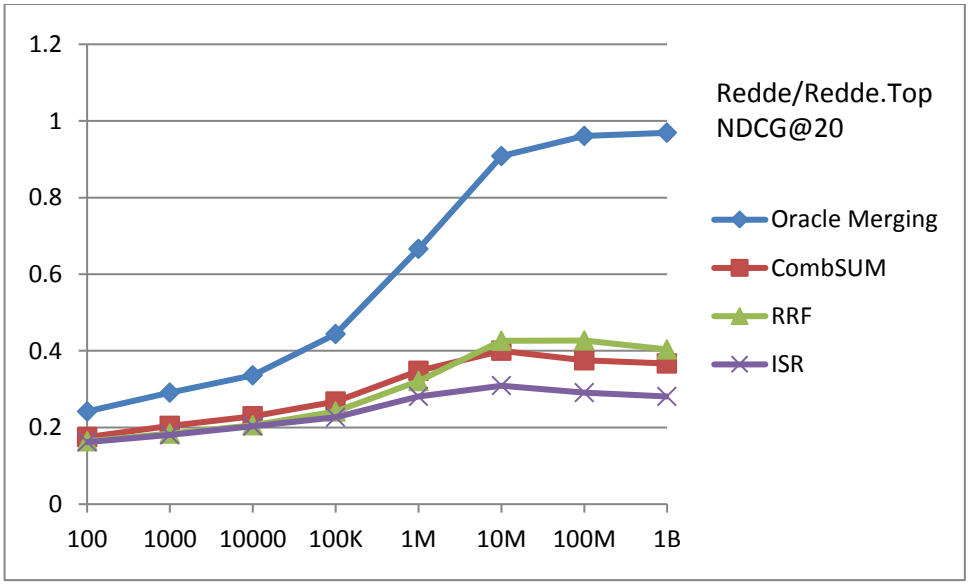

Figure B.1 ReDDE NDCG@20 score for different *L* thresholds.

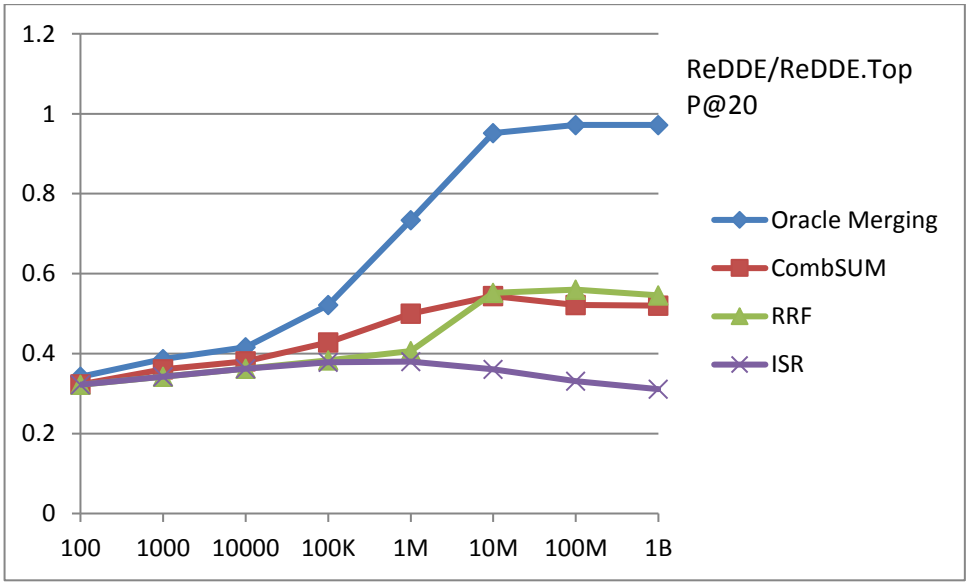

Figure B.2 ReDDE P@20 score for different *L* thresholds

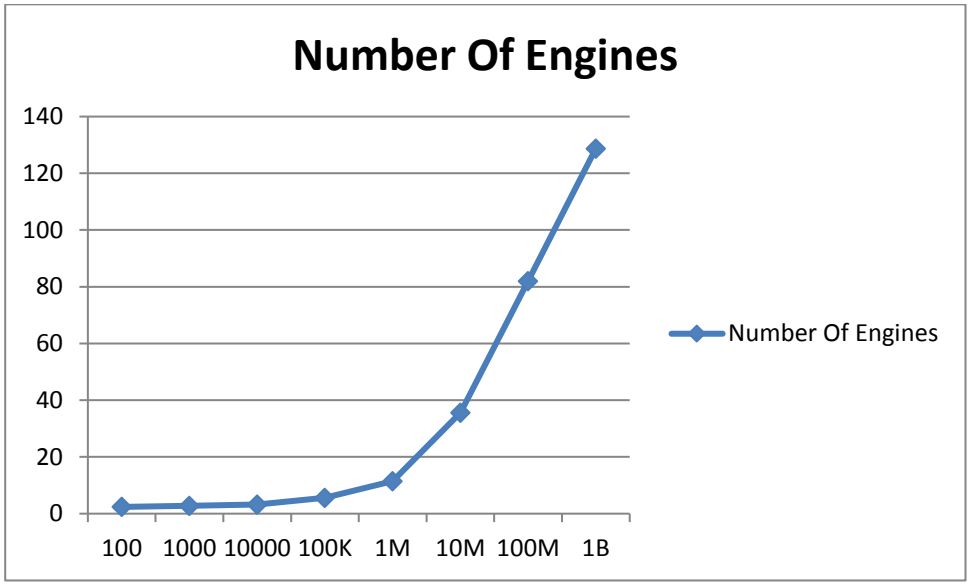

Figure B.3 ReDDE average number of selected engines for different *L* thresholds

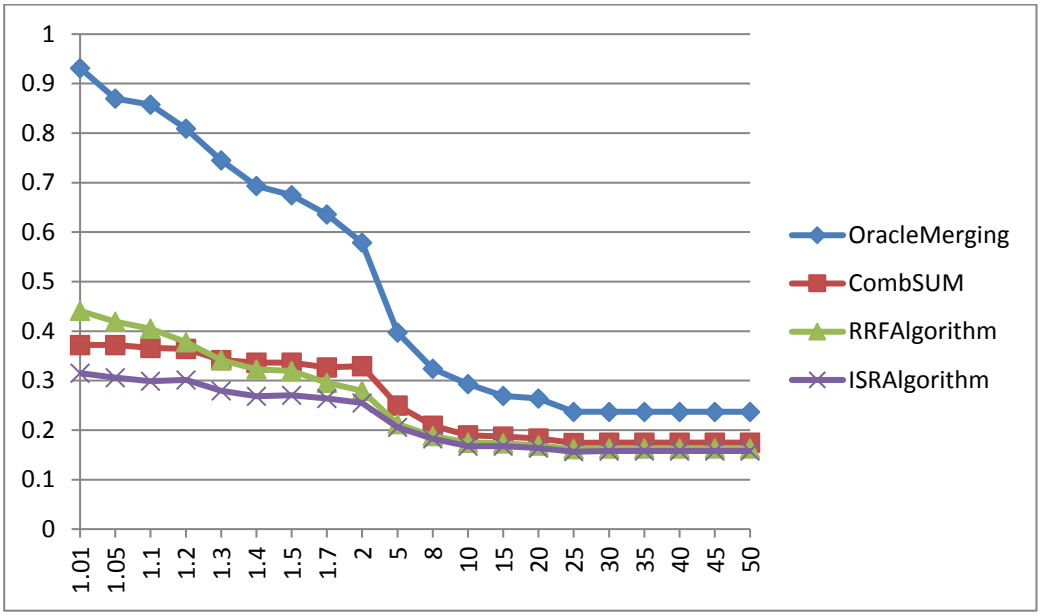

Figure B.4 Rank-S average NDCG@20 for different *B* values

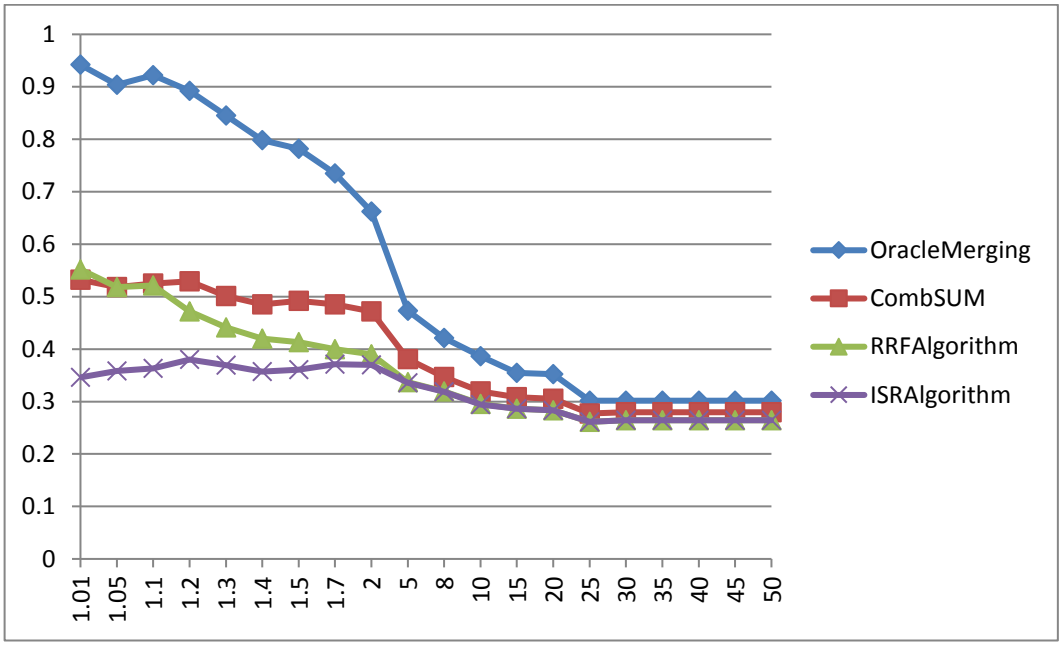

Figure B.5 Rank-S average P@20 for different *B* values

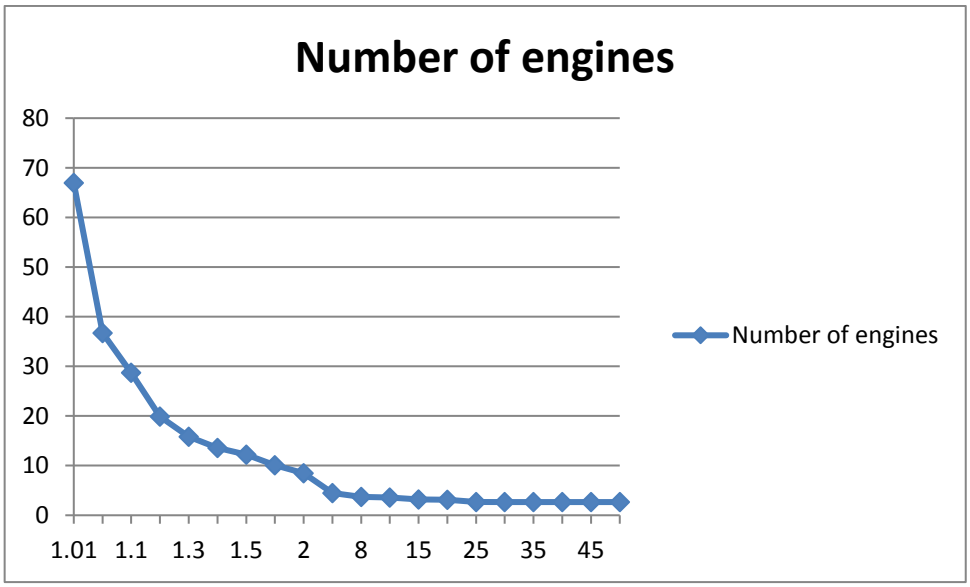

Figure B.6 Rank-S average number of selected engines for different *B* values