COMPARING OFFLINE AND ONLINE STATISTICS ESTIMATION FOR TEXT RETRIEVAL FROM OVERLAPPED COLLECTIONS

by

Bhaumik Chokshi

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

December 2007

COMPARING OFFLINE AND ONLINE STATISTICS ESTIMATION FOR TEXT RETRIEVAL FROM OVERLAPPED COLLECTIONS

by

Bhaumik Chokshi

has been approved

September 2007

ABSTRACT

In an environment of distributed text collections, the first step in the information retrieval process is to identify which of all available collections are more relevant to a given query and should thus be accessed to answer the query. Collection selection is difficult due to the varying relevance of sources as well as the overlap between these sources. Some of the previous collection selection methods have considered relevance of the collections but have ignored overlap among collections. They thus make the unrealistic assumption that the collections are all effectively disjoint. Overlap estimation can be done in two ways - offline or online. In this thesis, the main objective is to compare these two approaches for estimating statistics. One of the existing approaches(e.g., COSCO) uses offline approach to store the statistics for frequent item sets. It uses these statistics to estimate statistics for the user query. In this thesis, ROSCO is presented, which uses sample based online approach to estimate the overlap among collections for a given query. In addition to that, COSCO and ROSCO are compared with ReDDE(which does not consider overlap) under a variety of scenarios. The experiments show that ROSCO is able to outperform existing methods by 8-10% in presence of overlap among collections.

To my grandparents and parents for their love and support.

ACKNOWLEDGMENTS

I consider myself fortunate to have Dr. Subbarao Kambhampati as my advisor and would like to thank him sincerely for the opportunity to work under his guidance. He has not only helped me in the development of ideas and solutions, but also in the careful presentation of them. I would like to thank the members of my thesis committee, Dr. Yi Chen and Dr. Hasan Davulcu for their support throughout this research.

I am thankful to my colleagues at DB-Yochan, Hemal Khatri, Jianchun Fan, Garrett Wolf, Aravind Kalavagattu and Raju Balakrishnan for their constructive criticism and discussions. I would also like to thank the entire Planning Group at Yochan for their support and friendship; also, Luis Tari for providing me TREC Genomics data to run my experiments.

I am very grateful to my grandparents and parents, without whose support and blessings I would not be the person I am today.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Background

In multi-source information retrieval problem, searching every information source is not efficient. The retrieval system must choose which collection or subset of collections to call to answer a given query. This process is referred to as collection selection. The problem is important because redundant or irrelevant calls are expensive in many respects: in terms of query execution time, quality of results, post-query processing , network load, source load etc. As the number of collections increase, effective collection selection becomes essential for the performance of the overall retrieval system.

Figure 1 shows the general architecture of a multi-collection information retrieval system. In addition to collection selection component, the system also requires two other components: query execution and result merging. The query execution component is responsible for sending the user query out to the collections selected in the first phase and effectively retrieving the results from the collections. The purpose of the result merging component is to process the list of results obtained in the second phase in order to return the clean list of results to the user.

Most existing approaches for collection selection try to create a sample for each collection based on term and document frequency information, and then use that information at query-time to determine which collections are the most relevant for the incoming query (c.f. [7, 19, 20]) This general strategy works fairly well when the collections do not overlap. Perhaps not surprisingly, most of the published evaluations of collection selection focus on disjoint collections [19]. However, many real world collections have significant overlap. For example, multiple bibliography collections (e.g. ACMDL, IEEE XPlore, DBLP etc.) may store some of the same papers, and multiple news archives (e.g. New York Times, Washington Post etc.) may store very similar news stories. The approaches that fail to take into account overlap between collections when determining their collection order, may decide to call a collection which has no new documents when considering the documents which have already been retrieved at that point in time (e.g. in the case of two mirror collections). This leads to significant loss of performance in query processing. In this kind of scenario, the objective is to design a system that accesses collections in the order of estimated number of relevant and *new* (previously unseen) results they are likely to provide. To do so, the system must be capable of making two types of predictions:

- How likely is it that a given collection has documents relevant to the query
- Whether a collection will provide novel results given the ones already selected.

The intent here is to be able to determine, for a given user query, which collections are more relevant (i.e. which collections contain the most relevant results) and which set of collections is most likely to offer the largest variety of results (i.e. which collections are likely to have least overlap among their relevant results). Intuitively, one would want to call the most relevant collection first, and then iteratively choose the most relevant remaining collections that have most novel results compared to the already selected collection(s). Here as we assume that the sources are autonomous, we cannot directly obtain the coverage and overlap statistics. We have to obtain these estimates in indirect way by query probing. We also do not deal with the dynamism of sources (i.e., we do not deal with update of the sources). Coverage and overlap

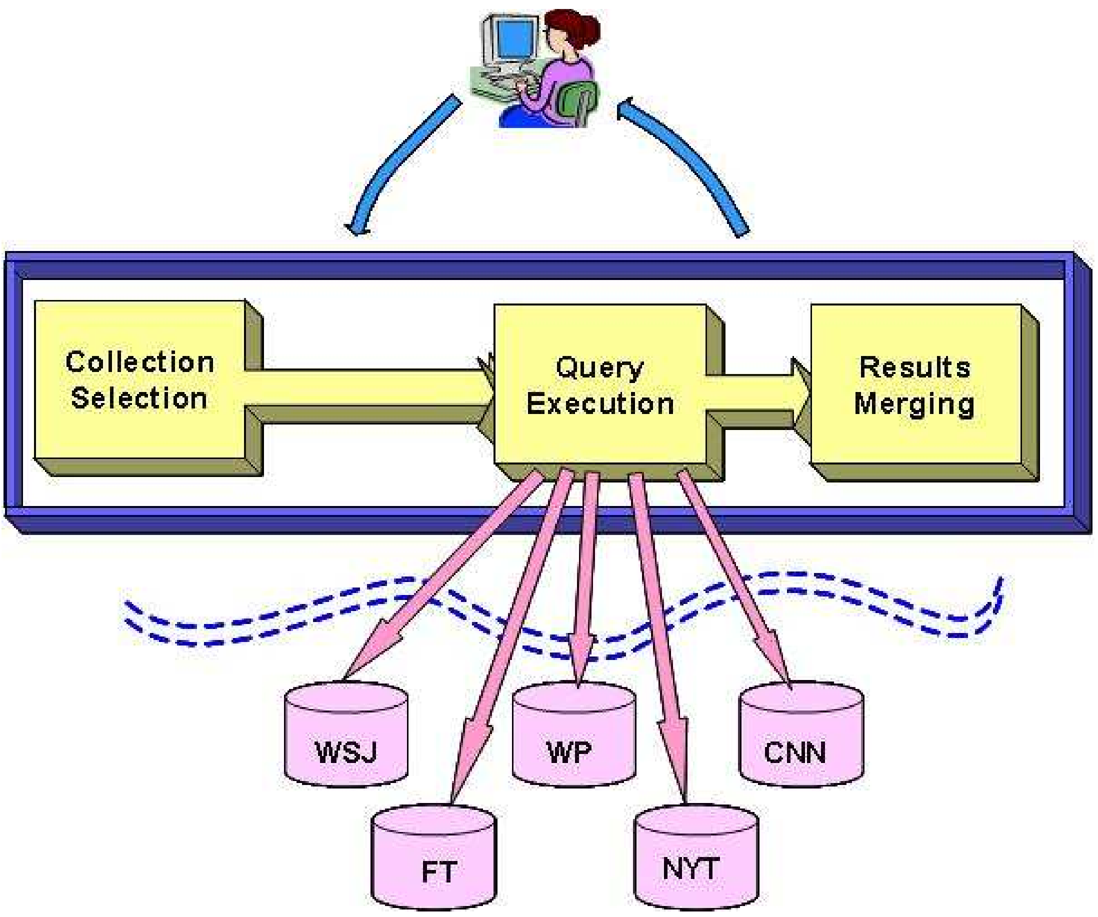

Fig. 1. Multi-collection information retrieval system

statistics estimation can be done in two ways. These estimates can be computed and stored for frequently used keyword sets(offline approach) or they can be computed from the samples of the collection at run time(online approach). One of the previous approaches(e.g., COSCO [13]) stores the coverage and overlap statistics w.r.t. frequent item sets mined from query log. At runtime, it maps the incoming query to these frequent item sets to estimate coverage and overlap statistics for the query. Statistics estimated based on frequent item sets can lead to an estimate that is different from the actual statistics for a particular query. Collection selection algorithm like ReDDE [20] estimates coverage statistics from sample at runtime which are query specific. These samples are created by query based sampling. However, ReDDE does not take overlap into account. In addition to estimating coverage statistics from these samples, overlap statistics can also be estimated from the samples for a given query at runtime. Major contributions of this thesis are:

- ROSCO, an online approach for collection selection which considers overlap among collections.
- Comparison of offline and online approach for statistics estimation for text retrieval from overlapping collections.

1.2. Collection Selection Problem

The text collection selection problem that we address in this work can be stated formally as follows. Given a set S_n of n text collections with unknown and possibly overlapping document contents, a keyword query Q, and two integers c and k, pick a subset S_c of size c from S_n such that accessing the collections in S_c will result in the highest percentage recall of top-k relevant documents for the query Q. The set of top-k relevant documents is taken to be the k most relevant documents that would have been returned if the query Q was applied to a single collection that contained the union of all documents in S_n. We denote this set $\mathcal{DK}$. If the collections in S_c each return the document sets D_i in response to the query, then the percentage recall provided by S_c is defined as:

$$R^*_{S_c} = 100 \times \frac{|(\cup_i D_i) \cap \mathcal{DK}|}{k} \tag{1.1}$$

There are several points worth noting about this formulation:

First, we note that the effectiveness of a collection selection approach is measured against the retrieval performance that could have been achieved if the query was processed against a single database that contained the union of all the collections.

Second, we note that the formula considers the intersection between the union of all results D_i and the set of top-k results $\mathcal{DK}$. Thus, returning the same results multiple times (as might be done when the collections are overlapping) doesn't increase the percentage recall R^*.

Finally we note that the percentage recall, as defined above, is *normative.* To optimize with respect to R^*, in practice we need to estimate statistics about the distribution of relevant documents across collections. Such estimates are often made through statistical sampling and representations of the collections.

Various methods in the literature focus on different approximations of R^* (see Section 2). Most existing methods assume that the collections are non-overlapping (i.e. disjoint), i.e., $|(\cup_i D_i)|$ is equal to $\sum_i |D_i|$, and thus focus exclusively on relevance.

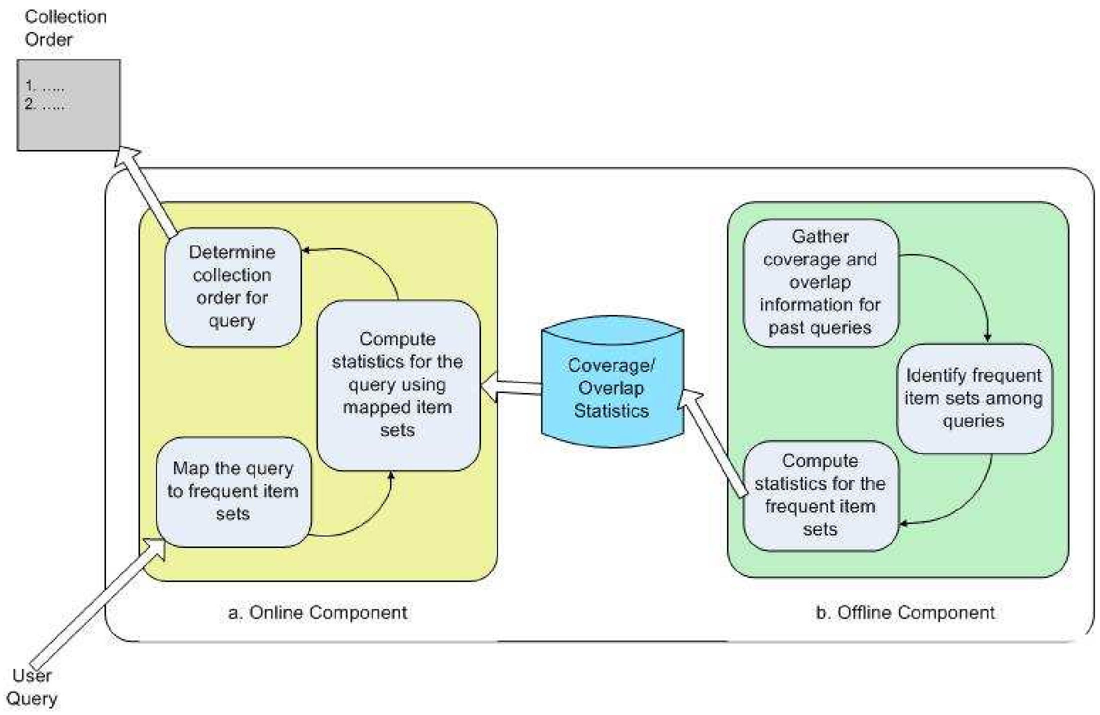
Collection Order
1. .....
2. .....
Determine collection order for query
Compute statistics for the query using mapped item sets
Map the query to frequent item sets
Coverage/ Overlap Statistics
Gather coverage and overlap information for past queries
Identify frequent item sets among queries
Compute statistics for the frequent item sets
a. Online Component
b. Offline Component
User Query

Fig. 2. Architecture of COSCO, a collection selection system.

In this work, we are interested in relaxing this assumption. In terms of our discussion above, ROSCO and COSCO account for the overlap between collections (i.e., they recognize that $|(\cup_i D_i)|$ may not be equal to $\sum_i |D_i|$).

1.3. Overview of the Challenges Involved and Proposed Solution

As described earlier, we want to compare two methods ROSCO and COSCO for collection selection. COSCO is described in [13]. Its schematic architecture is shown in figure 2. Here the challenges of text retrieval from overlapped collections and the approach taken by ROSCO is described. ROSCO builds on ReDDE [20], a state of the art relevance-based collection selection algorithm. Figure 3 shows the schematic architecture of ROSCO. Like ReDDE [20], ROSCO uses query-based

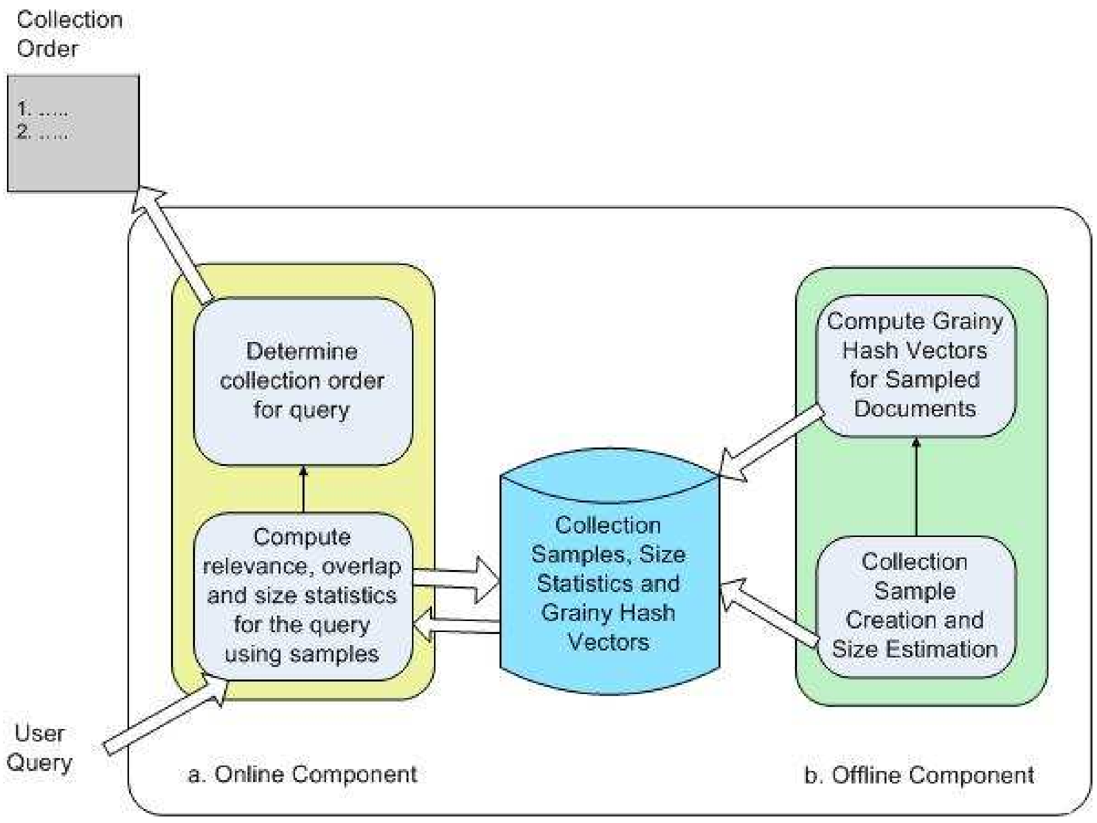

Fig. 3. Architecture of ROSCO our collection selection system.

random sampling of collections, to estimate their relevance with respect to a given query. Specifically, it builds a sample of each collection via query-based sampling, and uses such a sample to estimate the size of the collection and provide a basis upon which to estimate the relevance of a collection with respect to a query. The main extension in ROSCO is that it uses overlap statistics.

The challenges lie in effectively defining, efficiently computing, gathering, and then adequately using the overlap information. In ROSCO we compute overlap statistics from the samples built via query-based random sampling. The overlap statistics are computed from a sample of each collection at run time. When a new query comes, the documents are obtained for the query from the stored sample of each collection. Overlap statistics are then computed using the documents retrieved.

Once we have overlap statistics, the next challenge is appropriately combining them with relevance estimates. For this purpose, ROSCO uses the notion of residual relevance of a collection, given the set of collections that have already been selected. We will show how residual relevance is defined and computed using the relevance estimates (*a la* ReDDE) and overlap statistics.

We present a systematic evaluation for comparison of ROSCO and COSCO as well as their comparison with ReDDE which does not consider overlap. Our evaluation is done on documents from TREC 2005 genomics corpus. From this corpus, we formed testbeds with different amount of overlap among collections. Using these testbeds, we present a detailed experimental evaluation of these methods. Our experiments demonstrate that ROSCO outperforms the existing methods in the presence of overlap among collections by 8-10%.

Organization: The rest of the thesis is organized as follows. Existing work related to the particular problems presented above is discussed in chapter 2. The challenges in defining and estimating overlap statistics are described in chapter 3. The approach used by ROSCO for computing and using collection overlap statistics is described in chapter 4. In this chapter, we discuss how the overlap statistics are integrated into the relevance-based approach used by ReDDE. The main point of departure here is that once the first collection is chosen based on relevance, overlap statistics are used to estimate *residual relevance* of the remaining collections. The experimental setup and results are described in chapter 5. Finally,we conclude in chapter 6.

CHAPTER 2

RELATED WORK

Several approaches have been taken to solve the collection selection problem, all of which can be seen as aiming at maximizing some approximation of percentage recall R^* as defined in Equation 1. As mentioned by Powell and French [19] in their systematic comparison of collection selection algorithms, the main idea in existing systems is to try to create a representative for each collection based on term and document frequency information, and then use that information at query-time to determine which collections are most promising for the incoming query. This is the case for gGlOSS [12], CORI [7], and many other approaches [14–16]. In their survey, Powell and French [19] show that CORI is among the most effective of these approaches. CORI tends to be sensitive to the variation in the size of the collections–picking larger collections over smaller ones. Several later systems (e.g. [18, 20]) aim to deal with these limitations. One of these, ReDDE is used as the baseline in our work. ReDDE seeks to select the collections that give top-k documents that would have been returned by the same query had it been run on the union of all collections. This is a stronger form of relevance based ranking.

Most of these methods seek to approximate a relevance based ranking of the collections and assume that the collections are all non-overlapping. In contrast, ROSCO and COSCO methods explicitly take collection overlaps into account. As we mentioned, ROSCO adapts ReDDE techniques for relevance estimation, and extends them to consider collection overlap.

The work by Zhang *et. al.* [21] focuses on filtering redundant (overlapping) document results from a retrieved document stream. Like us, they too argue that considerations of document relevance and document redundancy (overlap) need to be independently modeled and combined for effective retrieval. Our work is complementary to theirs in that we focus on the "collection selection" phase and thus can effectively reduce the number of redundant results in the retrieved document stream (thus reducing the need for filtering). The work of Bender *et. al.* [5] also argued for using overlap statistics to collection selection. Their focus is however on peer to peer scenarios, and on efficiently disseminating the statistics between peers using bloom filters.

The presence of overlap among text collections has also been recognized by others. The work by Bernstein and Zobel [4] focuses on finding content-equivalent documents. They found that 16.6% of the documents in runs submitted to the TREC 2004 terabyte track were redundant. From these statistics it seems that the redundant documents can have a significant effect on search experience.

Using coverage and overlap statistics for source selection has been explored by Nie and Kambhampati [17] in the context of relational data sources. Our work, while inspired by theirs, differs in many significant ways. Their approach addresses the relational data model, in which overlap can be identified among tuples in a much more straightforward way.

COSCO [13] also addresses the problem of overlap aware text collection selection. It uses offline approach to determine the overlap among the collections. It stores coverage and overlap statistics w.r.t. frequent item sets. It uses these statistics to determine the collection order. Estimating overlap statistics from samples is another approach(online approach) which is more query specific. ROSCO uses the samples created from the collections to get the overlap statistics for a particular query. However, success of online approach depends on the quality of the samples created by query based sampling. We compare COSCO and ROSCO to understand the tradeoffs between offline and online approach.

CHAPTER 3

CHALLENGES IN DEFINING AND ESTIMATING COLLECTION OVERLAP

Given a keyword query Q, and a set of collections $C_1, C_2 \cdots C_{i-1}$ that have already been selected for access, our aim is to estimate the amount of overlap a collection C_i is expected to have with the results already returned by $C_1 \cdots C_{i-1}$. Answering this in general requires estimates of overlap between C_i and the already accessed collections with respect to the query Q. The overlap between collections can be defined in terms of exact duplicates as well as near duplicates. The duplicate determination technique should also be efficient in terms of processing time. In the following, we discuss these challenges, and our solutions.

Need for Query Specific Overlap: We start by noting that the overlap between two collections needs to be characterized in the context of specific queries. It is possible for two collections to have very low overlap, when they are taken as a whole, but still have a significant overlap in terms of their results to a specific query. Consequently, in our work, we use query-specific overlap between collections. We estimate overlap by evaluating the current query on the sample of the collections.

Overlap assessment online vs. offline Another issue is how much processing do we want to do online. If we want to do less processing online then we can compute and store coverage and overlap statistics for the queries offline. But we cannot store statistics for each query. So we may have to store only statistics for some general queries. This approach is taken by COSCO [13], a collection selection system which takes overlap into account. This approach can have a problem of approximation in terms of item sets. If we do online processing then we can estimate overlap statistics on the sampled documents. This incurs online processing but can be better in terms of estimates. The online approach is taken by ROSCO, the collection selection method presented in this thesis.

Overlap in terms of exact duplicate vs. near duplicate documents: Third issue is whether the overlap between collections is defined in terms of exact duplicates or in terms of near duplicate documents. Both types of overlap definitions are appropriate depending on the context. For example, in bibliography collections, where the same article may be present in multiple collections, overlap is best defined in terms of exact duplicates. In contrast, in the context of news collections, where "similar" but not identical news stories are reported by multiple news sources, overlap is best defined in terms of near duplicate documents.

Assessing overlap in terms of exact duplicates is reasonably straightforward – we take the intersection of the set of documents returned by individual collections in response to a query (c.f. [17]). Specifically, if $\mathcal{R}_iq$ and $\mathcal{R}_jq$ are the result sets from two collections C_i and C_j for a keyword query q, then, the overlap $ovlp_q(C_i, C_j)$ is defined as $|\mathcal{R}_iq \cap \mathcal{R}_jq|$.

It is trickier to define overlap so it takes into account near duplicate rather than just exact duplicate documents. There are two issues here: (i) how do we determine near duplicate documents and (ii) how do we quickly determine near duplicate documents. Bernstein et. al. [6] have shown a feature Grainy Hash Vector(GHV) to determine near duplicate documents. It uses the minimal-chunk sampling technique described by Fetterly et. al. [11]. Minimal-chunk sampling relies on the availability of a class of hash functions that are min-wise independent. Min-wise independence states that the class of permutations is unbiased with respect to the identity of the first element in the permutation. In the context of hashing, it means that any value in an arbitrary set has an equal probability of hashing to a value that is lowest in the set. They have shown that determining near duplicates using GHV is efficient. The details about GHV are described in detail in Chapter 4.

ROSCO supports overlap assessments in terms of both the exact duplicates and near duplicates, and the rest of the development in the thesis is largely independent of which overlap assessment is chosen. We should mention however that in our evaluations we used exact duplicate based overlap assessments, as these were most appropriate for our testbeds comprising TREC genomics documents.

CHAPTER 4

ROSCO

As described in section 1.1, we want to understand the tradeoff between the offline and the online approaches. The offline approach COSCO is described in [13]. The online approach ROSCO is presented here.

4.1. General Approach

The work described here addresses the problem of collection selection in the presence of overlap among collections. ROSCO is based on the collection selection algorithm ReDDE [20]. ReDDE creates collection samples through query based sampling. ROSCO considers overlap among collections during collection selection as well. The system is composed of an offline component which estimates collection size and also compute Grainy Hash Vectors for the sampled documents. The online component determines the collection ranking for a new incoming query at runtime.

4.2. Offline Component

The purpose of the offline component is to create samples of the collections and also compute size estimates and Grainy Hash Vectors which can be used by the online component. More precisely the offline component performs three functions. First it creates collection samples through query based sampling. Then collection size estimates are made using these samples. The offline component also computes Grainy Hash Vectors for sampled documents which can be used by the online component to estimate amount of overlap among collections.

4.2.1. Collection Representation through Query Based Sampling

To construct a sampled representation of each collection (for use in relevance judgements), a number of random queries (usually chosen from a training set) are sent to each collection and a portion of the results are kept for the sample. The queries that are chosen can easily be randomly picked from the training queries. It has been shown that a relatively small number of queries is required to obtain an accurate representation of each collection [3, 9]. A union of all the samples is maintained. An inverted index is built for each collection sample to provide single source text retrieval from the sample. Once we have the inverted indices we need not store the actual documents in the samples.

4.2.2. Estimating Collection Size

Once a sample from each collection is available, collection size estimates are made. ReDDE uses the sample-resample method [20] to estimate collection size. This involves sending a query to both the collection and its (random) sample, and using the ratio of the cardinalities of the result sets, and the (known) size of the collection sample to estimate the size of the actual collection. It can be formulated as: Let $C_i.EstimatedSize$ be the estimated size of collection C_i, $S_i.Size$ be the size of sample S_i, d_{C_i} and d_{S_i} be the number of documents returned by collection C_i and sample S_i respectively for randomly sent queries. Then

$$C_i.EstimatedSize = average(\frac{d_{C_i}}{d_{S_i}}) \times S_i.Size$$

These size estimates are stored for each collection and for the union of the collections. The estimates are used for normalization purposes at runtime.

4.2.3. Grainy Hash Vector Computation

As described in Section 3, Grainy Hash Vectors can be used to efficiently determine near duplicates. As described by Bernstein et. al. [6] GHV is a derivation of the minimal-chunk sampling technique. This technique relies on the availability of a class of hash functions that are min-wise independent. Min-wise independence states that the class of permutations is unbiased with respect to identity of the first element in permutation. In the context of hashing, it means that any value in an arbitrary set has an equal probability of hashing to a value that is lowest in the set. A GHV of n bits consists of ρ w-bit min-hashes and represented as follows:

$\rho(\text{n,w}) = \lfloor \frac{n}{w} \rfloor$

Each of the w-bit hashes is produced by a separate hash function. Specifically, a family of hash functions is defined by a single algorithm parameterized by some value(seed). A particular function in the family is thus defined by the parameter passed to a general algorithm. Thus the set of ρ min-wise independent hash functions are represented by a single hash function and an array of ρ independently chosen random seeds. The set of words in a document is passed through each of the ρ hash functions and the minimal hash under each permutation is stored. The min-wise independence property of the hash function means that, for two documents with resemblance r, the probability of the hash value in any given position on their corresponding vectors is independently r.

Clearly, generating random permutations over a large set of words and storing them to compute min-wise independent hashes is not feasible. Instead a set of independent random seed values, one for each hash function is generated and each word in a document is mapped to a hash value computed using the Id of the word and the seed value.

As described by Bernstein et. al., a 64-bit GHV with $w = 2$ is good enough to obtain reasonable near duplicates. There are 32 2-bit hashes that are merged into the vector for a 64-bit GHV with $w = 2$. For small values of w, there is a significant probability of collision between the outputs of different minimal hashes. On the other hand, using large values for w may be computationally expensive and reduces number of hashes per vector. Therefore GHV initially uses minimal sampling to produce ρ 32-bit hashes for each document. Then the least significant 2-bits of each value are used to give the GHV.

As near duplicates are determined for sampled documents, the GHVs are computed and stored for sampled documents.

4.3. Online Component

4.3.1. Assessing Relevance

As mentioned earlier, we adapt the ReDDE approach for relevance estimation. Given a new query, the ReDDE approach involves querying the collection representatives. Using the results from this sample index as well as the estimated collection sizes, an estimate of the number of relevant documents in each collection is made.

Algorithm 1 $CollectionSelection(query) \rightarrow OrderedCollectionList$

```
1: Query the total sample collection
2: $Count \leftarrow 0$
3: for all results $r$ in the query results in descending rank do
4:    $r.Document.Score \leftarrow Count$
5:    $Count \leftarrow Count + mean(r.EstimatedSize/r.SampleSize)$
6: for all collections $c$ do
7:    $c.Score \leftarrow 0$
8:    for all documents $d$ in $c$ do
9:       if $d.Score < Threshold$ then
10:          $c.Score \leftarrow c.Score + 1$
11:   $c.Score \leftarrow \frac{c.Score \cdot c.EstimatedSize}{c.SampleSize}$
12: while exists a collection $c$ with $c.Score > 0$ do
13:    Pick a collection with $argmax\{ResidualRelevance(Collection)\}$
14: Return Order of Collections
```

In this approach, the query is sent to the collection samples. The same query is also sent to the union of the individual collection samples. The union sample collection returns a ranked list of documents which are relevant to the query. Next, the *Count* is initialized to zero. This count indicates the estimated number of relevant documents encountered thus far. After this initialization, the algorithm iterates through all of the results with the most relevant results being visited first. The document that corresponds to the result has its score set to *Count* which is the number of relevant documents encountered. Therefore, the score of each document is the estimated number of documents that are more relevant than it in the entire corpus.

In the next step, each collection is examined and its score is initially set to zero. Then for all the documents which are in the sample collection and have a score less than some threshold, the collection will receive one more point. The documents that contribute represent the documents which are in the top-k documents overall where k is the threshold. Finally, the collection's score is scaled by the ratio of the estimated collection size to the sample size.

4.3.2. Assessing Overlap

Here the overlap estimation is done at run time, based on a sample of the collections. As described in previous section, the set of top-k documents from each sample is determined. The overlap among the collections is estimated using the top-k documents from the samples.

As described in section 3 the overlap can be defined in terms of exact duplicates or near duplicates. Assessing overlap in terms of exact duplicates is reasonably straightforward – we take the intersection of the set of documents returned by individual collections in response to a query. To assess overlap in terms of near duplicates we can use the GHVs stored for each of the sampled documents. To consider two documents to be near duplicate at max 8 mismatches of w-bit hashes are allowed as described by Bernstein et. al. If we want to consider only exact duplicate documents then number of mismatches allowed can be set to 0. The overlap of a sample with previously selected collections is determined as the set of documents that are overlapping with previously selected collections. The overlap of collection C is estimated as overlap of the corresponding sample with samples of previously selected collections scaled by the ratio of the estimated collection size to the sample size.

$$Overlap_q(C_i) = Overlap_q(S_i) \times \frac{C_i.EstimatedSize}{S_i.size}$$

Where, $Overlap_q(C_i)$ is the overlap of collection C_i with previously selected collections. $Overlap_q(S_i)$ is the overlap of sample S_i with previously selected samples.

$C_i.EstimatedSize$ is the estimated size of the collection C_i. $S_i.size$ is the size of the sample S_i.

4.4. Combining Relevance & Overlap

As described in section 4.3.1, the number of relevant documents from each collection are determined at run time. After determining first collection based on maximum number of relevant documents, rest of the collections are ordered in terms of the residual relevance. The equation for computing residual relevance is included below.

$$ResidualRelevance_q(C) = C.Score - Overlap_q(C)$$

Here C.Score gives the estimated no. of relevant documents in the collection. The overlap component is the number of documents in the collection that are duplicates of the documents in the previously selected collections.

4.5. Comparison of ROSCO with COSCO

COSCO computes and stores coverage and overlap statistics offline, while ROSCO computes relevance and overlap statistics online, i.e. at the query time. The offline and online methods have complementary characteristics. The online method is conceptually very simple. It estimates the overlap by "simulating" the query on the samples of the collections. An advantage of this method is that it is able to give a better estimate of the overlap with respect to the specific query at hand. However, its effectiveness is critically dependent on having an unbiased sample that is representative of the entire collection. Since samples are often generated by probing queries, sometimes this could be a difficult goal.

In contrast, the offline method estimates collection overlaps for the specific queries in the query log, and “generalizes” the statistics for other unseen queries with the help of frequent item sets. While this method doesn’t depend on a collection sample, it has its own potential drawbacks. It is possible that overlap statistics computed using item sets are significantly different from the overlap statistics for the query. For example, if the user query is ”data mining integration” and the statistics for it are computed using statistics for frequent item sets ”data mining” and ”data integration” then they can be different from the actual statistics for the query ”data mining integration”.

CHAPTER 5

EMPIRICAL EVALUATION

5.1. Experimental Setup

The experiments described in this chapter were designed to test whether the collection selection system ROSCO can in fact perform better in an environment of overlapping text collections than the systems commonly accepted as being the most effective by the information retrieval community(which do not consider overlap). We also want to compare ROSCO and COSCO in presence of overlap among collections. This chapter covers the entire experimentation procedure performed to test our system, including the experimental setup, training performed and finally the results obtained.

Test Data: We evaluated the collection selection systems on the collections derived from TREC 2005 Genomics data. As we mentioned in Section 1.2, the focus in collection selection algorithms is on how to select collections such that the retrieval performance will be competitive with the same retrieval techniques being applied to a centralized "union" database of all collections. Accordingly, our primary focus is on evaluating each collection selection method with respect to retrieval on the union database (using the same document retrieval strategy), in terms of the recall metric R. Recall metric R is commonly used to compare different resource selection algorithms [8]. We will also provide evaluation w.r.t. TREC relevance judgements.

Testbed Creation: In order to do an accurate examination of the performance of the proposed solution we designed two different testbeds. A large number of documents were required to create each testbed. For genomic data, we took 200,000 documents

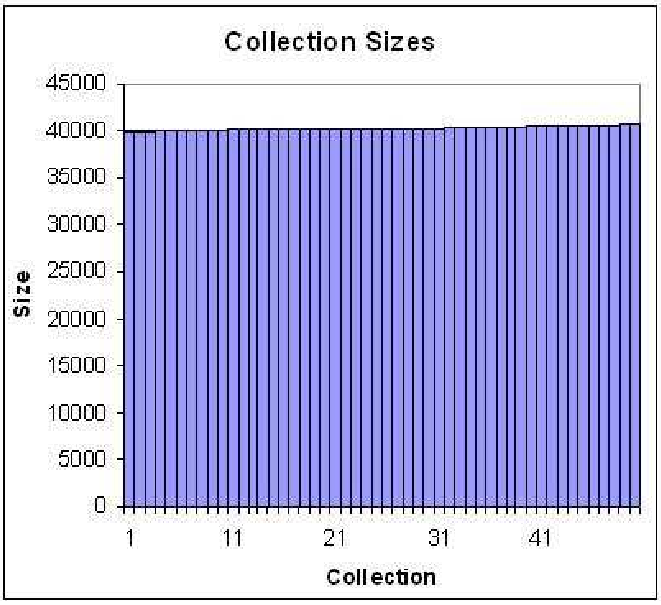

Fig. 4. The collection sizes in Testbed 1.

from the TREC collection. Using the clustering algorithm, 100 disjoint clusters are created from these documents. Clustering serves the purpose of creating topic specific overlap among the collections. We create two different testbeds from these documents.

Testbed 1: This testbed contains 50 collections. Each of the document is assigned to 10 randomly selected collections. Thus the total number of documents in this testbed is 2 million. Due to random assignment of documents, topic specific overlap among different collections could be low.

Figure 4 shows the size of each of the collections. It is almost uniform as the documents are distributed to 10 randomly selected collections. Figure 5 shows the distribution of relevant documents for the TREC topics among the collections in this testbed. Figure 6 shows the ratio of overlap between collection pairs. It shows that most of the collection pairs have 10-20% overlap.

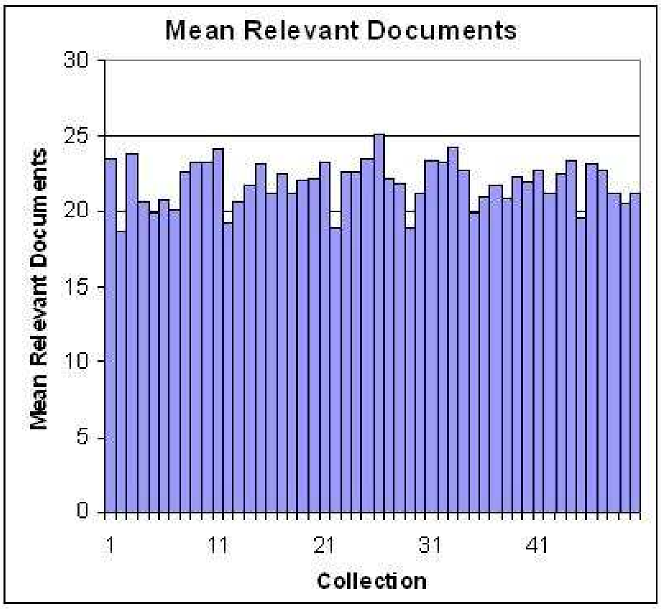

Fig. 5. The mean number of relevant results in Testbed 1.

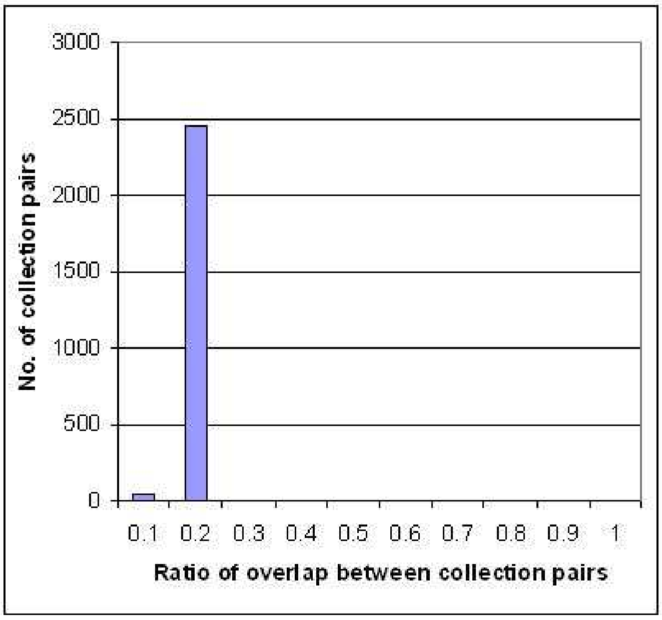

Fig. 6. Degree of overlap between collections in Testbed 1.

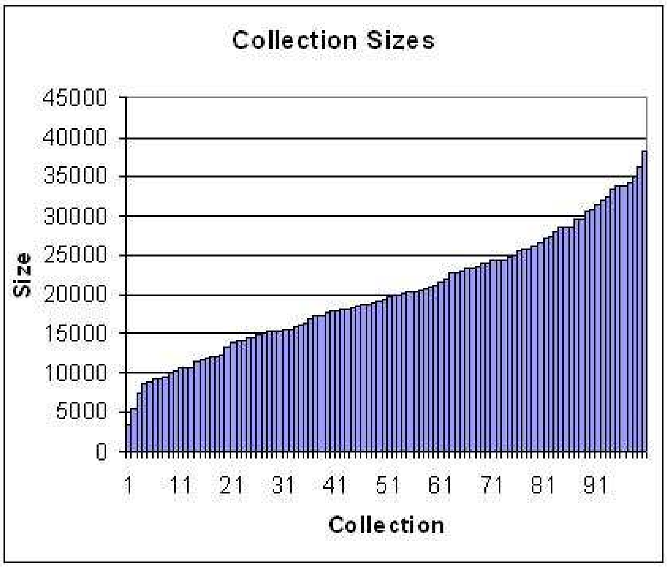

Fig. 7. The collection sizes in Testbed 2.

Testbed 2: This testbed contains 100 collections. Each of the clusters created is randomly assigned to 10 different collections. Thus the total number of documents in this testbed is 2 million. As one cluster is assigned to different collections, this testbed is expected to have higher topic specific overlap compared to testbed 1. Due to random assignment of clusters, the overlap among collections could vary.

Figure 7 shows the size of each of the collections. This testbed is different from testbed 1 in terms of the size of the collections, as the size of the collections vary in this testbed. Similar is the case for the mean relevant documents in each collection. Figure 8 shows the distribution of relevant documents for the TREC topics among the collections in this testbed. Figure 9 shows the ratio of overlap between collection pairs. This testbed has some collection pairs with overlap more than 20% as well.

Testbed 1 does not have collection pair with overlap greater than 20% compared

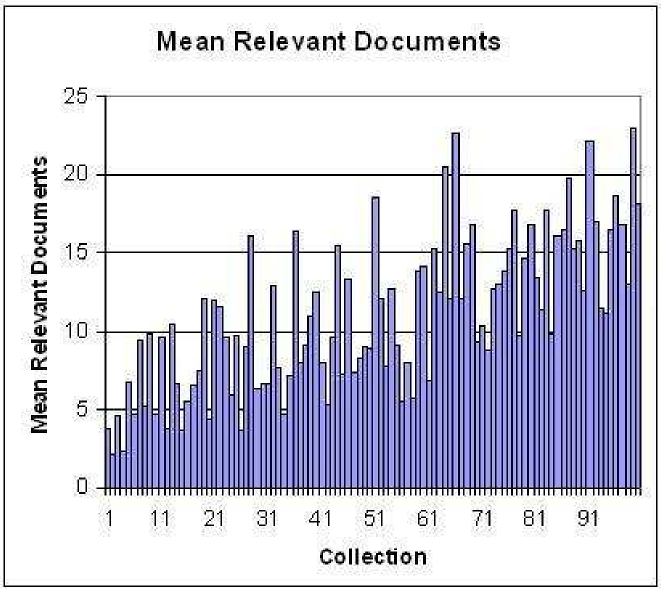

Fig. 8. The mean number of relevant results in Testbed 2.

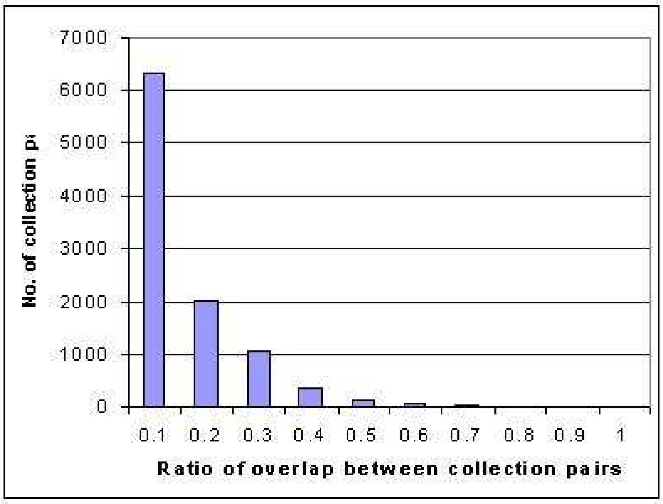

Fig. 9. Degree of overlap between collections in Testbed 2.

to testbed 2, which have almost 17% collection pairs with overlap greater than 20%. Testbed 1 is created by randomly assigning each document to 10 randomly selected collections resulting in less topic specific overlap compared to testbed 2. Due to these reasons, ROSCO and COSCO are expected to perform better on testbed 2 compared to testbed 1.

Tested Methods: We compared ROSCO and COSCO to ReDDE (which does not consider overlap between collections). ReDDE is built on top of CORI [7], which has been widely accepted as a stable method for collection selection. To establish bounds for the performance of our system we have also experimented with **Greedy Ideal** approach. This method attempts to greedily maximize the percentage recall (Equation 1.1), assuming oracular information. Specifically, greedy ideal assumes complete knowledge of every collection and will always pick the collection with the most documents in the top-k first followed by the collection with the real highest residual relevance next and so on. It understands both pair-wise and higher order overlap. For the empirical study, Greedy Ideal is implemented as a "post-facto" method–which calls all the collections with the query, and analyzes their results to decide on the ideal order. The method is "greedy" in the sense that given a collection subset of size c that is greedy maximal then it will pick a subset of size $c+1$ that is also greedy maximal and therefore it never backtracks.[1] Greedy ideal provides an upper bound on recall over the long run. Greedy Ideal ranking is used as a baseline ranking in the calculation of recall metric R.

[1]A non-greedy version will have to consider all possible $c+1$-sized subsets of collections and will thus be exponential even for the post-facto analysis!

Setting up COSCO: COSCO has offline and online components as described in [13]. The offline component of COSCO probes collections using training queries and then computes coverage and overlap statistics for these queries. COSCO also finds frequent item sets from the set of training queries. As we do not have frequency information about training queries, we assign equal frequency for each query. So all queries are equally important. Then COSCO computes statistics for frequent item sets using the statistics for training queries. COSCO sends 40 training queries to each of the collection and computes statistics based on documents retrieved from each collection.

Setting up ROSCO and ReDDE: ROSCO and ReDDE both use the same collection samples. The offline component of ROSCO was implemented as described previously. We selected 50 topic-based queries and used 40 of them for the training phase (with 10 left for the test phase).

When creating the collection sample, each collection in each testbed was sampled using randomly selected 25 of the 40 training queries. In our experiment, each sample contains approximately 10% documents from the collection. ReDDE and ROSCO both use same samples. Thus ReDDE and ROSCO both will have the same advantage of sample size. After creating the samples, 10 size estimates were made for each collection. The final size estimate is the mean of these estimates. To efficiently determine duplicates among the collections, Grainy Hash Vectors are used. Based on the experiments performed by Bernstein et. al. [6], we have selected parameters for GHV as follows. We are using 64 bit GHV with 32 hashes of 2-bits each. We are considering only exact duplicates for overlap among collections because we do not have any judgements in terms of near duplicates. To consider two documents as exact duplicates we are allowing 0 mismatches between the 2-bit hashes of their GHVs.

5.2. Experimental Results

We used 10 queries different from the training queries in the test phase. These 10 queries were sent to the four collection selection algorithms Greedy Ideal, COSCO, ReDDE and ROSCO. The recall metric R at each step (collection call) was determined for COSCO, ReDDE and ROSCO (with respect to union of all collections). Let B denote a baseline ranking and E a ranking provided by a resource selection algorithm. Ranking provided by Greedy Ideal method is used as a baseline ranking. Let B_i and E_i denote the number of relevant documents in the i'th ranked database of B or E. The recall metric R_n is defined as:

$$R_k = \frac{\sum_{i=1}^{k} E_i}{\sum_{i=1}^{k} B_i}$$

Evaluation with respect to union of all collection involves comparing the results to the top-100 results retrieved by running the same query on a database corresponding to the union of all the collections (see Section 1.2). The results are averaged over all the test queries in order to provide a clear look at performance. 5-fold cross validation is used to get the better estimate of the performance of collection selection algorithms. Ideally, we would like to get higher value of recall metric R at any given collection rank.

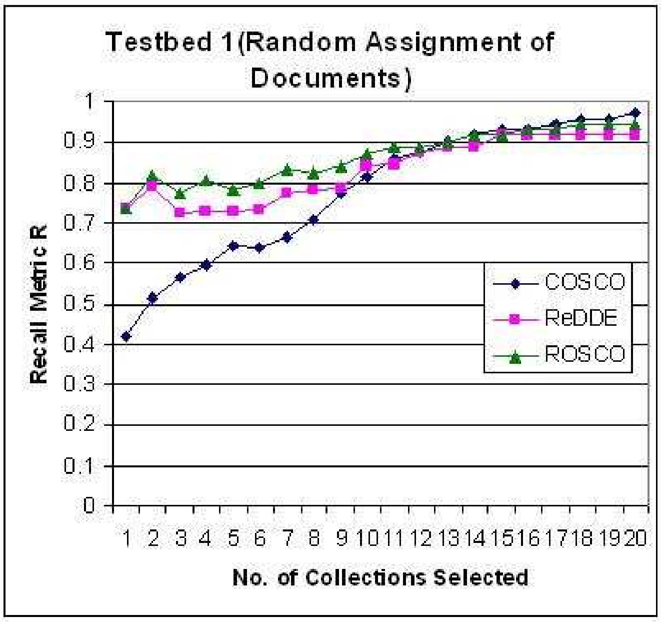

Fig. 10. Testbed 1, Evaluation on top 100 documents

COSCO uses *offline approach* to estimate the overlap statistics among the collections, while ROSCO uses *online approach* to estimate the overlap statistics among the collections. As described earlier, *online approach* estimates overlap statistics from the documents retrieved from the sample for a given query. While *offline approach* estimates overlap statistics for a query based on the overlap statistics for the item sets. *Online approach* can be better compared to the *Offline approach* as it estimates overlap statistics from the documents retrieved for the same query.

Figures 10 and 11 show the recall metric R at each step for each method (COSCO, ReDDE and ROSCO) for Testbed 1 and Testbed 2 respectively. We note that in all cases, ROSCO has a lead over ReDDE. Although the extent of the lead varies, the results show that ROSCO performs better than ReDDE in Testbed 1 by up to 7-8% when evaluation is done with respect to central union. The results for Testbed 2

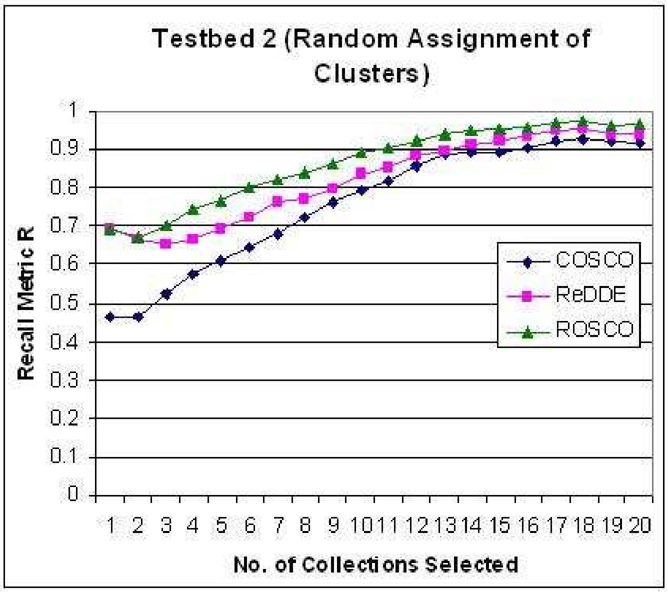

Fig. 11. Testbed 2, Evaluation on top 100 documents

show that ROSCO performs better than ReDDE in Testbed 2 by upto 7-8% when evaluation is done with respect to central union. ROSCO also performs significantly better than COSCO for top few collections. COSCO is worse compared to ReDDE even though it is taking overlap into consideration. The reason for this can be given that, as COSCO is computing coverage and overlap statistics based on averaging over the statistics for the frequent item sets they can be much different from the actual coverage and overlap statistics for a given query. While ReDDE is computing coverage statistics based on the samples, getting more accurate estimation for a given query if the sample is a good representative of the collection.

The recall improvement for ROSCO is consistently better on Testbed 2 since the overlap among collections is more prominent in this testbed.

Evaluation w.r.t. TREC Relevance Judgements: We have argued that col-

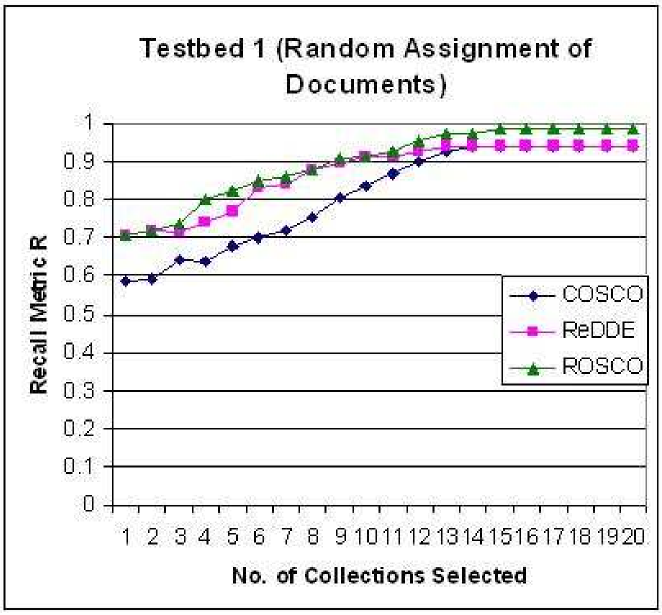

Fig. 12. Testbed 1, TREC Evaluation on top 100 documents

lection selection methods are best evaluated in terms of their competitiveness w.r.t. a central union database. However, for the sake of completeness, we also evaluated each collection selection method with respect to TREC relevance judgements (using the queries and associated relevance judgements that come with the TREC Genomics corpus). Here the results are presented for top 100 documents. Figures 12 and 13 show the results for both the testbeds. For TREC relevance judgements ROSCO performs better than ReDDE in Testbed 1 by 3-5% and in Testbed 2 by upto 8-10%. Here also ROSCO significantly outperforms COSCO.

Evaluation w.r.t. Space Requirements: For COSCO the space required is for storing frequent item sets as well as storing statistics for all frequent item sets. If the queries used for training are large then the set of frequent item sets can be large. ReDDE requires space to store the indexes of each sample and the size estimates

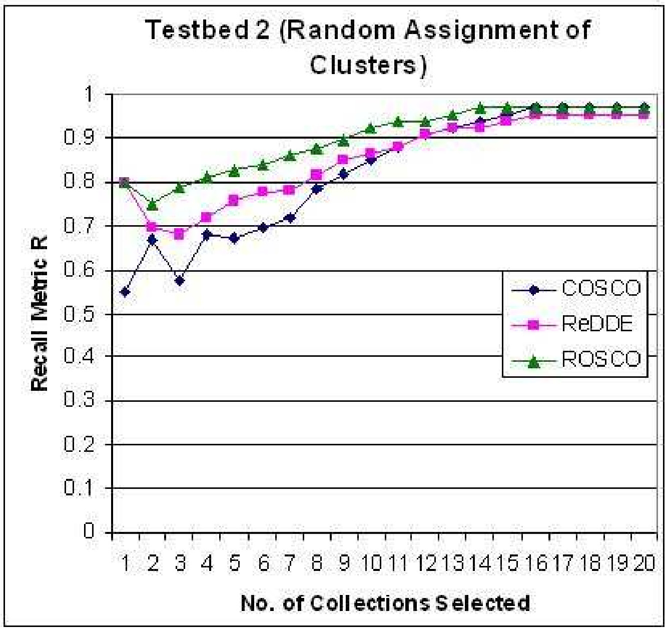

Fig. 13. Testbed 2, TREC Evaluation on top 100 documents

for each collection. Once the indexes are created the documents need not be kept. Only indexes can be used to determine the collection order. ROSCO is required to store these indexes, size estimates as well as the Grainy Hash Vectors. But for each document a Grainy Hash Vector of only 64 bits stored. So overall space requirement for ROSCO is little bit more than ReDDE in terms of storing GHV for each document. The space required in the current setup is shown in table I and table II.

Evaluation w.r.t. Processing Cost: There are two kinds of cost involved in these algorithms. 1) Offline Cost 2) Online Cost. For COSCO the offline cost comprises of probing the collections with set of training queries and then computing statistics for them, finding frequent item sets and then computing statistics for frequent item sets based on the statistics for queries. For ReDDE the offline cost includes creating samples of the collections and making size estimates of the collections. For ROSCO the

	Item Sets and Item Set Statistics	**Indexes**	**GHV**
COSCO	10	N/A	N/A
ReDDE	N/A	333	N/A
ROSCO	N/A	333	5

TABLE I

Space Requirements in MB for Testbed 1

	Item Sets and Item Set Statistics	**Indexes**	**GHV**
COSCO	34	N/A	N/A
ReDDE	N/A	316	N/A
ROSCO	N/A	316	4

TABLE II

Space Requirements in MB for Testbed 2

offline cost includes the cost incurred by ReDDE plus the cost incurred in computing GHVs for the sampled documents.

The online cost for COSCO includes mapping query to item sets and the estimating statistics for the query using statistics for the item sets. For ReDDE the online cost includes the cost of sending queries to collection samples and then determining number of relevant document from each collection. Online cost for ROSCO includes cost incurred by ReDDE plus the cost to estimate overlap using GHV for the sampled documents. For collection selection algorithms, major concern is the online processing cost. Figure 14 and 15 show the online processing cost for all three methods on both the testbeds. From the figures it can be seen that the processing cost for ROSCO is almost similar to ReDDE for testbed 1. For testbed 2, this cost is somewhat larger for ROSCO compared to ReDDE. ROSCO provides better performance for testbed 2 over ReDDE to account for greater processing cost. This cost is greater than that for COSCO but ReDDE and ROSCO perform significantly better than COSCO in terms of recall metric R.

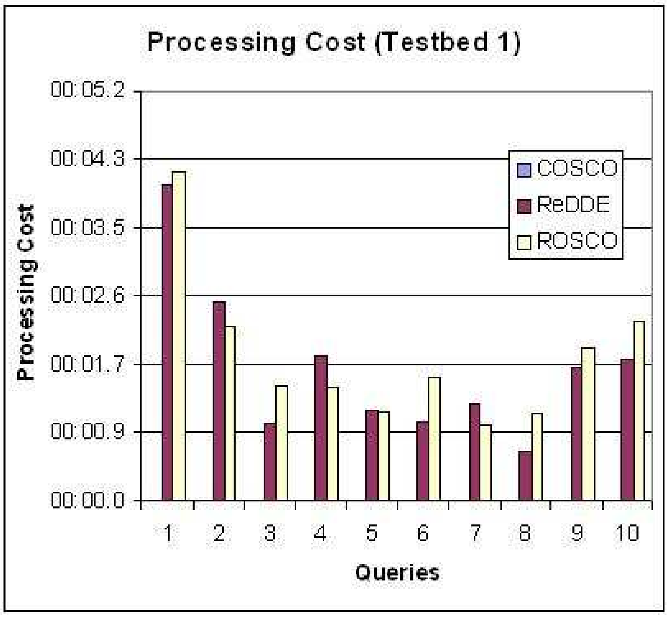

Fig. 14. Processing Cost on Testbed 1

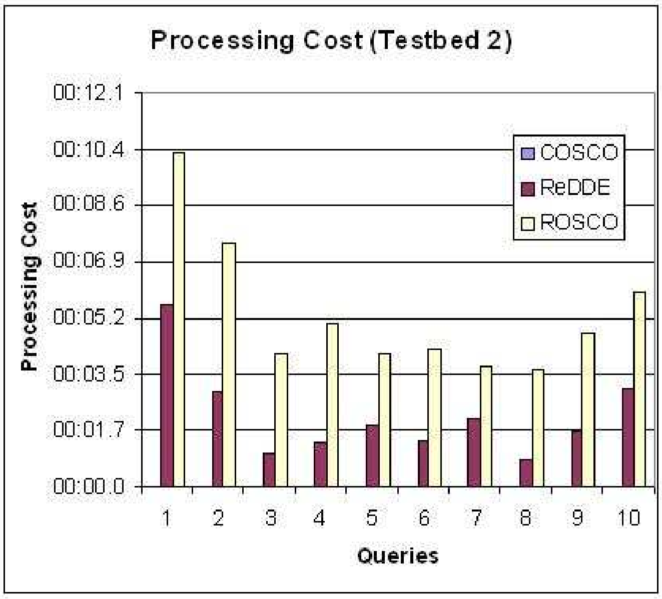

Fig. 15. Processing Cost on Testbed 2

CHAPTER 6

CONCLUSION

This thesis compared offline and online approaches for text retrieval in an environment composed of overlapping collections. We presented an online method called ROSCO which adapts a state-of-the-art relevance based collection selection method, ReDDE, to consider collection overlap. The strategy consisted in having an offline component to create samples of the collections and obtain the size estimates of the collection. The offline component also stores Grainy Hash Vectors for the sampled documents which are used by the online component to determine the overlap among collections. The online component uses all these to determine the order of the collections to be called for an incoming query.

We presented a systematic evaluation of the comparison of COSCO and ROSCO(which take overlap into account) with ReDDE(which does not take overlap into account) in presence of overlapping collections. Our experiments showed that ROSCO performs better than ReDDE, the current method of collection selection in presence of overlapping collections. ROSCO also performs better than COSCO in terms of the recall achieved after calling a fix number of collections. Although ROSCO(online approach) seems to be working better than COSCO(offline approach), its success depends on the quality of the sample from the collection.

We also evaluated all the algorithms in terms of space requirements. All the algorithms have different kind of space requirements. COSCO has space requirements for storing frequent item sets and coverage and overlap statistics stored for those frequent items sets. ReDDE and ROSCO both need to store the indexes of the sampled documents and also the size estimates of the collections. ROSCO also needs to store the GHVs for the sampled documents. Space requirements for ROSCO seems to be comparable with COSCO and ReDDE.

We evaluated all the algorithms in terms of processing cost as well. The processing cost comprises of offline as well as online cost. In terms of processing cost ROSCO and ReDDE require more time to do processing but a next collection in the order can be output comparatively fast.

REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB, 1994.

[2] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press / Addison-Wesley, 1999.

[3] M. Baillie, L. Azzopardi and F. Crestani. Adaptive Query-Based Sampling of Distributed Collections. SPIRE, 2006.

[4] Y. Bernstein and J. Zobel. Redundant documents and search effectiveness. CIKM, 2005.

[5] M. Bender, S. Michel, P. Triantafillou, G. Weikum and C. Zimmer. Improving Collection Selection with Overlap Awareness in P2P Search Engines. ACM SIGIR, 2005.

[6] Y. Bernstein, M. Shokouhi and J. Zobel. Compact Features for Detection of Near-Duplicates in Distributed Retrieval. SPIRE, 2006.

[7] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. ACM SIGIR, 1995.

[8] J. Callan. Distributed Information Retrieval. In W. B. Croft, editor *Advances in Information Retrieval*, 2000.

[9] J., Callan and M., Connell. Query-based sampling of text databases. *Information Systems*, 19(2):97-130, 2001.

[10] A. Das, M. Datar,A. Garg and S. Rajaram. Google news personalization: scalable online collaborative filtering WWW, 2007.

[11] D. Fetterly, M. Manasse and M. Najork. On the evolution of clusters of near duplicate web pages. In proc. of first Latin American Web Congress, pages 37-45. IEEE, 2003.

[12] L. Gravano, H. García-Molina, and A. Tomasic. GlOSS: text-source discovery over the Internet. ACM TODS, 24(2):229–264, 1999.

[13] T. Hernandez and S. Kambhampati. Improving text collection selection with coverage and overlap statistics. WWW (Special interest tracks and posters) 2005.

[14] P. Ipeirotis and L. Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. VLDB, 2002.

[15] Z. Liu, C. Luo, J. Cho, and W. Chu. A probabilistic approach to metasearching with adaptive probing. ICDE, 2004.

[16] W. Meng, C. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys*, 34(1):48–89, 2002.

[17] Z. Nie and S. Kambhampati. A frequency-based approach for mining coverage statistics in data integration. ICDE, 2004.

[18] H. Nottelmann, and N. Fuhr. Combining cori and decision-theoretic approach for advanced resource selection. ECIR, pp. 138–153, 2004.

[19] A. L. Powell and J. C. French. Comparing the performance of collection selection algorithms. ACM TOIS, 21(4):412–456, 2003.

[20] L., Si and J., Callan. Relevant Document Distribution Estimation Method for Resource Selection. SIGIR, 2003.

[21] Y. Zhang, J. Callan and T. Minka. Novelty and Redundancy Detection in Adaptive Filtering. SIGIR 2002.