ASU CSE TR 06-019; September 2006.

Relevance and Overlap Aware Text Collection Selection

BHAUMIK CHOKSHI , JOHN (WES) DYER , THOMAS HERNANDEZ
and
SUBBARAO KAMBHAMPATI
Arizona State University

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. 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. In this paper, we describe ROSCO, an approach for collection selection which handles collection relevance as well as overlap. We start by developing methods for estimating the statistics concerning size, relevance, and overlap that are necessary to support collection selection. We then explain how ROSCO selects text collections based upon these statistics. Finally, we demonstrate the effectiveness of ROSCO by comparing it to major text collection selection algorithm ReDDE under a variety of scenarios.

Categories and Subject Descriptors: H3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Design, Experimentation, Performance

Additional Key Words and Phrases: Distributed IR, Overlap Estimation

1. INTRODUCTION

Most existing approaches for collection selection 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 relevant for the incoming query (c.f. [Powell and French 2003; Si and Callan 2003; Callan et al. 1995]) This general strategy works fairly well when the collections do not overlap. Perhaps not surprisingly, all published evaluations of collection selection focus on disjoint collections [Powell and French 2003]. However, many real world

Authors names listed in alphabetical order. This research is supported in part by ASU Prop 301 grant ECR A601. The authors would like to thank Ullas Nambiar, Hemal Khatri, and Yi Chen for the many helpful discussions about this research.
Author's addresses: B. Chokshi, J. Dyer, T. Henandez, S. Kambhampati, Department of Computer Science and Engineering, Arizona State University, Tempe, AZ, 85282; email: bhaumik.chokshi@asu.edu, wesdyer@microsoft.com,thomas.hernandez@gmail.com, rao@asu.edu
Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 20acmYear ACM permissionCodeOne/acmYear/permissionCodeTwo-pageCode$5.00

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. Since the existing approaches fail to take into account overlap between collections when determining their collection order, they 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.

Our objective in this paper 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, our 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.

Our intent 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, we would want to call the most relevant collection first, and then iteratively choose the most relevant remaining collections that have least overlap with the already selected collection(s).

In this paper, we present an algorithm called ROSCO that is sensitive to both relevance and overlap among collections. Figure 1 shows the schematic architecture of ROSCO. ROSCO builds on ReDDE [Si and Callan 2003], a state of the art relevance-based collection selection algorithm. Like ReDDE [Si and Callan 2003], ROSCO uses query-based random sampling of collections, to estimate their relevance with respect to a given query. Specifically, it builds a representative of each collection via query 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 also computes and uses overlap statistics. The challenges lie in effectively defining, efficiently computing, gathering, and then adequately using the overlap information. In ROSCO we investigated two approaches for computing overlap which offer differing tradeoffs in terms of offline vs. online computation.

Offline approach. The first approach is to compute overlap statistics offline. In this approach, ROSCO stores overlap statistics with respect to queries. Doing so ensures that when a new query arrives, the system is able to find statistics relevant to that particular query. However, since it is infeasible to keep statistics with respect to every query, ROSCO stores them with respect to query classes. Query classes are defined in terms of frequent keyword sets, which are identified using item set mining techniques [Agrawal and Srikant 1994] from among past queries for which we have coverage and overlap statistics. Any new query can then be mapped to a set of known keyword sets. The benefit of using frequent keyword sets in place of

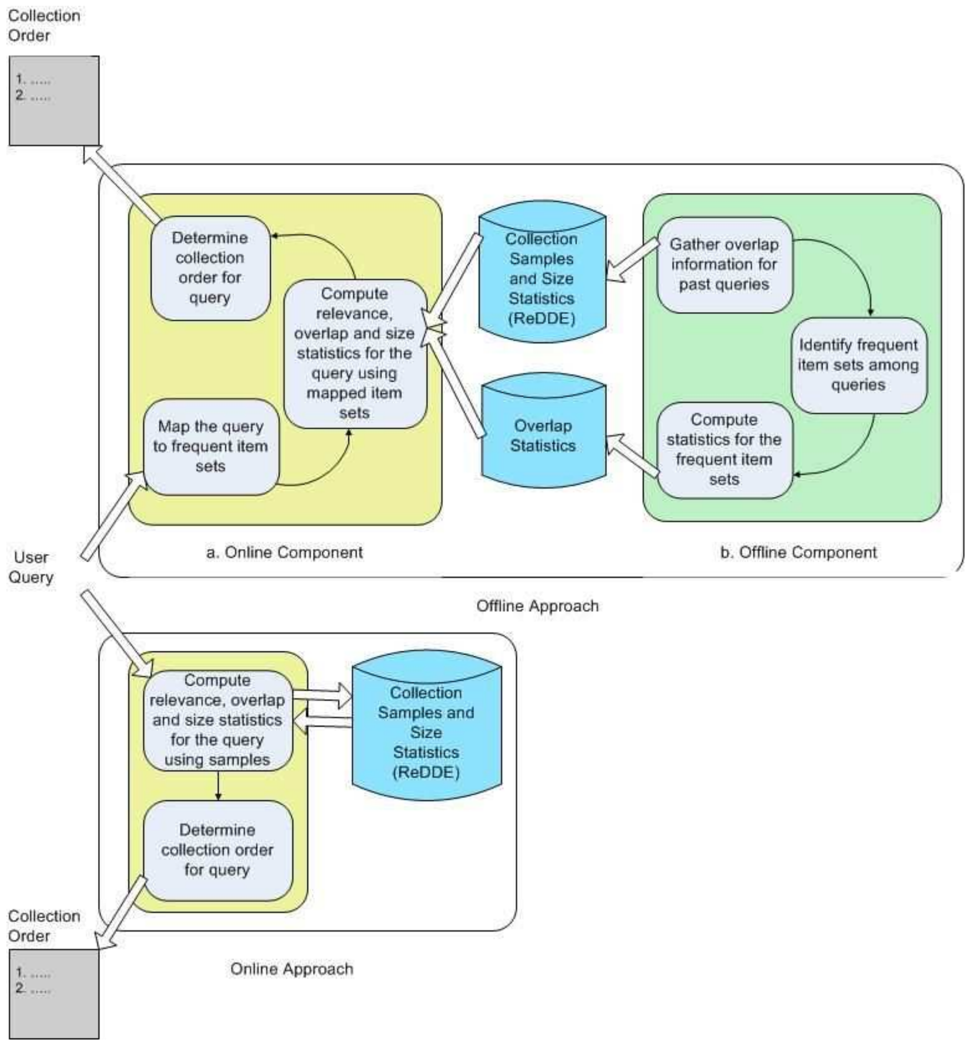
Collection Order
1. .....
2. .....
Determine collection order for query
Compute relevance, overlap and size statistics for the query using mapped item sets
Map the query to frequent item sets
Collection Samples and Size Statistics (ReDDE)
Overlap Statistics
Gather overlap information for past queries
Identify frequent item sets among queries
Compute statistics for the frequent item sets
a. Online Component
b. Offline Component
User Query
Offline Approach
Compute relevance, overlap and size statistics for the query using samples
Collection Samples and Size Statistics (ReDDE)
Determine collection order for query
Collection Order
1. .....
2. .....
Online Approach

Fig. 1. Architecture of ROSCO our collection selection system.

exact queries is that previously unseen queries can also be mapped to some item sets.

Online approach. The second approach is to compute overlap statistics from a representative sample of each collection at run time. In this approach, overlap statistics are not stored. Instead when a new query comes, the documents are obtained for the query from the stored sample (aka “representative”) of each collection. Overlap statistics are then computed using both the documents and the estimated size of the collections.

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 of the effectiveness of ROSCO in comparison to the existing techniques. Our evaluation is done on documents from TREC 2005 genomics corpus[TREC]. From this corpus, we formed testbeds with different amount of overlap among collections. Using these testbeds, we present a detailed experimental evaluation of ROSCO. Our experiments demonstrate that ROSCO outperforms the existing methods in the presence of overlap among collections.

Organization: The rest of the paper is organized as follows. Existing work related to the particular problems presented above is discussed in Section 3. Section 4 describes the challenges in defining and estimating collection overlap. The two approaches used by ROSCO for computing and using collection overlap statistics are described in Section 5. In Section 6, 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 is described in Section 7,followed by the results in Section 8. Finally,we conclude in Section 9.

2. COLLECTION SELECTION PROBLEM

The text collection selection problem that we address in this paper 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:

$$R^*_{S_c} = 100 \times \frac{|(\cup_i D_i) \cap \mathcal{DK}|}{k} \tag{1}$$

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 3). 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. In this paper, we are interested in relaxing this assumption. We present ROSCO, which combines the overlap statistics with relevance statistics. In terms of our discussion above, ROSCO accounts for the overlap between collections (i.e., it recognizes that $|(\cup_i D_i)|$ may not be equal to $\sum_i |D_i|$).

3. RELATED WORK

As mentioned by Powell and French[2003] 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 [Gravano et al. 1999], CORI [Callan et al. 1995], SavvySearch [Howe and Dreilinger 1997], and many other approaches [Meng et al. 2002; Ipeirotis and Gravano 2002; Liu et al. 2004]. In their survey, Powell and French [2003] show that CORI is among the most effective of these approaches and it had since been used in many further extensions [Craswell et al. 2003; Nottelmann and Fuhr 2004]. CORI tends to be sensitive to the size of the collections–picking larger collections over smaller ones. A recent system called ReDDE [Si and Callan 2003], is not as susceptible to variations in size. Furthermore, 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 approach explicitly takes collection overlaps into account. As we mentioned, ROSCO adapts ReDDE techniques for relevance estimation, and extends them to consider collection overlap.

ROSCO also builds on COSCO [Hernandez 2004; Hernandez and Kambhampati 2005], an earlier system from our group. The main difference between ROSCO and COSCO is that the former focuses solely on overlap, while ROSCO combines relevance *and* overlap. Contemporaneous with our work on COSCO, the work of Bender *et. al.* [2005] 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 work by Zhang *et. al.* [2002] 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 modelled 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 presence of overlap among test collections has also been recognized by others. The work by Bernstein and Zobel [2005] focuses on finding content-equivalent documents. They explore document fingerprinting techniques for finding redundant 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 [2004] 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. They use a large set of past queries with their associated coverage and overlap statistics and cluster them based on these statistics and the frequency of the training queries. Their query classification then allows them to correctly identify for a new user query which is the set of collections that has the maximum cumulative coverage (thereby taking into account overlap between collections). Note, though, that the main difference between the work described in this paper and Nie and Kambhampati's [2004] approach consisting of attribute-value hierarchies and query classes – is that, unlike in a relational environment, there are no attributes to classify queries on in a text collection environment.

Others have suggested using coverage information [Ipeirotis and Gravano 2002; Yerneni et al. 2000] or overlap information [Yerneni et al. 2000; Voorhees et al. 1994] in multi-collection scenarios, but none have actually learned and used both coverage and overlap statistics for the specific purpose of collection selection.

4. 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. In particular, we need pairwise overlaps of the form $ovlp(C_i, C_j)$, as well as the higher order overlaps of the form $ovlp(C_i, C_1 \cdots C_k)$. Although such overlap statistics have been used in the context of structured databases (in particular our own work in [Nie and Kambhampati 2004]), adapting such techniques to collection selection problem presents several challenges. 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. In the offline method, we learn statistics with respect to a log of queries. In the online method, we estimate overlap by evaluating the current query on the representative samples of the collections.

Estimating overlap over top-k documents: As discussed in Section 2, we need an estimate of overlap between collections such that we can obtain the highest percentage recall over the top-k documents. Making an effective estimate here will depend on the k value in top-k. Looking at equation 1 in Section 2, we see that what matters is not just the overlap between document sets D_i and D_j returned by two collections C_i and C_j, but rather the part of that overlap that intersects with $\mathcal{DK}$, the top-k documents returned by the centralized union. In particular, consider a query Q, and a collection C_i that has already been chosen for access. Suppose we are deciding which of two other collections C_j, C_k are best called as the second

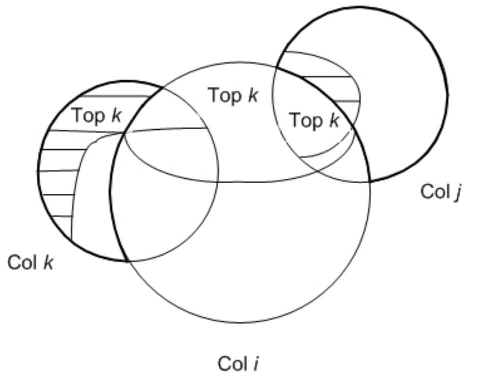

Fig. 2. Documents returned from collections i, j and k for a given query. Area enclosed by dark border on collections j and k shows the number of new documents from collections j and k. Area covered by horizontal lines shows the number of new top-k documents from collections j and k.

collection. We may have $|D_j - D_i| > |D_k - D_i|$ (which means that calling C_j will get us more new results), and yet have $|(D_i \cup D_j) \cap \mathcal{DK}| < |(D_i \cup D_k) \cap \mathcal{DK}|$ (which means C_k will give more new results in the top-k than C_j) as shown in Figure 2. This is less likely to happen when k in top-k is very large. In that case, each D_i will be an approximate subset of $\mathcal{DK}$, and thus the intersection with $\mathcal{DK}$ doesn't change the set. To handle this dependence on k, ROSCO considers two different approaches for overlap estimation that offer differing tradeoffs.

Overlap in terms of duplicates vs. highly similar documents: Another issue is whether the overlap between collections is defined in terms of "duplicates" or in terms of "highly similar" 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 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 highly similar documents.

Assessing overlap in terms of duplicates is reasonably straightforward – we take the intersection of the set of documents returned by individual collections in response to a query (c.f. [Nie and Kambhampati 2004]). Specifically, if $\mathcal{R}_i q$ and $\mathcal{R}_j q$ 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 $|\mathcal{R}_i q \cap \mathcal{R}_j q|$.

It is trickier to define overlap so it takes into account highly similar rather than just duplicate documents. There are two issues here: (i) how do we determine highly similar documents and (ii) how do we avoid doing many pair-wise similarity comparisons between the result sets of two collections. For the purposes of the first, the work by Zhang et. al. [2002] shows that the standard document similarity metrics (such as cosine similarity) are effective. We still have to generalize this to collections. To avoid costly comparisons between all pairs of documents across collections, we use an overlap approximation which amounts to considering the set of result documents for a keyword query over a particular collection as a single document. Overlap between two collections for a particular keyword query would thus be calculated as the intersection between the bag union of the results of the two collections for that query. Specifically, if $\mathcal{R}_i q$ and $\mathcal{R}_j q$ are the result sets from two collections C_i and C_j for a keyword query q, we start by first making the union of the bags corresponding to the individual documents[1] in the result sets, $\mathcal{B}(\mathcal{R}_i q)$ and $\mathcal{B}(\mathcal{R}_j q)$. The overlap $ovlp_q(C_i, C_j)$ is defined as the bag intersection[2] $|\mathcal{B}(\mathcal{R}_i q) \cap_B \mathcal{B}(\mathcal{R}_j q)|$.

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

Pairwise vs. higher-order overlap statistics: The fourth issue in defining collection overlap is that of handling overlap between more than two collections with respect to a query. This will be needed when we are trying to compute how many novel results are likely to be given by a collection C_i given that we have already selected collections $C_1 \cdots C_{i-1}$. In theory, this assessment can be done if we have access to overlap statistics for every subset of collections w.r.t. the query. In practice however, such an approach will lead to storing exponential number of overlap statistics with respect to every query. Furthermore, even computing overlap over more than two collections presents technical difficulties when we are interested in similarity-based (rather than duplicate based) overlap. For these reasons, we compute and store statistics for overlaps between *pairs of collections only.* The overlap between several collections will be approximated using only these *pairwise* overlaps at runtime.

5. APPROACHES TO GATHER AND STORE OVERLAP STATISTICS

As mentioned earlier we investigated two approaches to gather and use overlap statistics. The offline approach stores overlap statistics with respect to query classes and uses them to compute overlap statistics for an incoming query at runtime. In contrast, the online approach computes overlap statistics for incoming query from samples at runtime. In the following, we discuss both of these approaches.

5.1 Offline Approach

Storing Overlap Statistics w.r.t. Query Classes: Once we decide the method of overlap assessment, for each individual past query in the query log, we store a vector of overlap statistics $\overrightarrow{ovlp_q}$ where the components of the vector are $ovlp_q(C_i, C_j)$ for all i, j from 1 to n, with $i < j$. In addition to these overlap statistics, we also keep track of the result sizes, $\mathcal{R}_i q$ for each query q and collection i.

Keeping statistics with respect to each individual query would not only be costly, but also of limited use since the statistics could only be used for the exact same

[1]The bag representation of a document consists of a set of $(term, occurrence)$ pairs).
[2]Recall that the intersection $D_1 \cap_B D_2$ between two bags of words D_1 and D_2 is simply a bag containing each word and its minimum frequency across D_1 and D_2. The union in defined analogously, with "maximum" replacing "minimum".

query. A better approach (c.f. [Nie and Kambhampati 2004]) is to store statistics w.r.t. *query classes* (i.e., sets of related queries). For the case of keyword queries, query classes can also be defined in terms of keyword sets. A keyword query $k_i k_j$ corresponds to a query class which contains all the queries $k_i k_j k_1 \cdots k_l$ for all keywords $k_1 \cdots k_l$. Of particular interest are query classes that correspond to frequently occurring keyword sets among previously asked queries.

Essentially, our method consists in using the Apriori algorithm [Agrawal and Srikant 1994] to discover frequently occurring keyword sets among previously asked queries. (We are using TREC Genomics data for which we do not have query log. So we assume each query to be equally frequent.) For example, the query "*data integration*" contains three item sets: {*data*}, {*integration*}, and {*data, integration*}. All, some, or none of these item sets may be frequent, and statistics will be stored only with respect to the frequent ones. In addition to keeping the number of statistics relatively low, this method also improves the odds of having some partial statistics available for new queries, as we would possibly be able to map previously unseen queries to some item sets. Using the previous example, even though the query "*data integration*" may not have been asked as such, the idea is to use the statistics from the query "*data*" – if it is frequent enough – to estimate those for "*data integration*". The purpose of identifying the frequent item sets among the queries is to avoid storage of statistics for each query, and instead store statistics with respect to frequently asked keyword sets.

Computing statistics for frequent item sets:

Once the frequent item sets are identified, statistics for each of them need to be computed. The statistics of an item set are computed by considering the statistics of all the queries that contain the item set. Let $\mathcal{Q}_{IS}$ denote the set of previously asked queries that contain the item set IS. The statistics for an item set IS are defined as the weighted average of the statistics of all the queries in $\mathcal{Q}_{IS}$, according to the following formula:

$$\overrightarrow{ovlp_{IS}} = \sum_{q_i \in \mathcal{Q}_{IS}} \frac{freq_{q_i}}{\sum_{q_j \in \mathcal{Q}_{IS}} freq_{q_j}} \times \overrightarrow{ovlp_{q_i}} \qquad (2)$$

As apparent in Formula 2, the statistics of the queries are weighted by the frequency of each query, which was collected in the previous phase in addition to $\overrightarrow{ovlp_q}$. Using $\frac{freq_q}{\sum_{q_j \in Q_{IS}} freq_{q_j}}$ as the weight ensures that the statistics for the item set would be closer to those of the most frequent queries containing the item set. The statistics should thus be more accurate more often.[3]

A special case must also be dealt with when computing the statistics vectors of the frequent item sets, and that is for the *empty* item set, IS_{empty}. It is necessary to have statistics for the empty set in order to have statistics for entirely new queries (i.e. those which contain none of the frequent item sets identified by the offline component). The statistics for the empty set, $\overrightarrow{ovlp_{IS_{empty}}}$, are computed after having obtained all $\overrightarrow{ovlp_{IS}}$ vectors. $\overrightarrow{ovlp_{IS_{empty}}}$ is calculated by averaging the

[3]This assumes that the new queries will follow a distribution close to that of the previously asked queries.

statistics of all frequent item sets. Let us denote as *item_sets* the set of all frequent item sets. The formula we use is then:

$$\overrightarrow{ovlp_{IS_{empty}}} = \frac{\sum_{IS \in item\_sets} \overrightarrow{ovlp_{IS}}}{|item\_sets|} \tag{3}$$

The intuition behind this formula is that the statistics for the empty set should try to reflect the general coverage and overlap information of all collections, so that a query that cannot be mapped to any stored keyword set would be assigned average statistics which are representative of all collections.

Mapping the query to item sets at runtime: The system needs to map the incoming user query to a set of item sets in order to obtain some pre-computed statistics and estimate the coverage and overlap statistics for the query. More specifically, the goal is to find which group of item sets covers most, if not all, of the query. When several sets compete to cover one term, the set(s) with the most terms is(are) chosen. Consider for example the query "*data integration mining*", and suppose that only the item sets {{*data*}, {*mining*}, {*integration*}, {*data, mining*}, {*data, integration*}} are frequent. In that case, the query will be mapped to the two frequent two-term sets. Furthermore, if the item set {*data, integration, mining*} was frequent, then clearly the query would only be mapped to this three-term set.

The algorithm used to map the query to its frequent item sets is given in Algorithm 1. Practically speaking, the query q is mapped by first taking all frequent

Algorithm 1 mapQuery(query Q, frequent item sets FIS) $\rightarrow IS_Q$

```
1: $IS_Q \leftarrow \{\}$
2: $freqQTerms \leftarrow \{\}$
3: for all terms $t \in Q$ such that $t \in FIS$ do
4:     $freqQTerms \leftarrow freqQTerms \cup t$
5: $IS_Q \leftarrow$ PowerSet($freqQTerms$)
6: for all $IS_i \in IS_Q$ such that $IS_i \notin FIS$ do
7:     Remove $IS_i$ from $IS_Q$
8: for all $IS_i \in IS_Q$ do
9:     if $IS_i \subset IS_j$ for some $IS_j \in IS_Q$ then
10:        Remove $IS_i$ from $IS_Q$
11: Return $IS_Q$
```

item sets that are contained in the query (lines 3 to 7). Among these selected item sets, those that are subsets of another selected item set are removed (lines 8 to 10) on the grounds that the statistics of a subset would be less accurate. The resulting set, which we call IS_q, is the set of mapped item sets for the query q.

Computing overlap statistics for the query at runtime: Once the incoming user query has been mapped to a set of frequent item sets, the system computes overlap estimates by using the overlap statistics of each mapped item set. For example, if $IS_{q_{new}} = \{\{data, integration\}, \{mining\}\}$ then the system would use the statistics of both item sets {*data, integration*} and {*mining*} for its statistics estimates. The query statistics for q_{new}, noted as $\overrightarrow{ovlp_{q_{new}}}$, are calculated by averaging each of the mapped item set statistics. When the query q_{new} is not mapped to any item set (i.e. $IS_{q_{new}} = \{\} = IS_{empty}$), then we approximate $\overrightarrow{ovlp_{q_{new}}}$ as being equal to $\overrightarrow{ovlp_{IS_{empty}}}$. In summary, we can write the following definition for $\overrightarrow{ovlp_{q_{new}}}$:

$$\overrightarrow{ovlp_{q_{new}}} = \begin{cases} \frac{\sum_{IS \in IS_{q_{new}}} \overrightarrow{ovlp_{IS}}}{|IS_{q_{new}}|}, & \text{if } IS_{q_{new}} \neq IS_{empty} \\ \overrightarrow{ovlp_{IS_{empty}}}, & \text{if } IS_{q_{new}} = IS_{empty}. \end{cases} \tag{4}$$

5.2 Online Approach

In the online approach, the overlap estimation is done at run time, based on a representative sample of the collections. Specifically, before query time, ROSCO probes each collection and gets a representative sample of the collection. Since ReDDE already needs a collection sample to estimate collection sizes as well as do relevance based collection selection (see Section 6), we can just use these representatives for our purpose too. At the run time, the user query is evaluated over (i) each of the collection representatives as well as (ii) the union of all the representatives. Using the results retrieved from the union, the top-k documents from each sample are determined as described in Section 6.2. Next, the overlap among the samples is computed on these top-k documents. Overlap among collections C_i and C_j is computed using overlap among the representative samples S_i and S_j and the estimated size of the collections. Specifically:

$$ovlp_{q_{new}}(C_i, C_j) = ovlp_{q_{new}}(S_i, S_j) * minratio$$

where

$$minratio = min\left(\frac{C_i.estimatedSize}{S_i.size}, \frac{C_j.estimatedSize}{S_j.size}\right) \tag{5}$$

5.3 Tradeoffs between Online and Offline Methods

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 representative samples of the collections. An advantage of this method is that it is able to give a better estimate of the overlap over top-k documents. However, its effectiveness is critically dependent on having an unbiased sample that is representative of the entire collection. Since representatives are often generated by probing queries, 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 representative, it has its own potential drawbacks. To begin with it is possible that overlap statistics computed using item sets are significantly different from the overlap statistics for the query. Secondly, since item set based overlap statistics capture overlap at the entire collection level, overlap estimates computed using these statistics may be significantly different from the overlap for top-k results (especially for small values of k).

6. COMBINING RELEVANCE & OVERLAP

As mentioned earlier, we adapt the ReDDE approach for relevance estimation, and extend it to take overlap statistics into account. 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 (see below). The collection with the largest number of relevant documents is selected first. Up to this point, ROSCO follows the ReDDE methods. It is in selecting the remaining sources that the ROSCO approach diverges. In particular, ROSCO focuses on estimating the *residual relevance* of the remaining sources, given the already selected sources. It is in this computation that the overlap statistics are used.

6.1 Gathering Size and Relevance Statistics

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. It has been shown that a relatively small number of queries is required to obtain an accurate representation of each collection [Callan and Connell 2001]. Furthermore, a refinement can be made by using only the first few queries from the training data and obtaining subsequent query terms from the documents which are returned. During this exploration phase, the documents from each collection are separately stored. An inverted index is built for each collection sample to provide single source text retrieval from the sample.

Estimating Collection Size: Once a sample from each collection is available, collection size estimates are made. ReDDE uses the sample-resample method [Si and Callan 2003] 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. The sample-resample method however does not allow for overlap and thus requires an extension. So we modify their approach as follows: Let $\hat{N}$ be the sum of the estimated collection sizes, let $\hat{N}_{sample}$ be the total number of documents sampled, and let $\hat{N}'_{sample}$ be the total number of distinct documents sampled. Then the size of the union of all the collections, $\hat{N}'$, can be estimated as $\hat{N}' = \frac{\hat{N} \cdot \hat{N}'_{sample}}{\hat{N}_{sample}}$. These estimates are stored for each collection and for the union of the collections. The estimates are used for normalization purposes at runtime.

Finally, all of the documents that have been sampled are indexed together while noting from which sources each document has been obtained. It cannot be assumed that each document came from exactly one source, as it may have been sampled from multiple overlapping sources.

6.2 Answering Queries

When a query is posed to the ROSCO mediator, it will first use the relevance and size statistics to find the collection with the most top-k documents. Then the mediator will combine the relevance, size, and overlap estimates to find the collections with the most remaining top-k documents. This continues until the relevance estimates have been exhausted at which point the result sizes are used instead of top-k relevance estimates. The ROSCO collection selection algorithm is described in Algorithm 2 and is discussed below.

Algorithm 2 $CollectionSelection(query) \rightarrow OrderedCollectionList$

```
1: Load Overlap and Result Size statistics for the query
2: Query the total sample collection
3: $Count \leftarrow 0$
4: for all results $r$ in the query results in descending rank do
5:    $r.Document.Score \leftarrow Count$
6:    $Count \leftarrow Count + mean(r.EstimatedSize/r.SampleSize)$
7: for all collections $c$ do
8:    $c.Score \leftarrow 0$
9:    for all documents $d$ in $c$ do
10:       if $d.Score < Threshold$ then
11:          $c.Score \leftarrow c.Score + 1$
12:    $c.Score \leftarrow \frac{c.Score \cdot c.EstimatedSize}{c.SampleSize}$
13: while exists a collection $c$ with $c.Score > 0$ do
14:    Pick a collection with $argmax\{ResidualRelevance(Collection)\}$
15: while exists a collection $c$ not yet selected do
16:    Pick a collection with $argmax\{ResidualCoverage(Collection)\}$
17: Return Order of Collections
```

As mentioned, ROSCO aims to estimate the relevance and residual relevance of each individual collection given a query. The (residual) relevance of a collection is defined as the fraction of new relevant documents that it is expected to give (where relevance is measured in terms of top k results returned by the union database; see Section 2). The idea of Algorithm 2 is to find the collections with the highest number of remaining top-k documents first and then find the collections with the most remaining results. It accomplishes this by assigning each document a score equal to the number of documents which are estimated to be more relevant than itself. Each collection is then assigned a score which is the estimated number of top-k documents in the collection. The parameter k is set to a fraction of the total collection size. Finally, those collections with the most remaining top-k documents are chosen. The algorithm is described in more detail below.

Our algorithm can use either the offline or the online approach for overlap statistics (described in the previous section). For the offline approach, the algorithm begins by computing all non-empty subsets of the query and finding the corresponding frequent item sets. If no frequent item sets are found then the empty set statistics are used. Otherwise, the overlap statistics are the mean of the statistics of the frequent item sets that are found (as described in Section 5.1). For the online approach, the query is sent to the collection representatives. The overlap is then computed on the top-k documents retrieved from each sample. Once the overlap statistics are computed, the query is sent to the complete sample collection, which is the union of the individual collection samples. The complete 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 then 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 collection. To see why, note that *Count* is incremented by the mean of the ratio of each collection's estimated size to its sample size. The collections that are included in this computation are those in which the result can be found. The mean of this ratio is the number of documents in the real union of collections that the sample result is representing.

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.

At this point, each collection's score is an estimate of its share of the number of documents in the top-k documents overall. The algorithm then proceeds to select the collection with the highest residual relevance while there exist collections with a score greater than zero. Thus all of the collections that originally were thought to contain documents in the top-k documents are selected before any of the collections thought to not contain such documents. The equation for computing residual relevance is included below.

$$ResidualRelevance_q(C) = C.Score \times (1 - \frac{Overlap_q(C)}{C.EstimatedSize}) \tag{6}$$

The overlap component is the number of documents in the collection that overlap with documents in the previously selected collections. Therefore, this essentially reduces the estimated number of relevant documents in the collection. The overlap equation is:

$$Overlap_q(C) = \sum ovlp_q(C, C_i) \tag{7}$$

Where $ovlp_q$ is estimated by the offline or the online approach. Each C_i is a previously selected collection.

Once all of the collections which probably contain top-k documents have been selected, ROSCO can either stop (default), or continue to select additional collections by expanding its notion of relevance to include all results instead of just top-k documents. In the latter scenario, ROSCO will switch into a mode where it will continue to pick the collection with the highest residual coverage until all collections have been picked. Residual coverage is computed as:

$$ResidualCoverage_q(C) = \mathcal{R}_i q - \sum Overlap_q(C) \tag{8}$$

where $\mathcal{R}_i q$ are the result set size statistics as discussed in Section 5.1. Now that all of the collections have been selected, the order in which they were selected is returned.

7. EXPERIMENTAL SETUP

Test Data: We evaluated ROSCO on the collections derived from TREC 2005 Genomics data [TREC]. As we mentioned in Section 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^*, as discussed in Section 2. 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 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 use two different strategies to assign these clusters to the collections.

Testbed 1: This testbed contains 100 collections. Each of the clusters created is randomly assigned to 10 different collections. Due to random assignment of clusters, the overlap among collections could be negligible. We do not expect significant improvement by our approach over the existing approaches for this testbed.

Testbed 2: In real world scenario, we expect to have a significant amount of overlap among collections. Similar to Bender *et. al.* [2005] we use a sliding window over the clusters to create collections with controlled overlap. The first collection receives clusters Cl_1 to Cl_r (where r is the window size). The next collection receives $Cl_{1+offset}$ to $Cl_{r+offset}$ clusters. Here the window size r is kept at 10 and offset is kept at 2. So a total of 50 collections are created using 100 clusters.

The number of common clusters in a set of collections represent the degree of overlap. By construction, Testbed 1 assigns clusters randomly to the collections while Testbed 2 uses a sliding window approach. So Testbed 1 has less number of common clusters in a set of collections compared to Testbed 2. Thus the overlap among collections in Testbed 2 is expected to be more prominent compared to Testbed 1.

Tested Methods: In order to demonstrate the efficiency of ROSCO we compared it to ReDDE (which does not consider overlap between collections). ReDDE is built on top of CORI [Callan et al. 1995], 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), 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.[4] Greedy ideal provides an upper bound on recall over the long run.

Training Phase: 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). Since we did not have frequency information about the queries, all training queries were considered equally important.

When creating the collection sample, each collection in each testbed was sampled using 40 training queries. In our experiment, each sample contains approximately 20% documents from the collection. ReDDE and ROSCO both use same samples. Thus ReDDE and ROSCO both will have the same advantage of large sample size. After creating the samples, 10 size estimates were made for each collection. The final size estimate is the mean of these estimates.

8. EXPERIMENTAL RESULTS

We used 10 queries different from the training queries in the test phase. These 10 queries were sent to the three collection selection algorithms Greedy Ideal, ReDDE and ROSCO. The recall R^* at each step (collection call) was determined for every method (with respect to union of all collections). Evaluation with respect to union of all collection involves comparing the results to the top-k results retrieved by running the same query on a database corresponding to the union of all the collections (see Section 2). The results are averaged over all the test queries in order to provide a clear look at performance. Ideally, we would like to get high recall with the fewest collection calls.

We experimented with both the offline (item set based) approach and the online (representative sample based) approach for estimating overlap statistics. As discussed in Section 5 either the offline or the online approach can be used to compute overlap statistics among collections for a given query. We use two different values of top-k (i.e., $k = 500$ and $k = 2000$) to evaluate these two approaches. Based on the discussion in Section 5.3, our hypothesis is that if the collection representative is an unbiased sample, then for small value of k, online approach will perform better than the offline approach because the overlap statistics computed using this approach will better approximate the overlap among top-k documents. For large value of k both approaches should have similar performance because for large value of k , the item set based (offline) approach also approximates the overlap between top-k documents.

Figures 3-6 show the recall R^* at each step for each method (Greedy Ideal, ReDDE, ROSCO) for Testbed 1. Results for top-$k = 500$ are shown in Figures 3-4. Results for top-$k = 2000$ are shown in Figures 5-6. We note that in all cases, ROSCO has a lead over ReDDE. Although the extent of the lead varies, the results

[4]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!

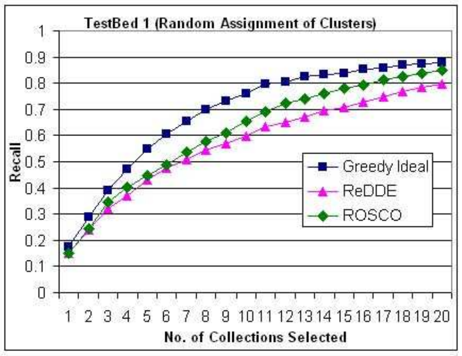

Fig. 3. Offline Computation on Testbed 1, Evaluation on top 500 documents

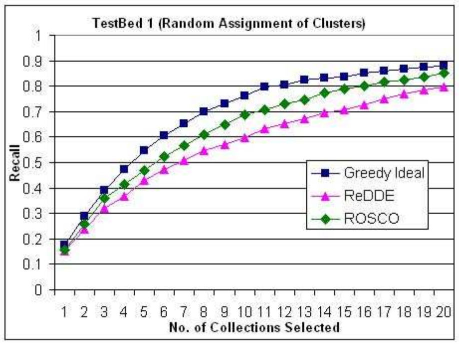

Fig. 4. Online Computation on Testbed 1, Evaluation on top 500 documents

show that recall wise ROSCO performs better than ReDDE in Testbed 1 by up to 10% when evaluation is done with respect to central union. It is worth noting that the performance of the greedy ideal shows the room available for improvement over ReDDE. In all cases, ROSCO is able to realize a significant portion of the possible improvement. To see the significance of this improvement, notice that Greedy Ideal is approximately 20% better than ReDDE. Thus, ROSCO is able to take ReDDE 50% of the way to Greedy Ideal!

The results in Figures 3-4 show that for top-k = 500, the online approach for overlap computation performs better than the offline item set based overlap computation. These results support the hypothesis that for a small value of top-k, overlap statistics over the top-k documents can be significantly different from the overlap statistics over the entire collection. The results in Figure 5-6 show that for top-k = 2000, the offline approach and online approach perform almost similar. So all the following results are presented for experiments using online approach only.

Figure 7 shows the recall R^* at each step for each method (Greedy Ideal, ReDDE, ROSCO) for Testbed 2. The results are presented for top-k = 500. ROSCO outperforms ReDDE approximately by upto 15% in Testbed 2. Once again, given that the Greedy Ideal is about 30% better than ReDDE, we are getting 50% of the way from ReDDE to Greedy Ideal. The recall improvement for ROSCO is better on Testbed 2 compared to Testbed 1. The explanation for this is that by construc-

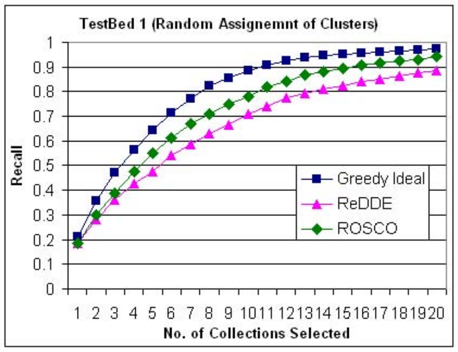

Fig. 5. Offline Computation on Testbed 1, Evaluation on top 2000 documents

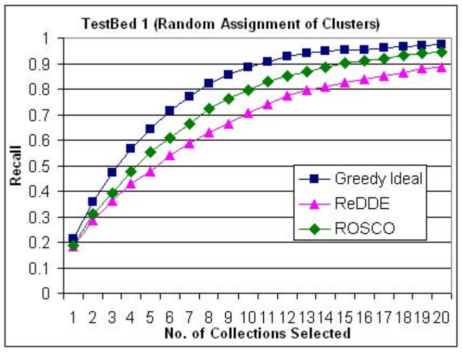

Fig. 6. Online Computation on Testbed 1, Evaluation on top 2000 documents

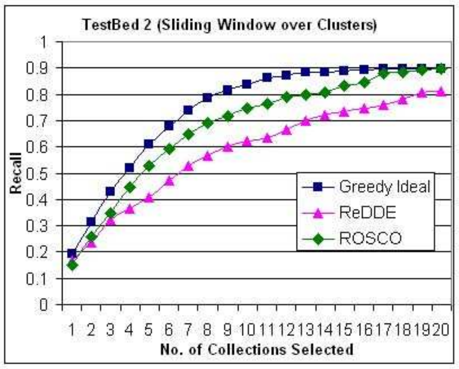

Fig. 7. Online Computation on Testbed 2, Evaluation on top 500 documents

tion, the overlap among collections for Testbed 2 is more prominent compared to Testbed 1.

Evaluation w.r.t. TREC Relevance Judgements: We have argued that collection 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-$k = 100$ because all the queries have TREC relevance judgements in the range of 100 to 300. Figures 8-9 show the results for both the testbeds. For TREC relevance judgements ROSCO performs better than ReDDE in Testbed 1 by 5-7% and in Testbed 2 by upto 15%. Once again it can be seen that ROSCO is able to go 50% of the way to Greedy Ideal which is significant.

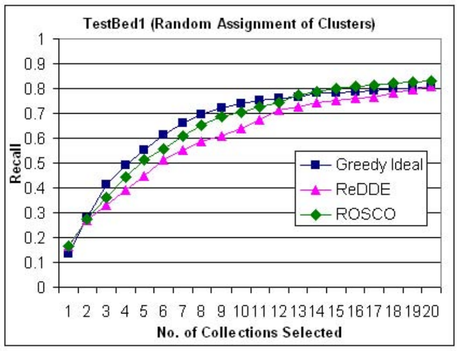

Fig. 8. Online Computation on Testbed 1, TREC Evaluation on top 100 documents

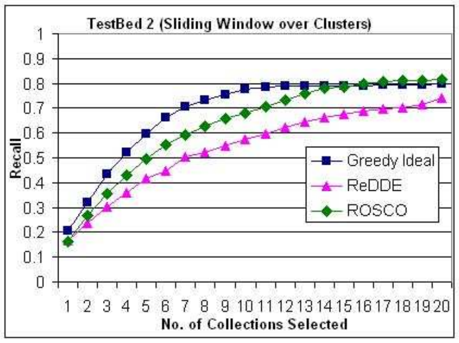

Fig. 9. Online Computation on Testbed 2, TREC Evaluation on top 100 documents

Evaluation in terms of Processing Cost: As shown earlier ROSCO achieves significant improvement over ReDDE in terms of recall. To analyze the processing cost overhead for ROSCO, we compared the processing time required by ROSCO and ReDDE to determine collection order. On 10 test queries, we present the analysis of processing cost for Testbed 2 in Figure 10. For all the queries, the processing time required by ROSCO is not more than 0.5 seconds higher than that needed by ReDDE. ROSCO and ReDDE both estimate the number of relevant documents for a given query from sample of each collection. The extra processing time required by ROSCO is to compute pairwise overlap statistics between collections and use those statistics to determine collection order. This amount is negligible compared to the time required in estimating the number of relevant documents from each collection.

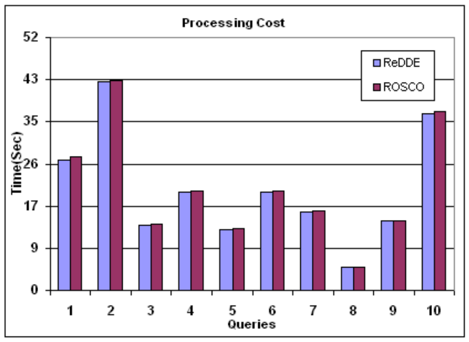

Fig. 10. Processing Cost on Testbed 2

Summary: We have evaluated ROSCO on collections derived from TREC Genomics data. ROSCO is evaluated w.r.t. central union of all collections as well as against TREC relevance judgements. We compare the performance of ROSCO to ReDDE.Our experiments show that ROSCO performs significantly better than ReDDE when the collections have overlap. The amount of improvement does of course depend on the type of overlap that is present among the collections. To quantify the overall room available for improvement, we also show the results from an oracular method Greedy Ideal (which establishes a rough upper bound on the attainable recall). We have also compared offline and online approaches to gather and use overlap statistics among collections. For small value of k in top-k, online approach performs better compared to offline approach. For large value of k in top-k, both approaches perform almost similar. It seems that the online approach is always at least as good as the offline approach. But the effectiveness of the online approach is dependent on having an unbiased sample that is representative of the entire collection. Since representatives are generated by probing queries, this could be a difficult goal. We have presented evaluations of our approach in terms of relevance of documents to the query as well as processing cost.

9. CONCLUSION

This paper addressed the issue of collection selection for information retrieval in an environment composed of overlapping collections. We presented a method called ROSCO which adapts a state-of-the-art relevance based collection selection method, ReDDE, to consider collection overlap. We presented a systematic evaluation of the effectiveness of ROSCO over existing methods. Our experiments showed that ROSCO outperforms current best methods when there are overlapping collections with a very small increase in processing cost.

REFERENCES

Trec 2005 genomics track data.

Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proceedings of the VLDB Conference*.

BENDER, M., MICHEL, S., TRIANTAFILLOU, P., WEIKUM, G., AND ZIMMER, C. 2005. Improving collection selection with overlap awareness in p2p search engines. In *Proceedings of the SIGIR Conference.*

BERNSTEIN, Y. AND ZOBEL, J. 2005. Redundant documents and search effectiveness. In *Proceedings of the CIKM Conference.*

CALLAN, J. AND CONNELL, M. 2001. Query-based sampling of text databases. *ACM Trans. on Information Systems 19,* 2, 97–130.

CALLAN, J. P., LU, Z., AND CROFT, W. B. 1995. Searching distributed collections with inference networks. In *Proceedings of the SIGIR Conference.*

CRASWELL, N., BAILEY, P., AND HAWKING, D. 2003. Server selection on the world wide web. In *Proceedings of the ACM Digital Libraries Conference.*

GRAVANO, L., GARCÍA-MOLINA, H., AND TOMASIC, A. 1999. GlOSS: text-source discovery over the Internet. *ACM Trans. on Database Systems 24,* 2, 229–264.

HERNANDEZ, T. 2004. Improving text collection selection with coverage and overlap statistics, M.S. Thesis. Dept. of CSE. Arizona State University.

HERNANDEZ, T. AND KAMBHAMPATI, S. 2005. Improving text collection selection with coverage and overlap statistics. In *WWW (Special interest tracks and posters).*

HOWE, A. AND DREILINGER, D. 1997. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine 18,* 2, 19–25.

IPEIROTIS, P. AND GRAVANO, L. 2002. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the VLDB Conference.*

LIU, Z., LUO, C., CHO, J., AND CHU, W. 2004. A probabilistic approach to metasearching with adaptive probing. In *Proceedings of the International Conference on Data Engineering.*

MENG, W., YU, C., AND LIU, K.-L. 2002. Building efficient and effective metasearch engines. *ACM Computing Surveys 34,* 1, 48–89.

NIE, Z. AND KAMBHAMPATI, S. 2004. A frequency-based approach for mining coverage statistics in data integration. In *Proceedings of the International Conference on Data Engineering.*

NOTTELMANN, H. AND FUHR, N. 2004. Combining cori and decision-theoretic approach for advanced resource selection. In *Proceedings of the ECIR conference*, pp. 138–153.

POWELL, A. AND FRENCH, J. 2003. Comparing the performance of collection selection algorithms. *ACM Trans. on Information Systems 21,* 4, 412–456.

SI, L. AND CALLAN, J. 2003. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

VOORHEES, E. M., GUPTA, N. K., AND JOHNSON-LAIRD, B. 1994. The collection fusion problem. In *Text REtrieval Conference, TREC.*

YERNENI, R., NAUMANN, F., AND GARCIA-MOLINA, H. 2000. Maximizing coverage of mediated web queries. Technical report, Stanford University.

ZHANG, Y., CALLAN, J., AND MINKA, T. 2002. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the SIGIR Conference.*

Received Month Year; revised Month Year; accepted Month Year