Relevance and Overlap in Text Resource Selection

Wes Dyer

April 7, 2005

Chapter 1

Introduction

1.1 Background

1.1.1 Traditional Text Retrieval

Information retrieval is an integral part of modern life. People routinely need to query information sources to acquire the knowledge they need. Often the information is stored in text documents of some sort such as web pages, news articles, emails, or even books. Text retrieval is information retrieval from text sources. Text retrieval originated from the need to manage the vast quantity of information stored in libraries [4]; however, over the past decade text retrieval has been extended to the web.

The fundamental problem of text retrieval is to retrieve as many of the most relevant documents in a collection of text documents as possible with respect to a query from the user. Many models have been proposed for text retrieval but the vector space model [24] has been generally accepted as the current best method. The vector space model uses an inverted term index in order to calculate term frequencies (tf) and inverse document frequencies (idf). These two values measure respectively the frequency of a term within a document and the frequency of a term within the collection. The two measures are then combined to score each document with respect to a query. Experiments and user studies have shown that the method seems to capture some sense of relevance [4].

1.1.2 Distributed Text Retrieval

With the success of text retrieval systems both on the web and in other areas, the number of text retrieval systems has grown rapidly over the past years. This in turn poses new challenges to text retrieval. Often the answer to a query is found in multiple collections which are possibly even separated geographically. This inspired the idea of a virtual collection which mediates the content supplied by other text collections [6]. When a query is submitted to the mediated collection, the mediator predicts which text collections will have the largest number of highly relevant documents within them. The query is then forwarded to these collections for processing. The mediator then ranks the results while removing irrelevant or redundant answers. The final sorted list is then returned to the user.

1.1.3 Text Resource Selection

The problem of selecting the best subset of the available collections to call is referred to as resource selection or collection selection [6]. In the case of collections of text documents, it is called text resource selection. Text resource selection can be performed over cooperative or uncooperative text collections. The latter assumption complicates the problem because a representative for each collection must be learned since the information is not readily available. The text resource selection methods that use this assumption differ in how they build a representative of the environment and the assumptions they make about that environment.

One example of the need for text resource selection is found in the mediation of various scientific abstract collections. Some of the many computer science abstract providers are ACM Digital Library [1], ACM Guide [2], ScienceDirect [25], Compendex [12], CiteSeer [10], DBLP [13], and CSB [11]. Often these collections vary in their ability to answer a given query. For example, ACM Digital Library may contain more valuable information regarding queries on *compilers* whereas DBLP could have better information about *databases*. The mediator should know to prefer ACM Digital Library for the first query and DBLP for the second. Instead of querying both sources, the mediator could call only the best or best-k sources. This reduces network load and has even been shown to improve overall retrieval performance compared to single source text retrieval [6]. Although this example shows the utility of text resource selection, the benefit is apparent when mediation

over hundreds or even thousands of collections is considered. This kind of environment is found in the news domain. It is apparent that some news sources are more authoritative on financial matters while they may be less so concerning sports for example. Furthermore, even a politically oriented news feed may be quite biased in the articles that it provides. When the mediator receives a query, it must quickly and accurately pick a small subset of the collections that captures the best articles from the union of all of the collections. The subset must also represent the diversity of information available. Herein lies the difficulty of text resource selection in an uncertain environment.

1.2 Challenges

It has already been noted that the primary challenge of text resource selection in an uncooperative environment is the uncertainty regarding the contents of each collection. It is also important to note that there are many important characteristics of these collections which affect text resource selection performance. Some of these characteristics are the quantity and quality of relevant documents within a collection with respect to a query, the size of each collection, and the overlap between collections.

The quantity of relevant documents in a collection is always measured with respect to a query; however, it is not easy to measure. In order to count the number of relevant documents, the notion of relevance must be quantified; this has proven to be a very difficult problem. Once relevance is quantified then how relevant must a document be to be counted? Furthermore, how are two documents, which vary in relevance, to be counted? The quality of relevant documents in a collection is also important. Intuitively users prefer a smaller number of highly relevant results to a larger number of mediocre results. How can this be measured and compared between collections? It is clear that collections will vary in the quality of their relevant documents, yet how can this be captured? Another important characteristic of the constituent collections is their sizes. How can accurate size estimates be obtained from uncooperative sources? Also there exists the possibility that collections contain overlap between each other. Overlap is a measure of the number of similar enough documents that exist between two or more collections. These characteristics should be reliably measured in order to perform text resource selection under varying circumstances. Finally, it is

nontrivial to combine these notions together in order to score the overall merit of a collection.

1.3 Overview of Proposed Solution

This thesis assumes that the environment is uncooperative, collections vary in relevance with respect to queries, collections differ in size, and overlap exists between collections. The solution presented in this thesis will address all of these assumptions which have not heretofore been examined together. The proposed solution is referred to as Relevance and Overlap Statistics for COllection selection (ROSCO). The aim of ROSCO can be easily described with the notion of the complete collection. The complete collection is the union of all collections. It contains every document that is contained in at least one collection. Distributed text retrieval over all of the collections can now be thought of in terms of single source text retrieval from the complete collection. The proposed solution will attempt to retrieve the highest percentage of top-k documents from the complete collection with respect to a given query for each subset of size n of collections. The top-k documents are those documents which are the first k documents in relevance ranked order. Furthermore, we can consider that certain documents which are similar enough form document classes and the problem is therefore to find the highest percentage of top-k document classes.

ROSCO first builds a representative of each collection via query sampling. The sampling serves three purposes. The first purpose is to provide a basis upon which to estimate the relevance of a collection with respect to a query. The second purpose is to determine the overlap between collections. The third purpose is to estimate the size of each collection. The first and second purposes are accomplished together by using random sampling whereas the third purpose specifically samples from frequent queries. This phase constitutes the offline portion of the solution.

Once the representative has been constructed, the online portion can be executed. When a new query arrives, the representative is used to determine the collection which has the highest probability of containing the most documents in the top-k. This collection is selected first. Then the collection with the highest residual number of documents in the top-k is selected. The term residual is used because a collection may overlap with the first collection and therefore contain several documents in the top-k documents in both collec-

tions. Only new relevant documents should be considered. This continues until there are no collections which are thought to contain top-k documents. At this time, ROSCO then ceases to consider top-k documents and rather considers all documents which are results for the given query. The number of documents which are results to the given query is termed coverage and the number of new documents with respect to a query and previously selected collections is called residual coverage. So the method selects the collection with the highest residual coverage and proceeds until all collections have been selected. Then the top-n collections are chosen as the best subset of size n.

This approach considers both relevance to a query and overlap between collections. It accurately estimates collection merit and improves upon existing methods as is shown in this thesis.

1.4 Contributions of this Work

This work makes three primary contributions. First, it proposes a new method that incorporates both relevance and overlap which has not been done previously. Second, eight test beds are provided which vary three fundamental attributes: relevance, overlap, and size. These test beds allow a method to be tested under a variety of conditions; this provides an understanding of the effect that these factors have on the method's performance. Third, a detailed evaluation of the proposed method as well as three other major methods is undertaken using the eight test beds. This evaluation describes the performance of these methods in detail.

1.5 Overview of Thesis

The thesis first discusses related work in Chapter 2. A detailed description of ROSCO is given in Chapter 3. The results of the experiments as well as discussion relating to the results is provided in Chapter 4. Suggestions for future work are described in Chapter 5. Finally, the conclusions of this thesis are made in Chapter 6.

Chapter 2

Related Work

There are many methods related to text resource selection. They differ drastically in what assumptions they make about the underlying environment, how they build a representation of that environment, and how they use the information to make predictions. Other important related research areas are query based sampling, collection size estimation, duplicate detection, and gathering/using overlap statistics.

2.1 Text Resource Selection

Some methods assume that the underlying collections are cooperative and provide the necessary metadata. These methods tend to emphasize protocols of communication. One such method is STARTS [15] which differs sharply from ROSCO which does not make this assumption. Most methods assume that the collections are not cooperative. gGLOSS [16] assumes that the collections use a common similarity measure when responding to queries. This allows gGLOSS to use the notion of the sum of similarities since similarity of a document with respect to a query can be compared across collections. ROSCO differs from gGLOSS because it does not assume that collections use a common similarity measure. In fact, the similarity measure of each collection is not important. Most methods also do not make this assumption. CVV [32] attempts to use the distribution of query terms across all collection in order to rank collections. CORI [8], which is the most widely accepted method, builds a virtual document corresponding to each collection through query sampling. Then it performs single source text retrieval over the virtual

documents. The ranking of the virtual documents with respect to the query is used to order the collections. Intuitively, this captures some sense of relevance and corresponds closely with traditional information retrieval. French and Powell [14] provide a survey of CORI, CVV, and gGLOSS. In this survey, they show that CORI outperforms the two other methods in a wide variety of settings. Each of these methods seek to approximate a relevance based ranking of the collections; however, French and Powell also showed that they often pick larger collections over smaller collections which corresponds to a size based ranking. To address this, a new method named ReDDE was designed that is not as susceptible to variations in size. Furthermore, ReDDE seeks to locate the top-k documents which is a stronger form of relevance based ranking. ROSCO differs from CORI, CVV, and gGLOSS because it considers size. ROSCO also differs from ReDDE since it takes overlap into account as well. Finally, COSCO [18] is a method that estimates the coverage and overlap of each collection with respect to a query. COSCO probes the collections to build a set of frequent item sets. Coverage and overlap statistics for each collection are maintained with respect to these frequent item sets. When a query is asked, the query is mapped to one or more frequent item sets which then provide reliable estimates of coverage and overlap for each collection with respect to the query. ROSCO contrasts with COSCO because it attempts to find the top-k documents rather than all documents which are returned by the collections.

2.2 Query Based Sampling

Query based sampling [7, 28] is used to build an accurate representative of each collection. This representative is in turn used to estimate relevance and size of the corresponding collections. Query based sampling can build accurate representatives with a small number of queries and documents.

2.3 Collection Size Estimation

There are two main methods of collection size estimation. These methods are the capture-recapture method [20] and the sample-resample method [29]. The capture-recapture method sends multiple queries to a collection and assumes that the distribution of the results of the queries are independent.

This enables the method to measure the overlap between queries and use this to estimate the collection size. The sample-resample method uses the representative of a collection as well as the actual collection. A query is sent to both the representative and the corresponding collection. The percentage of documents that are returned by the representative is known. Then it is assumed that the representative is a good sample and therefore roughly the same percentage of documents of the total collection are returned from the actual collection. Thus it can estimate the collection size. The sample-resample method requires significantly less interactions with the collection. Both ROSCO and ReDDE make use of the sample-resample method to estimate collection size.

2.4 Duplicate Detection

Measuring overlap requires the detection of duplicate or highly similar documents. A number of methods have been applied to this problem such duplicate detection (exact similarity) [26, 9, 27] and document fingerprinting as well as text retrieval techniques [17, 31].

2.5 Gathering/Using Overlap Statistics

Coverage and overlap statistics were first applied to the relational model by Nie and Kambhampati [22]. Past queries are used to probe the sources and build statistics based on frequency. New queries are then mapped to frequent item sets to provide reliable statistics. In fact, COSCO is an extension of this approach to the text retrieval domain.

Chapter 3

Relevance and Overlap

ROSCO is centered around the notions of relevance and overlap. It incorporates and extends ideas from ReDDE [28] and COSCO [18]. This is important because on the one hand ReDDE addresses the need to find the top-k most relevant documents but it does not deal with overlap. On the other hand, COSCO deals with overlap but does not address top-k relevance. Furthermore, it is nontrivial to extend these methods to include the ideas that they lack. This chapter describes how these ideas are used and how the methods are extended to make them more viable.

3.1 Overview

There are two distinct components which are necessary for ROSCO: an online component and an offline component. The goal of the offline component is to build an accurate representation of the collections. This includes statistics about collection relevance, collection size, and overlap between collections. The offline component uses query based sampling [7] to acquire the necessary sample for building the representative. Once the query based sampling has finished then the statistics must be computed. Each collection's size is estimated as well in order to normalize later computations by the collection's size. An inverted term index is built for the union of the collection samples while the source of each document in this index is noted. Finally, training queries are used to find frequently jointly occurring query terms called frequent item sets. Overlap statistics are then computed and stored in relation to these frequent item sets. At this point the system is ready to answer

queries.

The second component of the system is the online component. When a new query arrives, ROSCO queries the centralized inverted term index. Using the results from this sample index as well as the estimated collection sizes, an estimate of the number of top-k documents in each collection is made. The collection with the largest number of top-k documents is selected first. At this point the number of top-k documents is adjusted for each remaining collection by the estimated overlap with regard to the query. The collection with the highest residual number of new top-k documents is selected second. This continues until all of the collections which are estimated to have at least one top-k documents are selected. At this point, ROSCO loosens its notion of relevance to allow all answers to a query. It will then use the COSCO method to select the remaining collections. In the following sections, the two components will be described in detail.

3.2 Gathering Statistics

In order to build an accurate representation of each collection, ROSCO must acquire the information necessary to support both the idea of relevance and the idea of overlap. There are four parts to the offline component: query based sampling, size estimation, computing relevance statistics, and computing overlap statistics. To support each portion of the offline component, it is assumed that a set of training queries are supplied to the system. A very effective source for these training queries are those queries which have been posed to the system in the recent past [18].

3.2.1 Query Based Sampling

During query based sampling, a number of random queries 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 [7]. 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.

3.2.2 Estimating Collection Size

Now that a sample from each collection is available, collection size estimates are made. The sample-resample method [28] is used to estimate collection size. It assumed that the sample is representative of the real collection. Then a query is randomly selected from the training queries. This query is sent to both the real collection and the sample. Note that the sample collection size is known and is denoted as N_{sample}. The number of results in the sample is denoted as R_{sample} whereas the number of results from the real collection is denoted as R. Let N be the size of the collection. So the probability, $P(A)$ that the real collection contains the query term and the probability $P(B)$ that the sample contains the query term are shown below.

$$P(A) = \frac{R}{N} \tag{3.1}$$

$$P(B) = \frac{R_{sample}}{N_{sample}} \tag{3.2}$$

Now since, $P(A) \approx P(B)$ then we have the following.

$$\hat{N} = \frac{R \cdot N_{sample}}{R_{sample}} \tag{3.3}$$

Si and Callan showed that when the mean of several estimates is used, the absolute error ratio of the size estimate is small [28]. Note that the size of the union of all of the collections can now be estimated. It is just the sum of all of the individual estimates; however, this is not accurate in the presence of overlap. The sample-resample method does not allow for overlap and thus requires an extension. 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 follows.

$$\hat{N}' = \frac{\hat{N} \cdot \hat{N}'_{sample}}{\hat{N}_{sample}} \tag{3.4}$$

These estimates are stored for each collection and for the union of the collections. The estimates are used in the online component for normalization purposes.

3.2.3 Computing Relevance Statistics

Finally, all of the documents that have been sampled are indexed together while noting from which sources each document has been obtained. Unlike ReDDE, it cannot be assumed that each document came from exactly one source. It is entirely possible that one document was sampled from multiple sources. Furthermore, it is possible to not only consider duplicates as exact replicates but also as documents which are similar enough. Again it can easily be termed as document classes rather than just documents. Once an inverted index is built for the union of the collections then the individual collection indices can be removed.

3.2.4 Computing Overlap Statistics

Finally, it is necessary to compute statistics related to overlap. COSCO [18] provides a comprehensive way to accomplish this goal and its approach is adopted into ROSCO. The overlap between two collections i and j is defined as the size of the intersection of the results from the collections. Let R_i denote the set of results from collection i whereas R_j denotes the set of results from collection j.

$$overlap(C_i, C_j) = |R_i \cap R_j| \tag{3.5}$$

Note that higher order overlap statistics are not computed for efficiency reasons. Also, intuitively these overlap statistics should provide diminishing returns. Therefore for space and time considerations they are not included.

Before computing the overlap statistics, frequent item sets must be identified. The training set of queries is used to identify frequent item sets. Each query is considered as a set of words as opposed to a bag of words where multiple occurrence matters. Then the set of all non-empty subsets of the query term set is the item set for this query. The Apriori algorithm [3] is applied to these sets to discover which item sets are frequent.

The most frequent training queries are sent to each source to find result sizes and compute overlap as mentioned above. Then the statistics for each

frequent item set is computed as a weighted average of the statistics for each query that contains the frequent item set as a subset. It stores both the result size and the overlap estimates in relation to every other collection. Finally, the empty set is computed as the mean of all item sets. The empty set statistics are necessary because some queries will not map to any item sets although this should be rare if past behavior is indicative of future behavior. These statistics are then stored.

3.3 Answering Queries

Now that statistics regarding relevance, size, and overlap have been stored, it is now possible to answer queries. When a query is posed to the 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 resource selection algorithm is described below.

3.3.1 Finding the Most Relevant Collections

The idea of algorithm 1 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. Finally, those collections with the most remaining top-k documents are chosen and then it selects the rest of the collections by choosing which collection has the most remaining results. The algorithm is described in more detail below.

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 statistics are the mean of the frequent item sets that are found. The query is then sent to the total sample collection. Recall that this is the union of the individual collection samples. The sample complete collection returns a ranked list of documents which are relevant to the query. Next, the *Count* is initialized

Algorithm 1 $ResourceSelection(query) \rightarrow OrderedResourceList$

Load Overlap and Result Size statistics for the query
Query the total sample collection
$Count \leftarrow 0$
for all results r in the query results in descending rank **do**
  $r.Document.Score \leftarrow Count$
  $Count \leftarrow Count + mean(r.EstimatedSize/r.SampleSize)$
end for
for all collections c **do**
  $c.Score \leftarrow 0$
  for all documents d in c **do**
    if $d.Score < Threshold$ **then**
      $c.Score \leftarrow c.Score + 1$
    end if
  end for
  $c.Score \leftarrow \frac{c.Score \cdot c.EstimatedSize}{c.SampleSize}$
end for
while exists a collection c with $c.Score > 0$ **do**
  Pick a collection with $argmax\{ResidualRelevance(Collection)\}$
end while
while exists a collection c not yet selected **do**
  Pick a collection with $argmax\{ResidualCoverage(Collection)\}$
end while
Return Order of Collections

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 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 exists 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.

$$ResidualRelevance(C) = C.Score \cdot (1 - \frac{Overlap(C)}{C.EstimatedSize}) \tag{3.6}$$

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 described below.

$$Overlap(C) = \sum Overlap(C, C_i) \tag{3.7}$$

Each C_i is a previously selected collection and the statistics for this have been computed at the first step of the algorithm.

3.3.2 Using Overlap to Find New Results

Once all of the collections which probably contain top-k documents have been selected, then the notion of relevance is expanded to include all results instead of just top-k documents. This is the goal of the COSCO algorithm. Therefore, ROSCO will now switch into COSCO 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 follows.

$$ResidualCoverage(C) = C.ResultSize - \sum Overlap(C) \tag{3.8}$$

Now that all of the collections have been selected then the order in which they were selected is returned. This ordering constitutes the approximated best subset of collections of size n where n is the first n collections in the list.

The algorithm for resource selection is based on ideas first proposed in COSCO and ReDDE; however, ROSCO clearly extends both of these ideas by unifying the notions of relevance and overlap. This is important because in real world scenarios it most likely the case that collections vary in relevance with respect to a given query and collections have overlap. The algorithm is very efficient because most of the computation is performed offline and users will not see the more intensive calculations there.

Chapter 4

Experimental Evaluation

In order to evaluate the performance of ROSCO, an extensive series of experiments were performed under varying circumstances. These experiments show many facets of the performance of ROSCO but also demonstrate various properties of the other resource selection methods including COSCO [18], ReDDE [28], and CORI [8]. This chapter begins by describing how performance of the various methods is measured and describing the upper and lower bounds of performance. Then it describes in detail how the test beds were created to evaluate performance. A detailed analysis of these test beds shows that they do indeed provide an diverse and substantial array of environments in which to test the various methods. Each of the tested methods will be described in detail as well as how the training was performed. The actual testing process is described in the next section followed by the results of these tests.

4.1 Measuring Performance

It is the goal of this thesis to provide a method to select a subset of resources such that they return the highest possible percentage of top-k documents for such a subset size. This naturally leads to a metric to describe the performance of the resource selection system. Percent recall is the cumulative number of documents in the top-k documents that have been returned divided by the total number of documents in the top-k documents. Therefore, if collection i returns results R_i and the total number of documents in the top-k documents is R then the following equation describes percent recall.

$$R* = \frac{|\cup R_i|}{|R|} \tag{4.1}$$

The method that maximizes the $R*$ overall is called greedy ideal. It is greedy because given a subset of size n that is greedy maximal then it will pick a subset of size $n+1$ that is also greedy maximal and therefore it never backtracks. The greedy ideal has 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 higher order and lower order overlap. Greedy ideal provides an upper bound on performance over the long run. This is the method that ROSCO tries to approximates. Another metric that illustrates collection performance is the ratio between a method's percent recall and greedy ideal's percent recall. This shows how well the algorithm approximates the greedy ideal.

Interestingly, most methods with the notable exception of COSCO do not attempt to approximate greedy ideal. They approximate relevance based ranking. Relevance based ranking is the same as greedy ideal when selecting the first collection but relevance based ranking does not take overlap into account on subsequent selections. Therefore, when overlap does not exist then greedy ideal is the same as relevance based ranking; however, when overlap does exist then the difference between these two rankings shows how much effect overlap has on resource selection.

French and Powell [14] showed that most resource selection algorithms and especially gGLOSS and CVV inadvertently follow a size based ranking. Size based ranking selects the largest collection first and then the next largest and so on. Therefore, including the size based ranking in the experiments allowed an examination of whether the various methods seem to follow that ranking.

Finally, random resource selection provides a lower bound on long run performance. Any algorithm should outperform random in the long run. The difference between a given method and the random method shows the degree of improvement over the baseline performance.

4.2 Test Bed Creation

One of the goals of this thesis was to provide an accurate examination of the performance of the proposed solution in a variety of settings. Therefore, three important factors were chosen that describe various collections: the variability of the size of collections, the distribution of relevant documents, and the presence or absence of overlap. The first factor was chosen because it has been shown that some methods seem to perform poorly when collection sizes differ whereas others perform well with similar sized collections. The second factor is important because intuitively some collections are more relevant on some topics than others. Furthermore, relevance based methods assume that this is the case. Therefore, the effect of the distribution of relevant documents should be important to these methods. Finally, it is necessary to show the performance of the methods both with overlap and without it. Varying these three factors produces eight combinations. These combinations form the basis of the eight test beds included in the experiments.

In order to form the eight test beds, a large number of documents were required. Therefore, 38323 abstracts were obtained from various scientific abstract providers [1, 2, 11, 12, 25, 10, 13, 19, 21]. Each test bed has 100 collections within it. The difference between the test beds is how they distribute the abstracts amongst the collections. The collections are always presented in the same order which allows a quick inspection to determine the effect of size of various attributes of the test bed. The first two figures in each section describe the sizes of the contained collections as well as the average overlap of each collection with every other collection. The third figure shows the number of queries for which the collections could provide at least 1 result in the top-100 results. The figure graph shows the mean number of results that the collections have when they have at least 1 result. Finally, the last figure describes the performance of greedy ideal, relevance based ranking, size based ranking, and random ranking in terms percent top-k recall.

4.2.1 Test Bed 1: Same Size, Random Distribution, No Duplicates

The first test bed was created by distributing the abstracts in a round robin fashion. Therefore, each collection has roughly the same number of documents (Figure 4.1) and the documents were distributed randomly (Figures 4.3 and 4.4). Also, there are no duplicates since each document was assigned

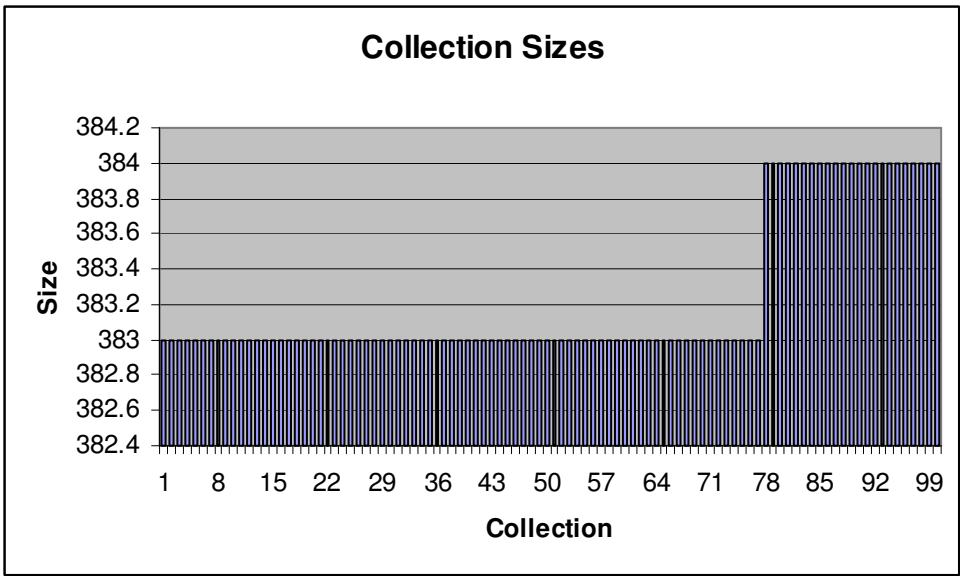

Figure 4.1: The collection sizes in test bed 1

to exactly one collection (4.2).

Figure 4.5 shows that with no overlap that greedy ideal and relevance based ranking perform identically. Also, since all collections are the same size, size based ranking performs similar to random ranking.

4.2.2 Test Bed 2: Same Size, Random Distribution, Duplicates

This test bed was created by randomly assigning 1000 documents to each collection. The documents were picked with replacement which leads to overlap (Figure 4.7), but note that the collections are the same size (Figure 4.6) and have a random distribution of relevant documents (Figures 4.8 and 4.9).

Figure 4.10 shows the effect that duplicates have on relevance based ranking. Again size based ranking does not perform well when all collections have the same size. Also note that random ranking is not linear in the presence of overlap.

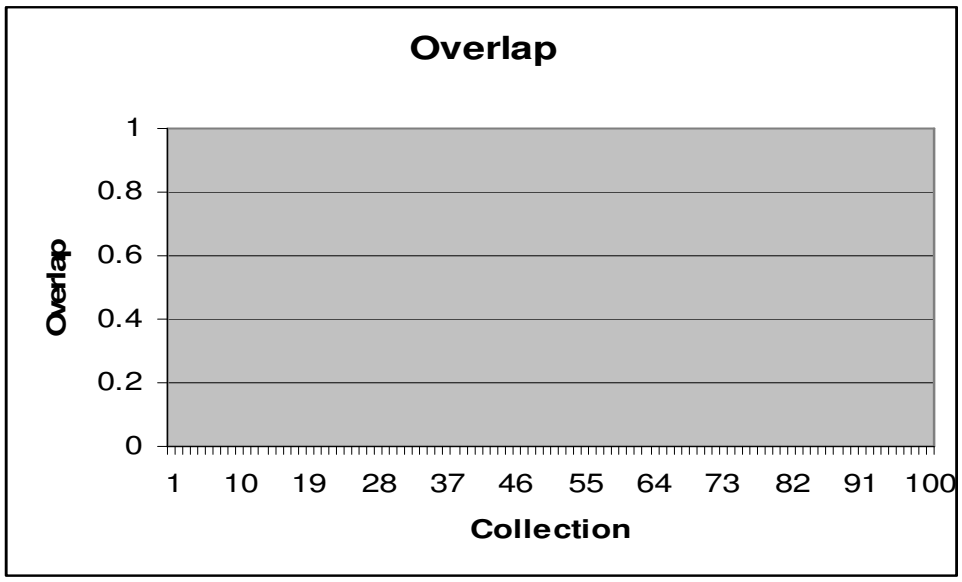

Figure 4.2: The overlap in test bed 1

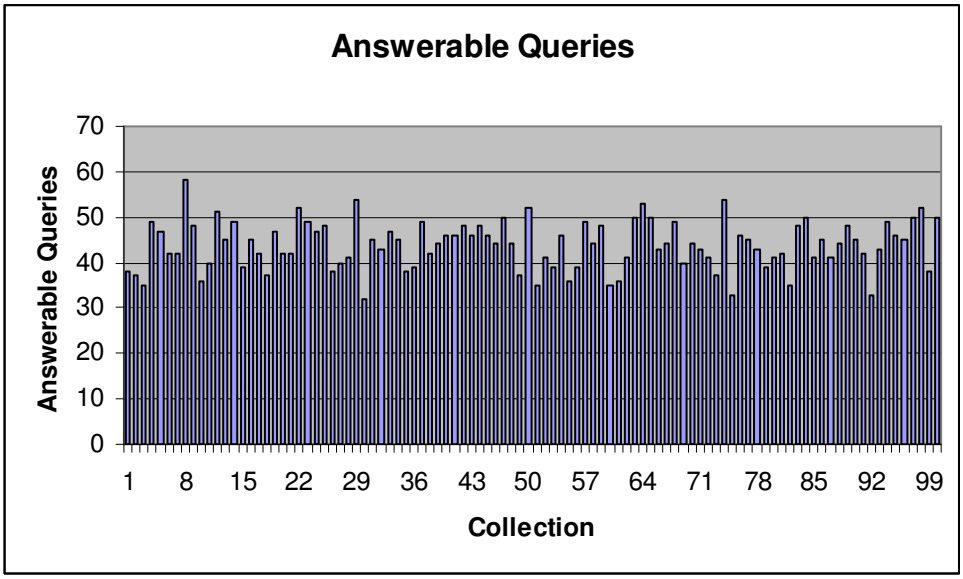

Figure 4.3: The number of answerable queries in test bed 1

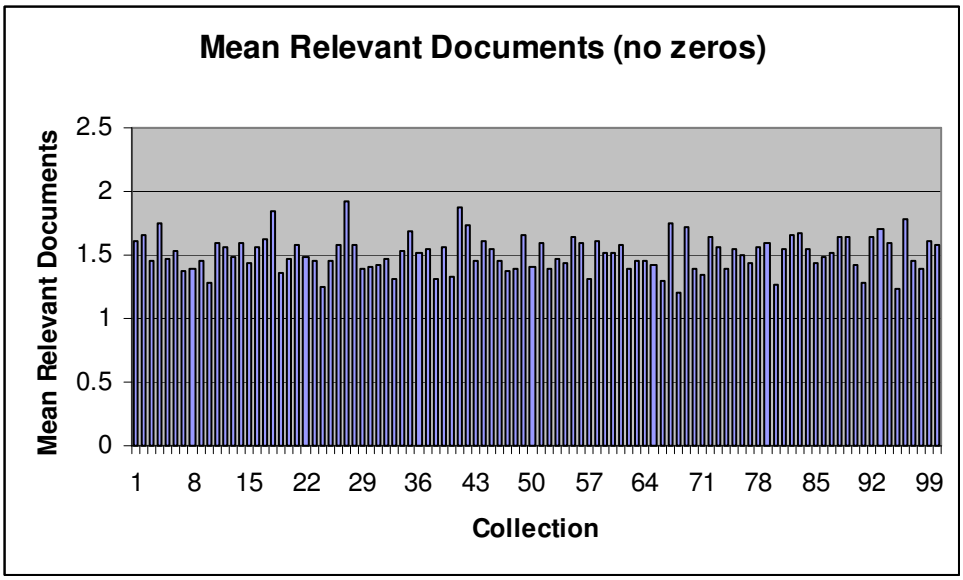

Figure 4.4: The mean number of relevant results for answerable queries in test bed 1

4.2.3 Test Bed 3: Varying Size, Random Distribution, No Duplicates

The third test bed also used a round robin assignment but assigned varying numbers of relevant documents to each collection (Figure 4.11). With most of the documents going to only a handful of collections. Since assignment did not include replacement, there are no duplicates (Figure 4.12). Also, the assignments were random so relevance is randomly distributed (Figures 4.13 and 4.14).

Figure 4.15 shows several important points regarding test bed 3. Note that greedy ideal and relevance based ranking perform equally well since there is no overlap; however, size based ranking significantly outperforms random because the collections vary in size and documents are randomly distributed. Clearly, larger collections have a better chance of containing relevant documents.

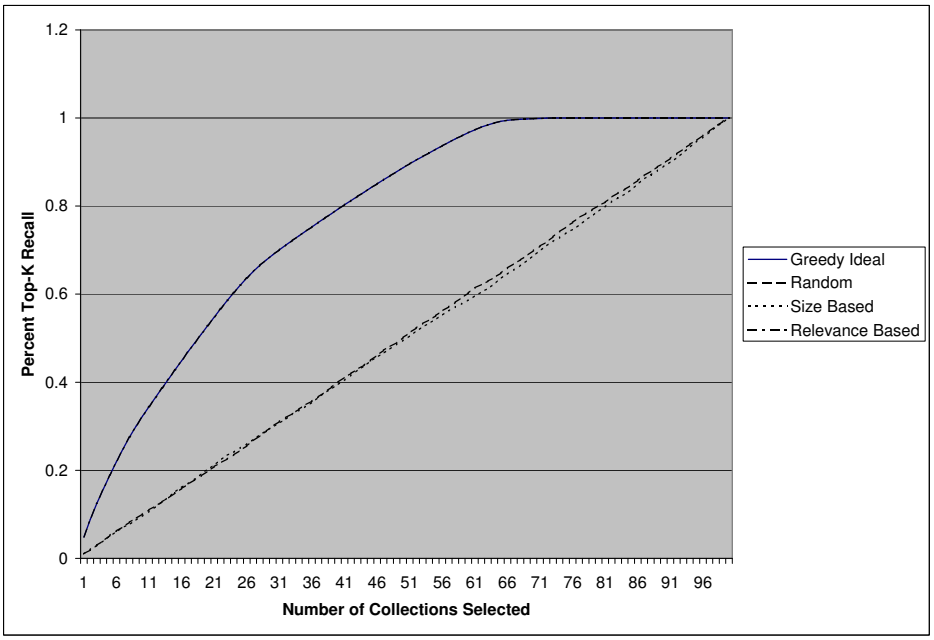

Figure 4.5: Percent recall performance in test bed 1

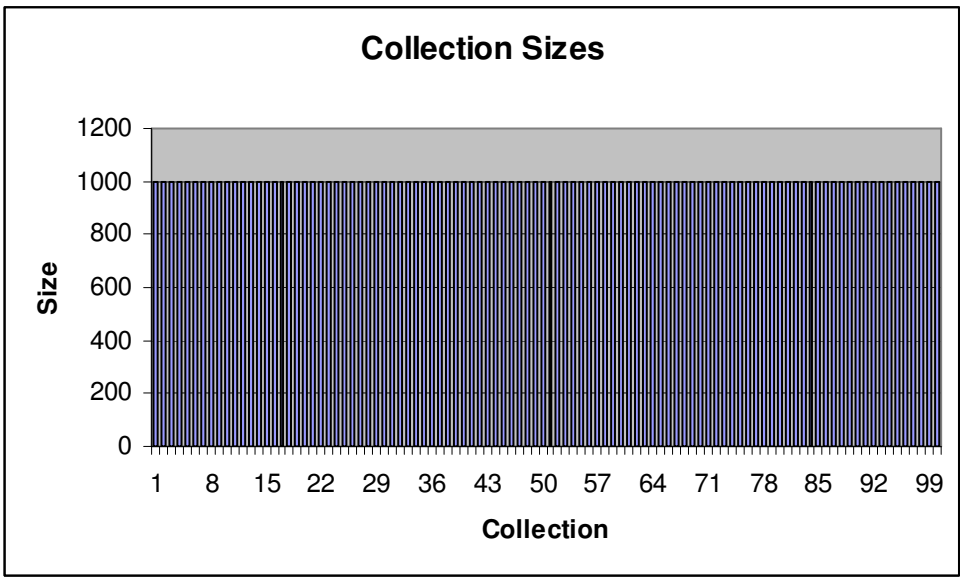

Figure 4.6: The collection sizes in test bed 2

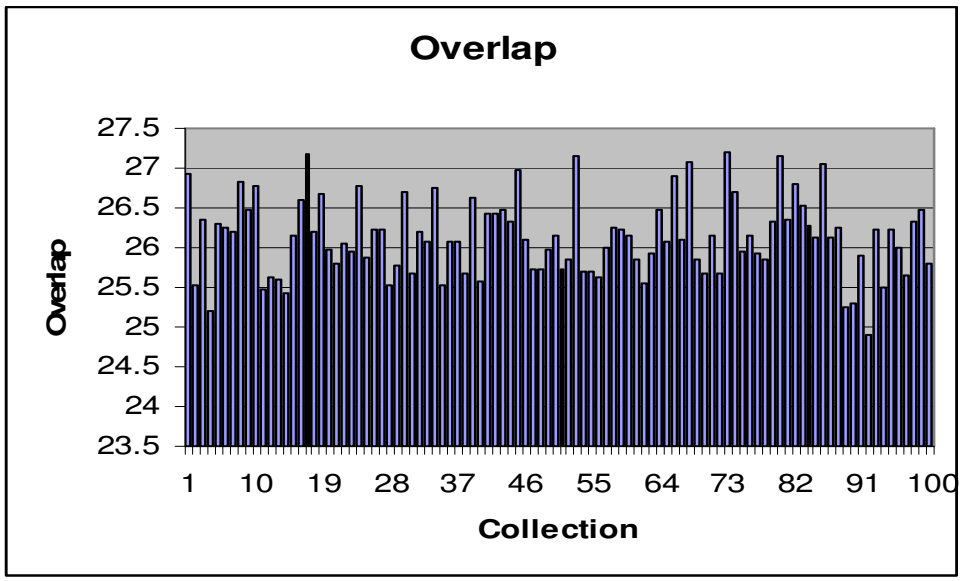

Figure 4.7: The overlap in test bed 2

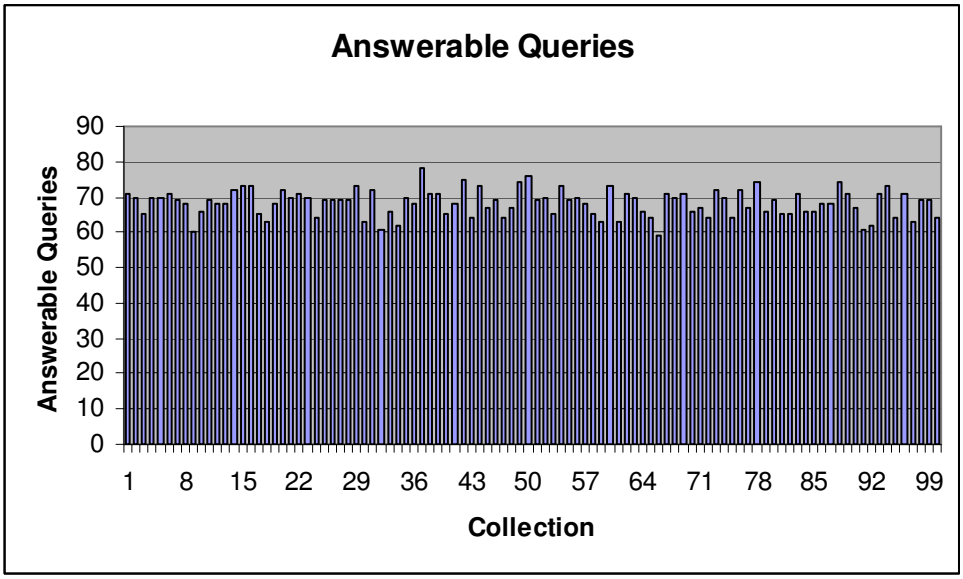

Figure 4.8: The number of answerable queries in test bed 2

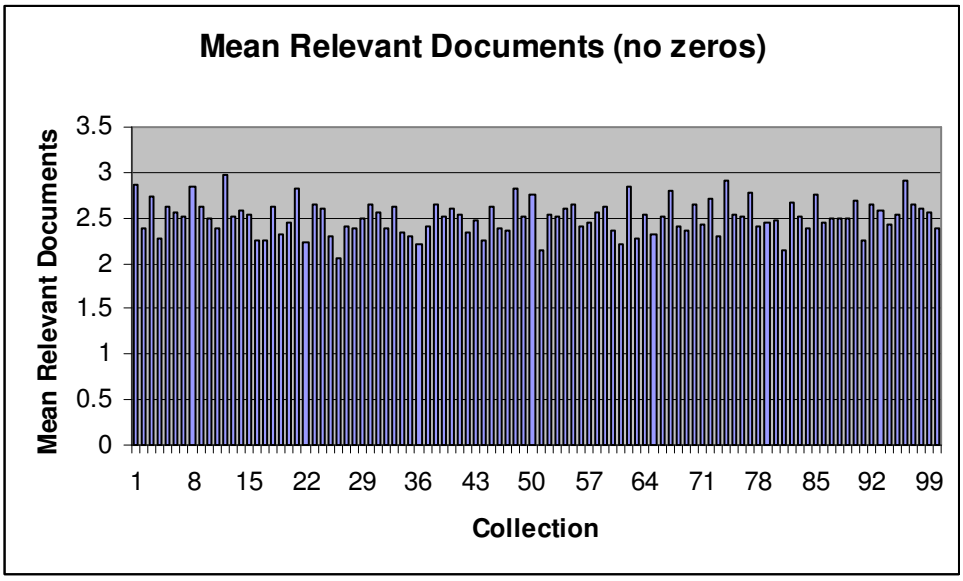

Figure 4.9: The number of relevant results for answerable queries in test bed 2

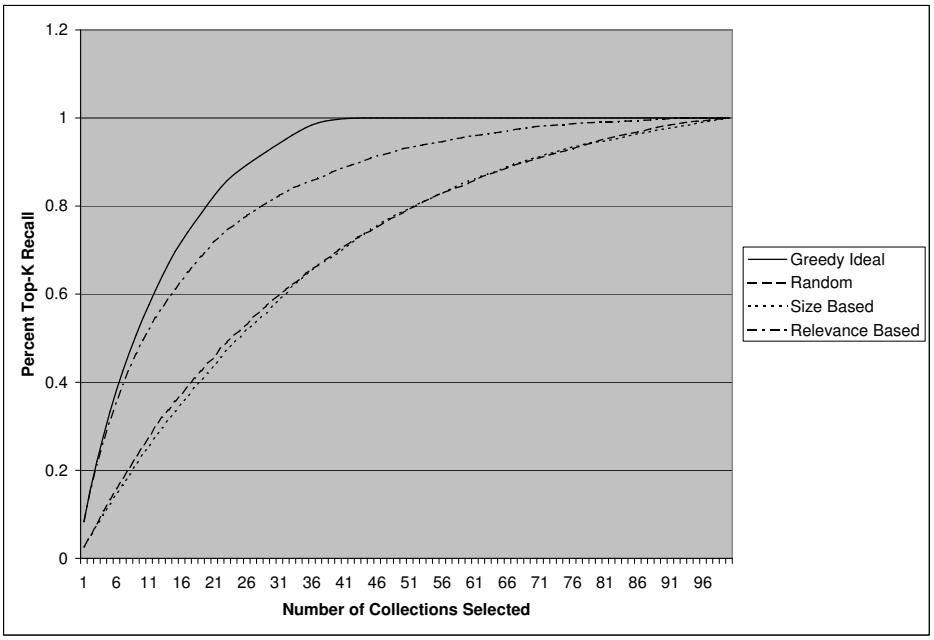

Figure 4.10: Percent recall performance in test bed 2

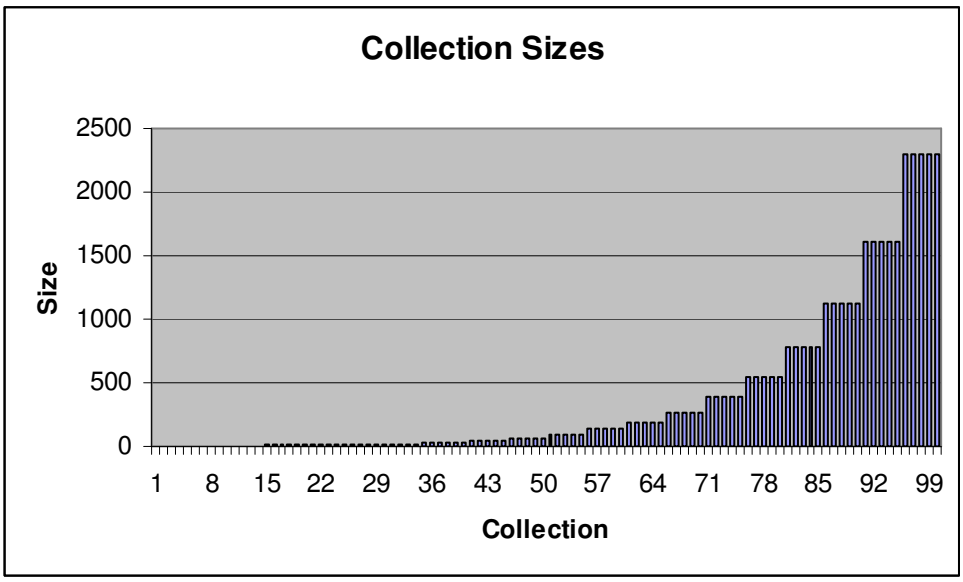

Figure 4.11: The collection sizes in test bed 3

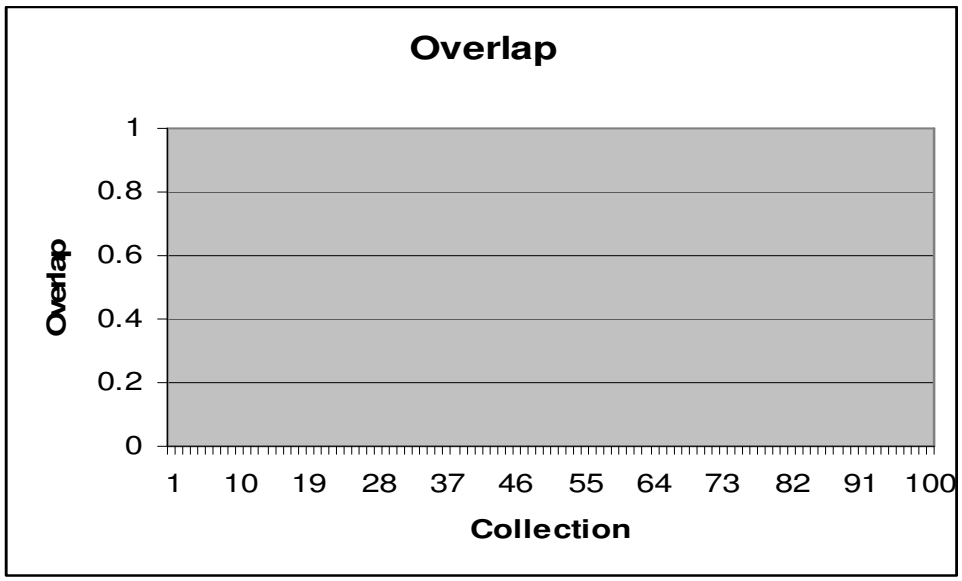

Figure 4.12: The overlap in test bed 3

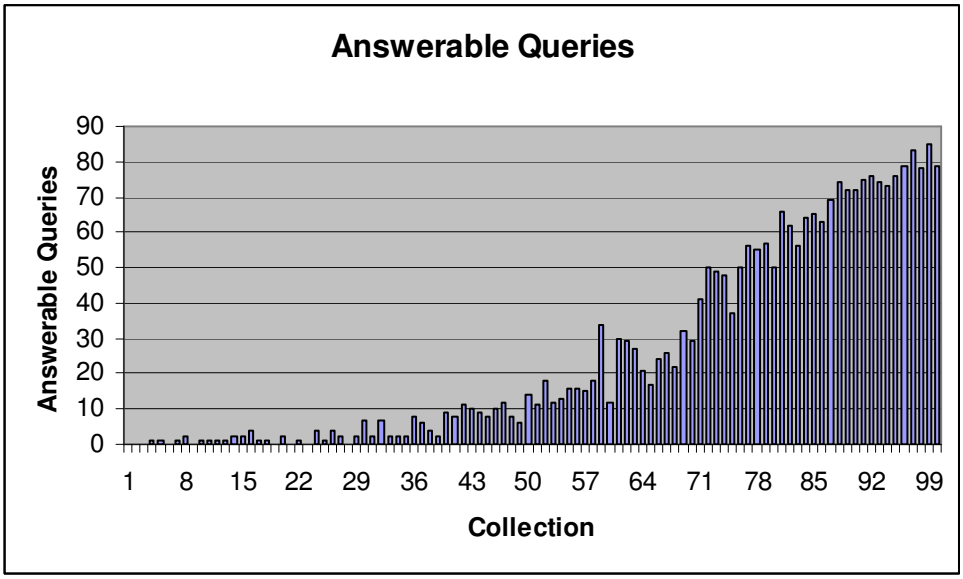

Figure 4.13: The number of answerable queries in test bed 3

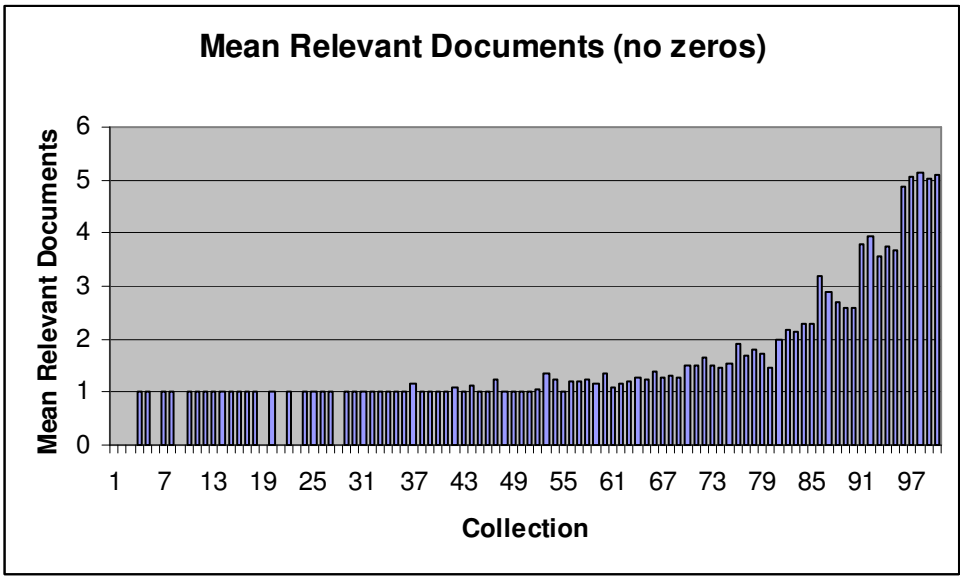

Figure 4.14: The number of relevant results for answerable queries in test bed 3

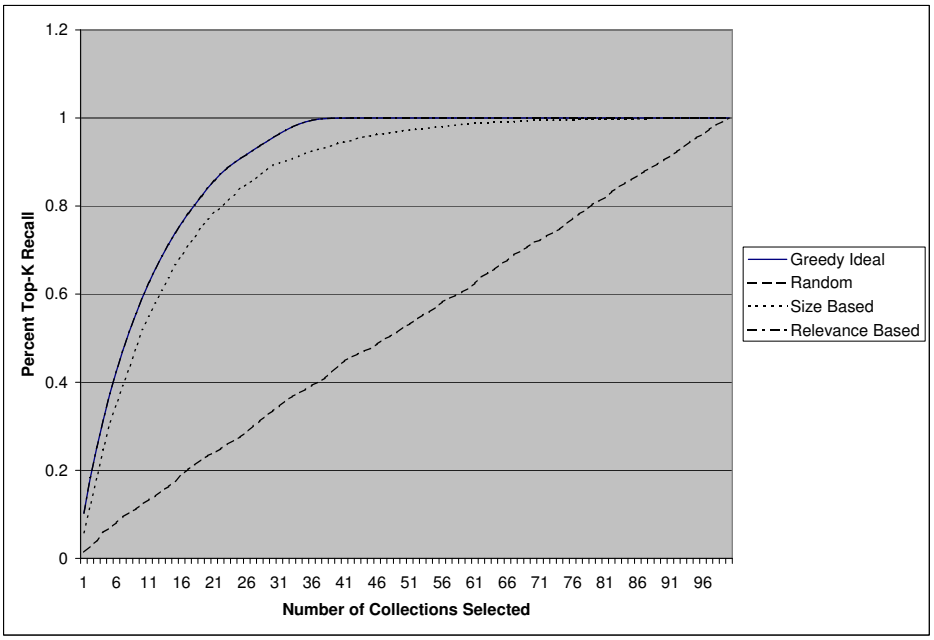

Figure 4.15: Percent recall performance in test bed 3

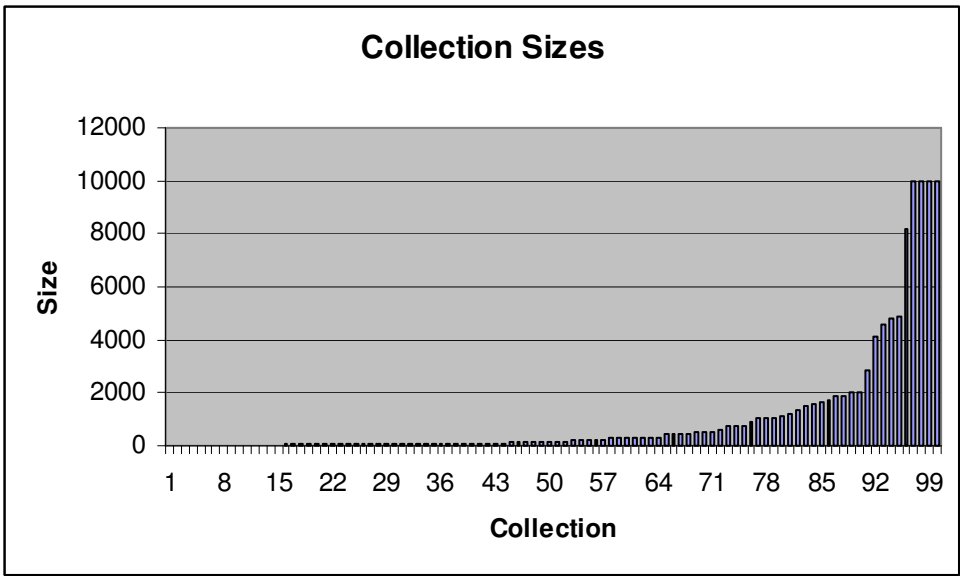

Figure 4.16: The collection sizes in test bed 4

4.2.4 Test Bed 4: Varying Size, Random Distribution, Duplicates

This test bed randomly picked a size for each test bed and then randomly picked the documents with replacement from the total pool of documents. The sizes were picked in an exponential fashion thereby yielding significant differences in size (Figure 4.16). Also note that there is overlap in this test bed (Figure 4.17). Again, the distribution of relevance is random in this test bed (Figures 4.18 and 4.19).

This test bed is interesting (Figure 4.20) because several of the collections contained over a fourth of the total documents each. This means that most of the documents can be accessed by querying just the largest collections. This is shown by the excellent performance of sized based ranking.

4.2.5 Test Bed 5: Same Size, Clustered Distribution, No Duplicates

In order to create this test bed, k-means clustering was used on the pool of documents to create 250 clusters. Then clusters were combined together to

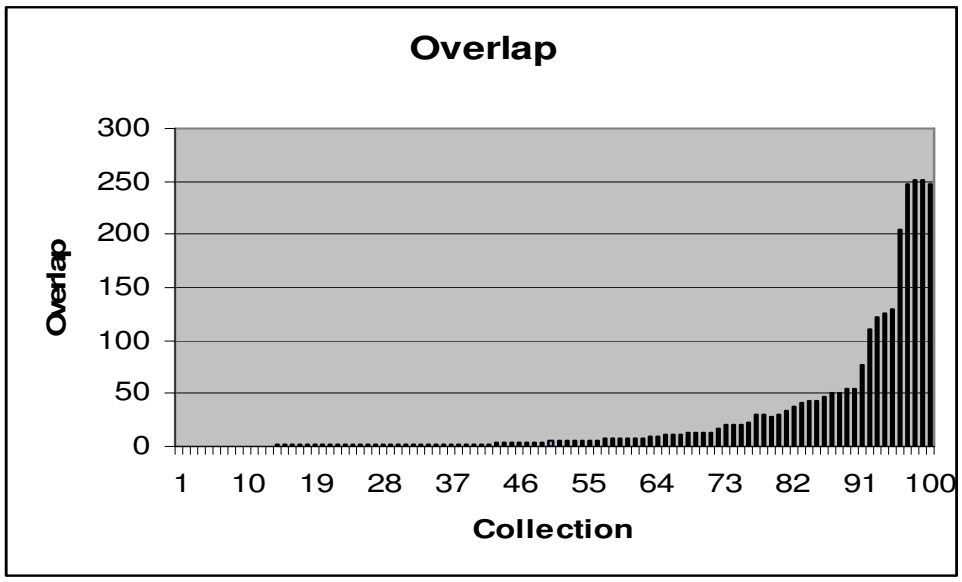

Figure 4.17: The overlap in test bed 4

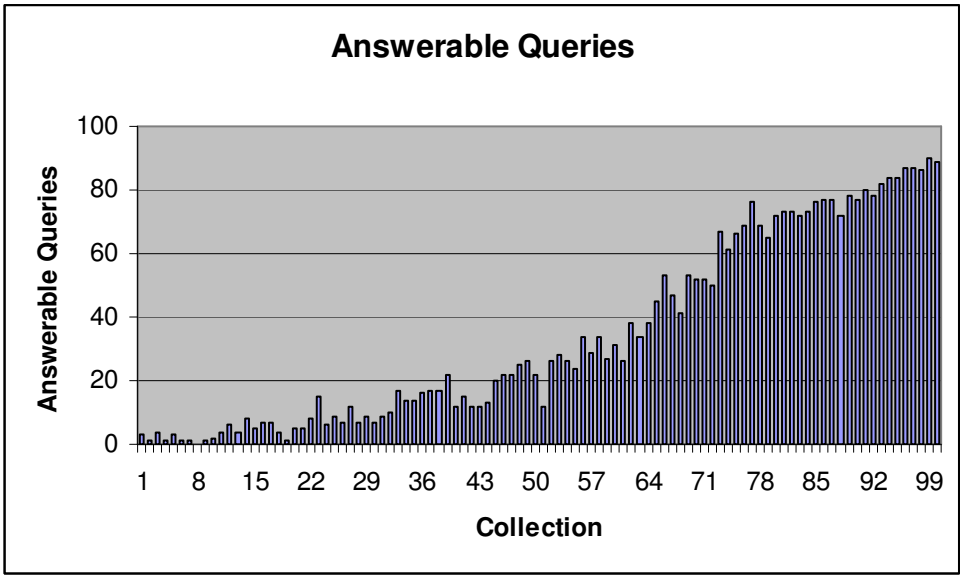

Figure 4.18: The number of answerable queries in test bed 4

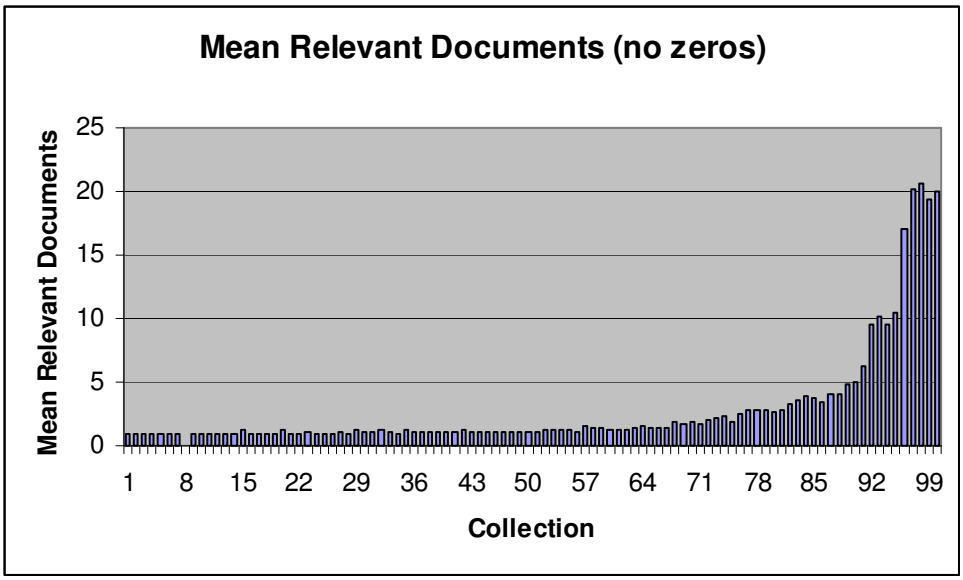

Figure 4.19: The number of relevant results for answerable queries in test bed 4

form 100 roughly equivalent size test beds (Figure 4.21). In reality, the sizes of the collections appear to be normally distributed but this is close enough for the purposes of the experiment. The k-means clustering serves to provide varying relevance (Figures 4.23 and 4.24) among collections with respect to various queries. Also note that there is no overlap (Figure 4.22).

Figure 4.25 shows that greedy ideal performs the same as relevance based ranking as expected. Size based ranking performs nearly identical to random ranking because collections are roughly the same size. Random ranking is linear because there is no overlap.

4.2.6 Test Bed 6: Same Size, Clustered Distribution, Duplicates

This test bed also used k-means clustering to create 250 clusters; however, it randomly assigned clusters with replacement to create 100 collections of nearly identical size (Figure 4.26). Therefore, there is overlap in this test bed (Figure 4.27) and clustered distribution (Figures 4.28 and 4.29).

In this test bed, the effect of overlap can be clearly seen by the disparity

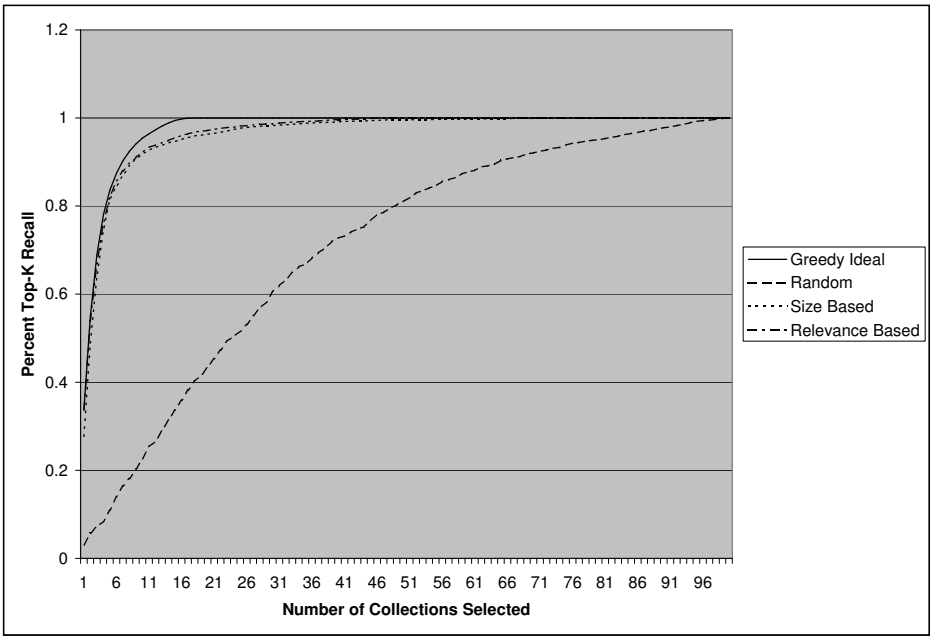

Figure 4.20: Percent recall performance in test bed 4

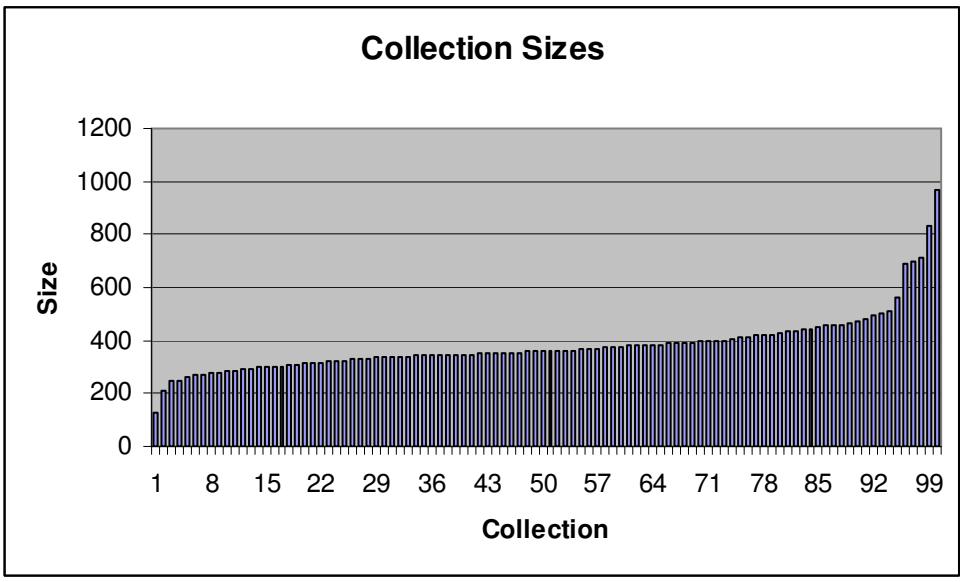

Figure 4.21: The collection sizes in test bed 5

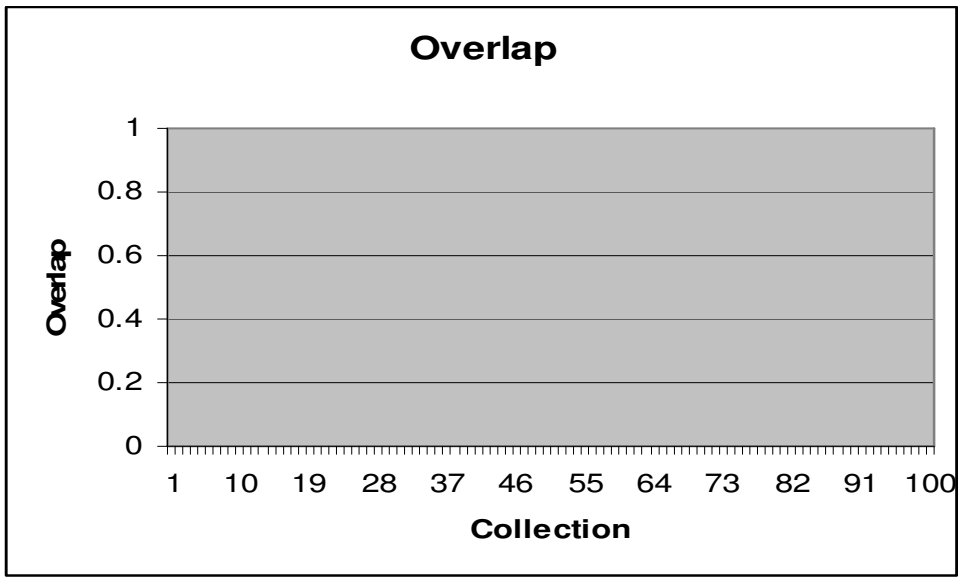

Figure 4.22: The overlap in test bed 5

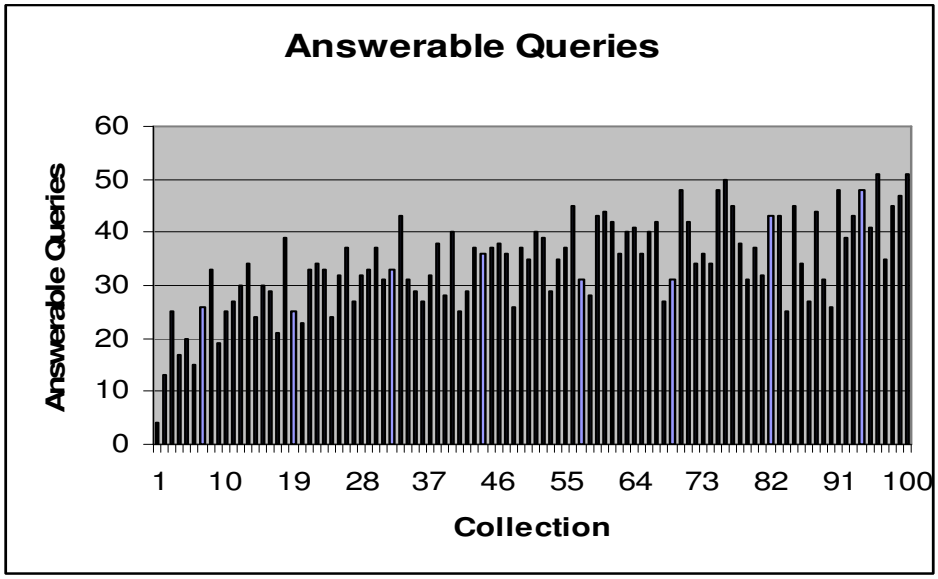

Figure 4.23: The number of answerable queries in test bed 5

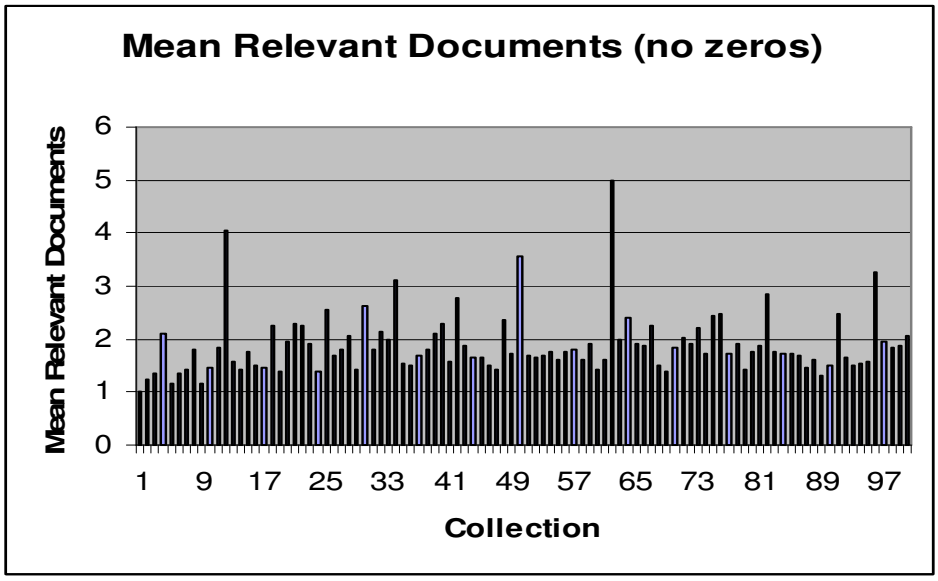

Figure 4.24: The number of relevant results for answerable queries in test bed 5

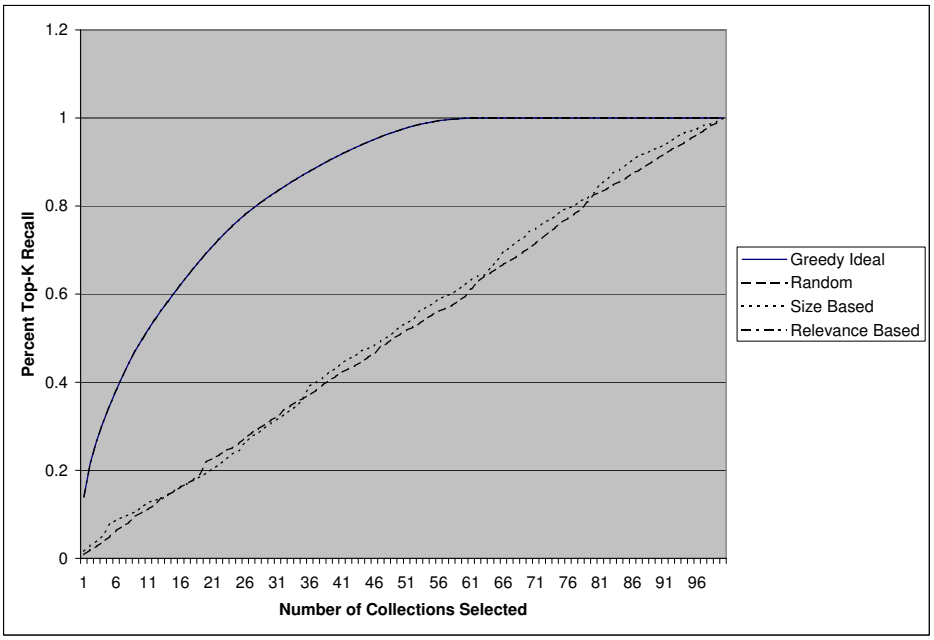

Figure 4.25: Percent recall performance in test bed 5

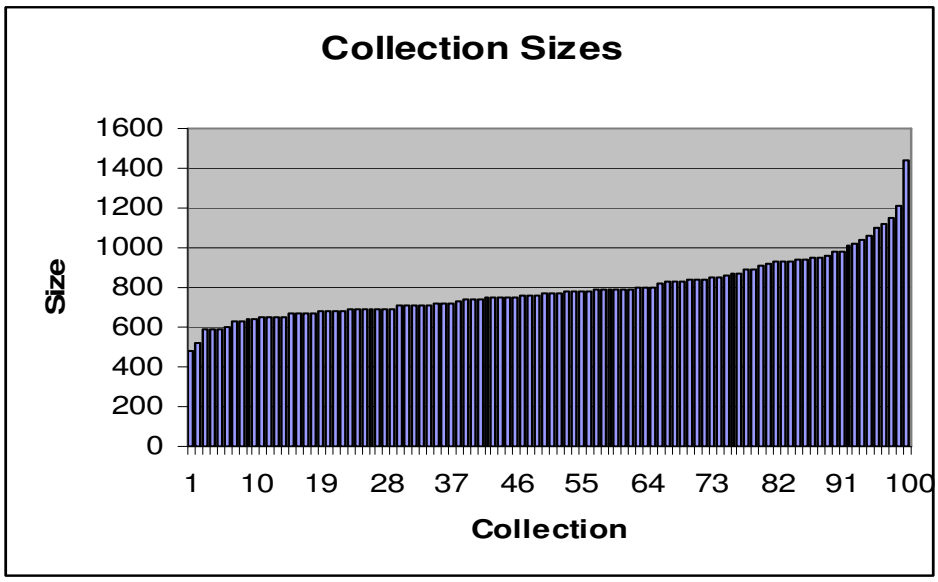

Figure 4.26: The collection sizes in test bed 6

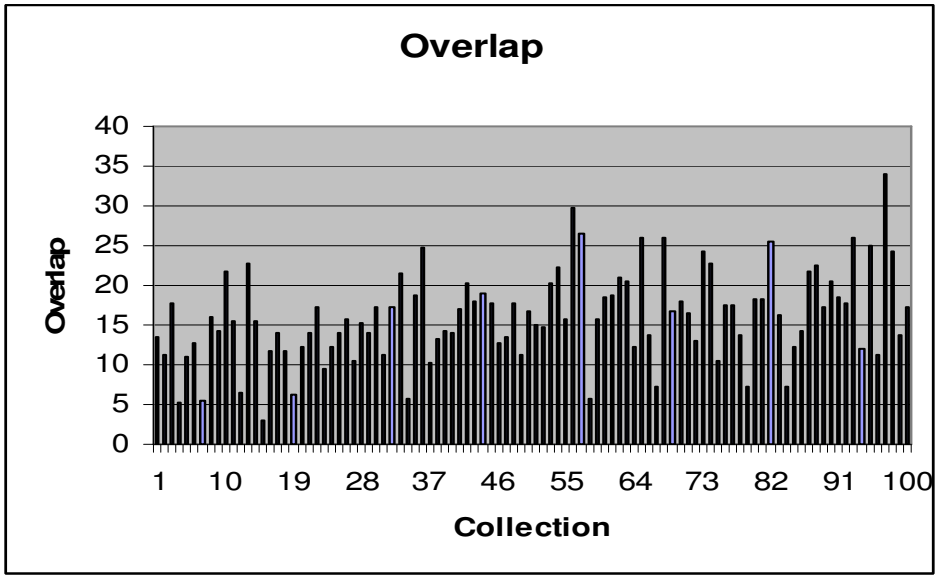

Figure 4.27: The overlap in test bed 6

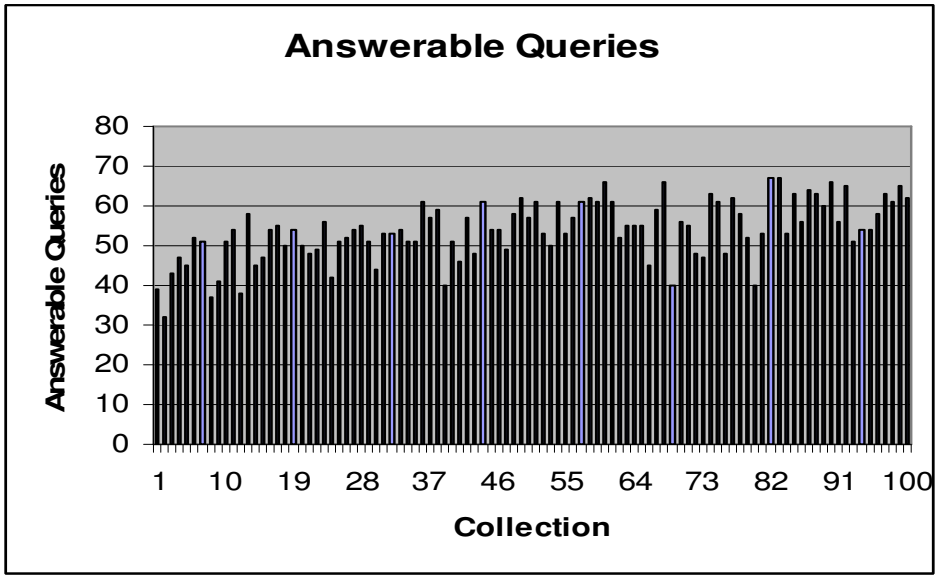

Figure 4.28: The number of answerable queries in test bed 6

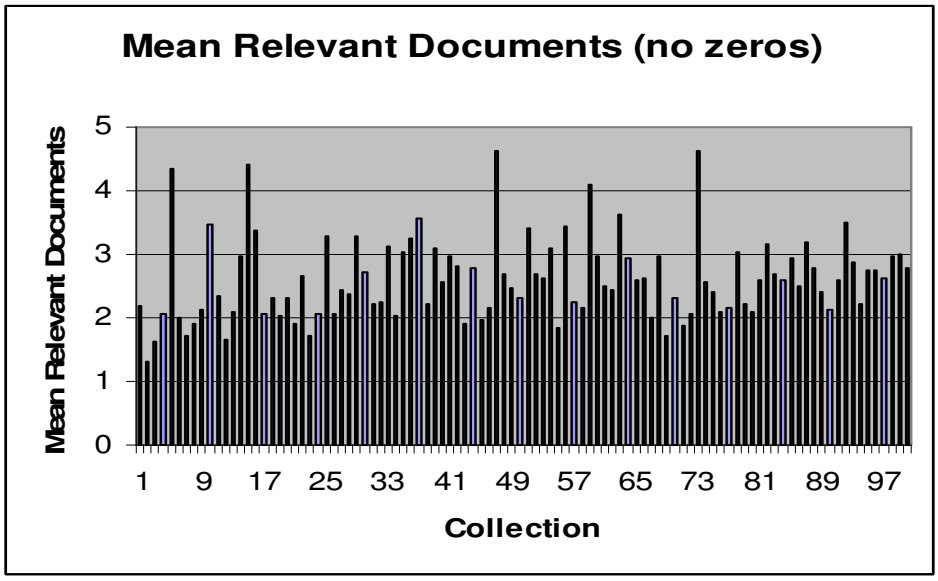

Figure 4.29: The number of relevant results for answerable queries in test bed 6

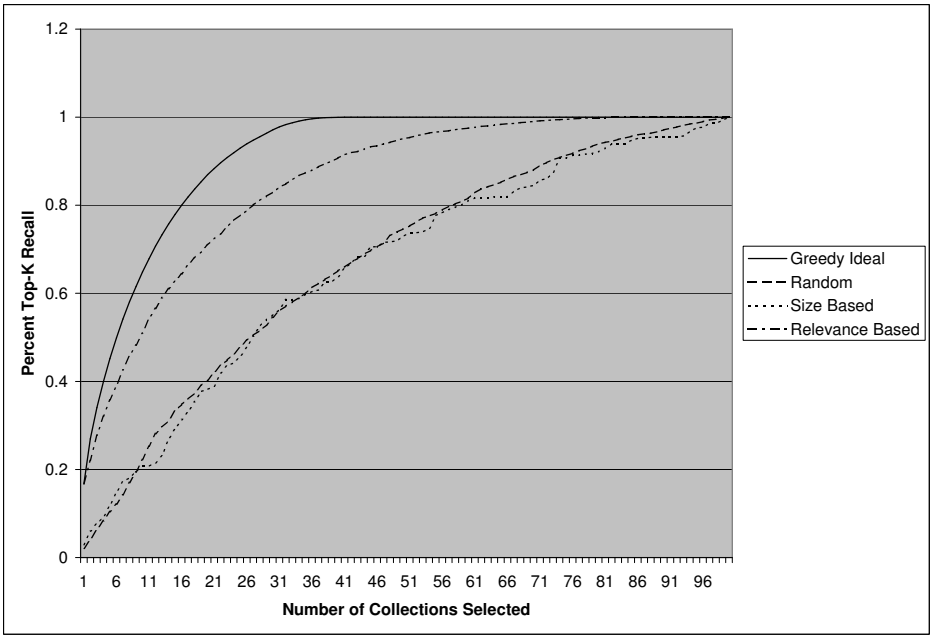

Figure 4.30: Percent recall performance in test bed 6

between greedy ideal and relevance based ranking in figure 4.30. Since collection sizes are approximately the same, sized based ranking and random ranking perform similarly.

4.2.7 Test Bed 7: Varying Size, Clustered Distribution, No Duplicates

To create this test bed, k-means was used to cluster the documents but only 100 clusters were created. These clusters became the 100 collections for the test bed. Thus, there is no overlap (Figure 4.32) and the collection sizes vary (Figure 4.31). Also, the collections vary in relevance (Figures 4.33 and 4.34).

Figure 4.35 shows that greedy ideal and relevance based ranking perform identically while size based ranking performs significantly better than random ranking because of the variable size.

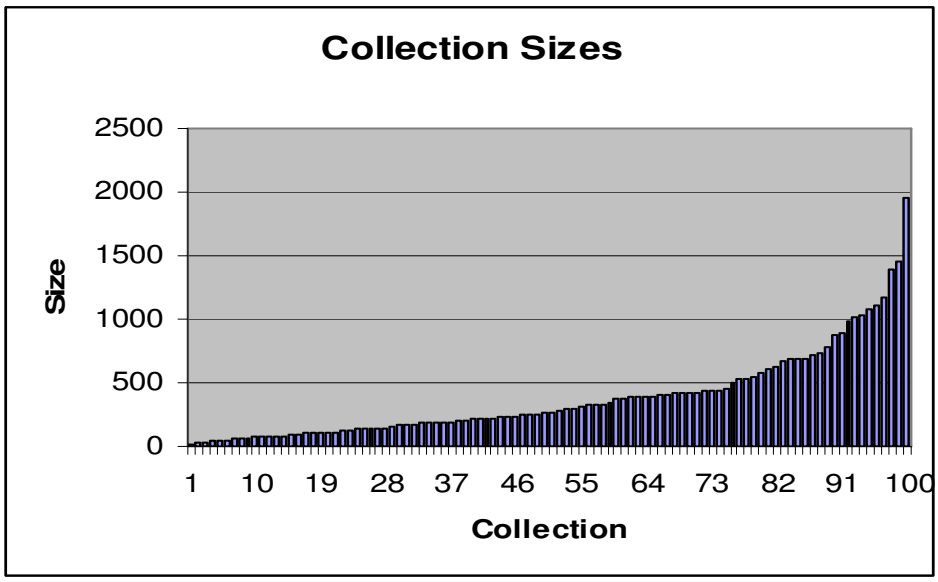

Figure 4.31: The collection sizes in test bed 7

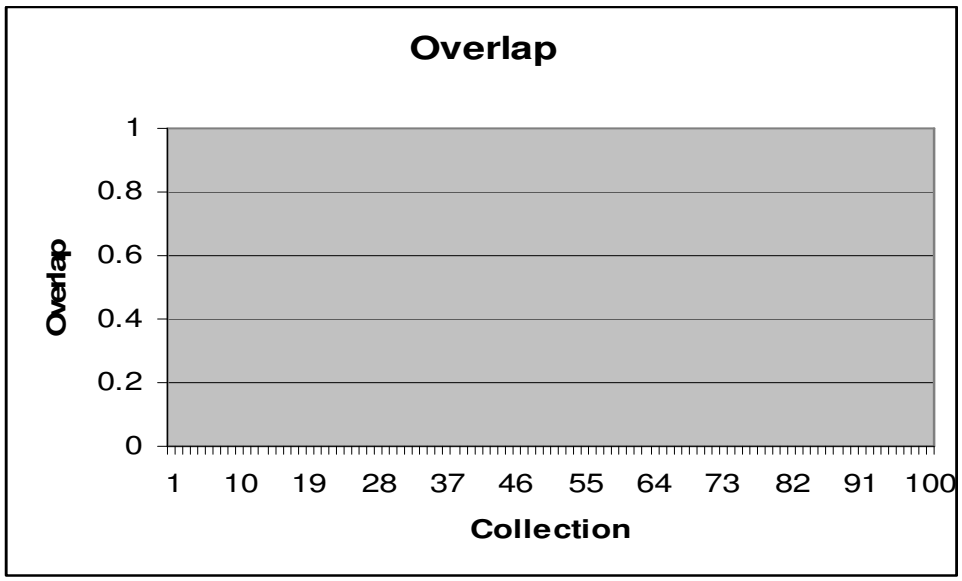

Figure 4.32: The overlap in test bed 7

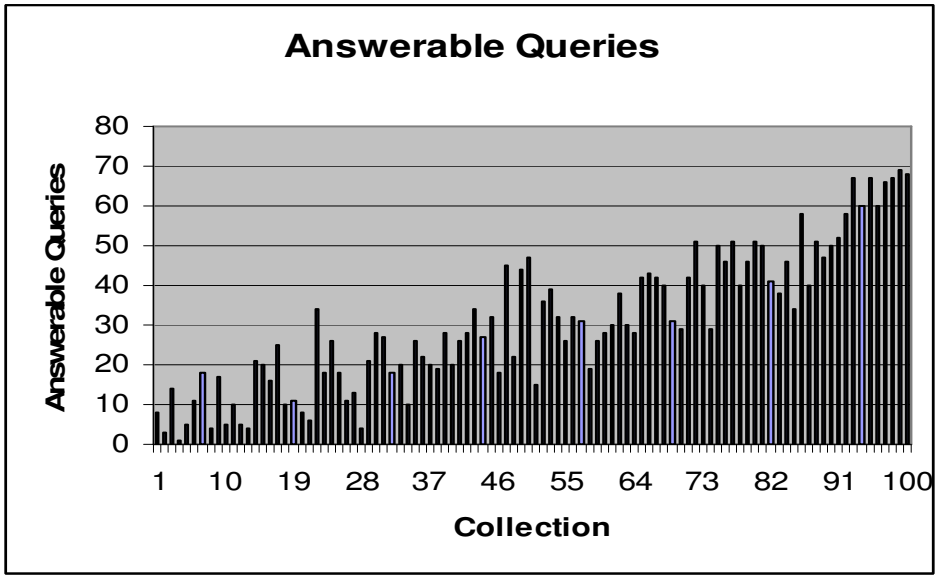

Figure 4.33: The number of answerable queries in test bed 7

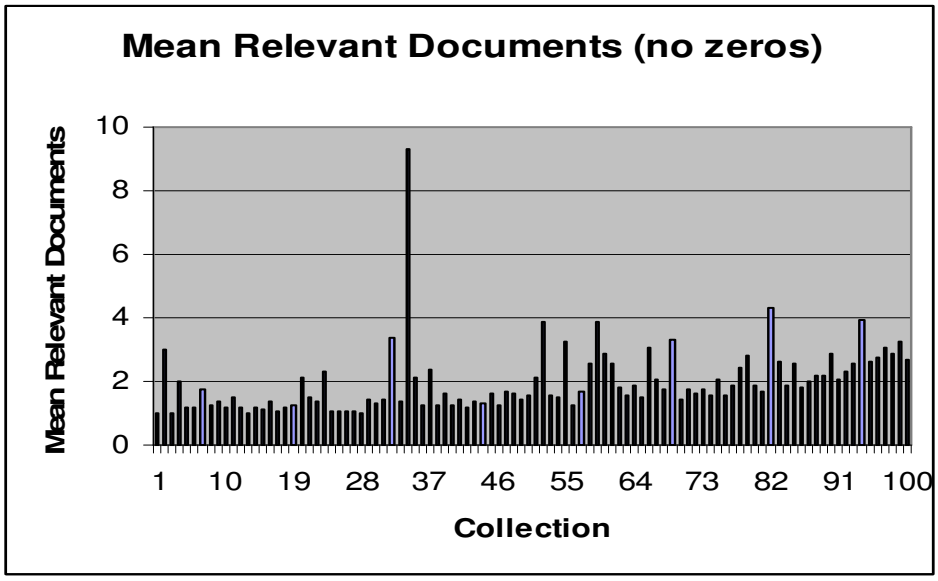

Figure 4.34: The number of relevant results for answerable queries in test bed 7

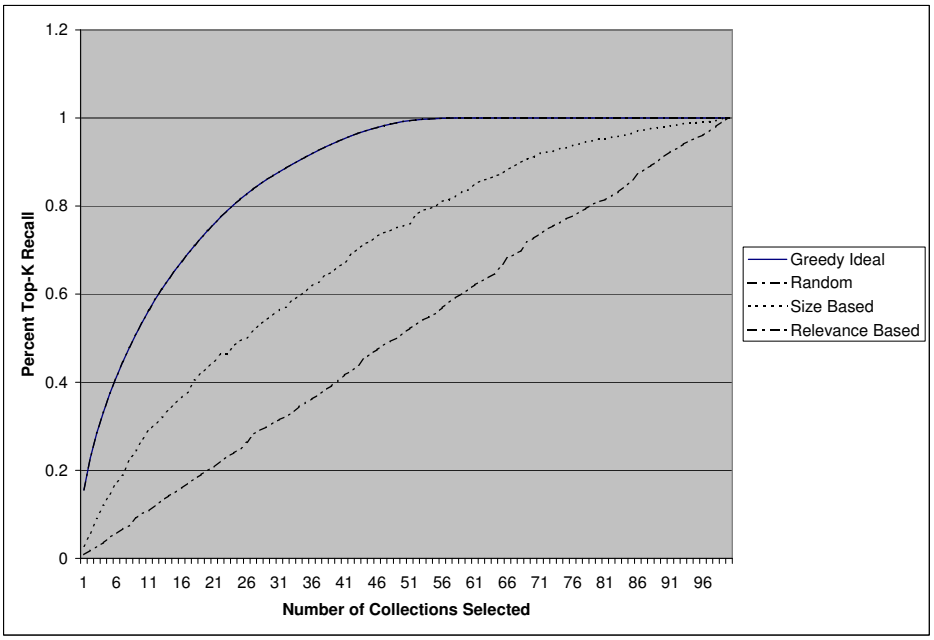

Figure 4.35: Percent recall performance in test bed 7

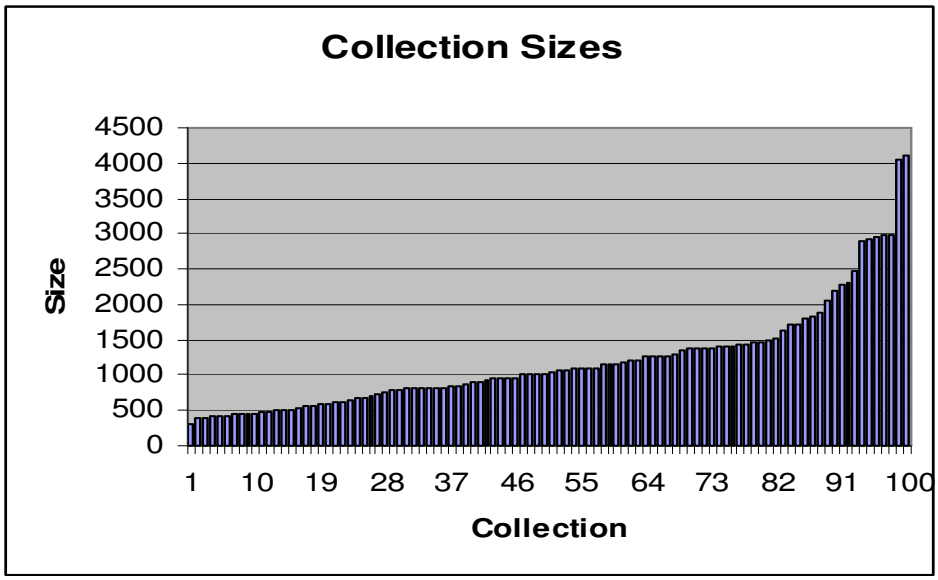

Figure 4.36: The collection sizes in test bed 8

4.2.8 Test Bed 8: Varying Size, Clustered Distribution, Duplicates

The last test bed was also created using k-means clustering with 250 clusters. This time though, each collection was randomly assigned 3 clusters with replacement. The sizes of the collections vary quite a bit (Figure 4.36) and there is overlap between the collections (Figure 4.37). Also, it is a clustered distribution so relevance varies from collection to collection according to the query (Figures 4.38 and 4.39).

This test bed is particularly interesting because it is probably most like real world scenarios. Figure 4.40 shows that overlap does have an effect while sized based ranking improves on random ranking because of the variable sizes.

4.3 Tested Methods

In order to demonstrate the efficiency of ROSCO, it is compared to three other methods. The other methods that are examined COSCO [18], ReDDE [28], and CORI [8]. In the following sections, the implementation of each

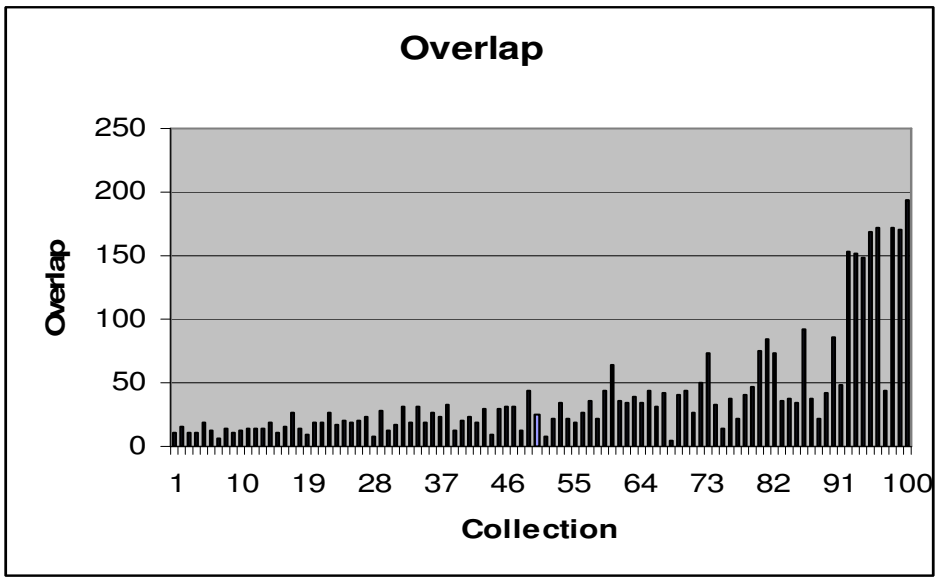

Figure 4.37: The overlap in test bed 8

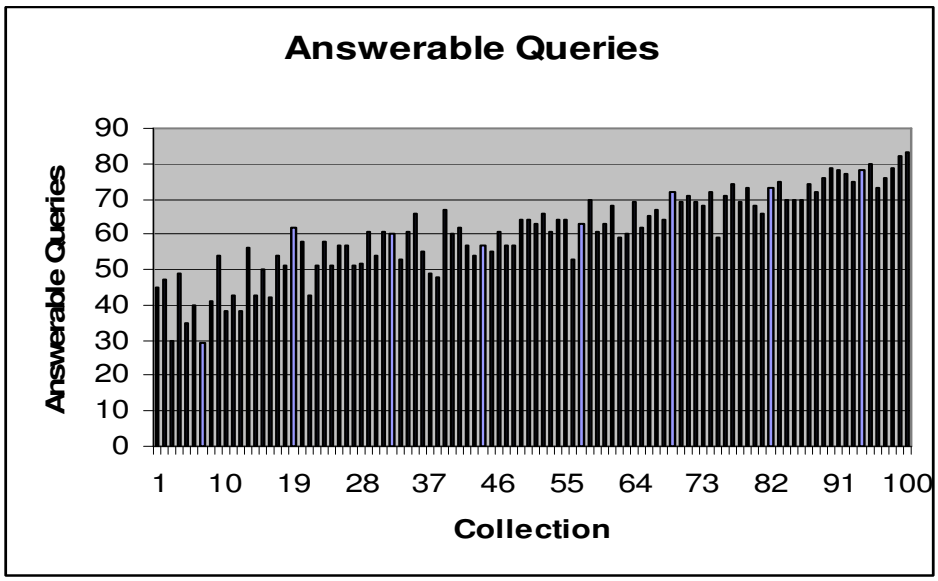

Figure 4.38: The number of answerable queries in test bed 8

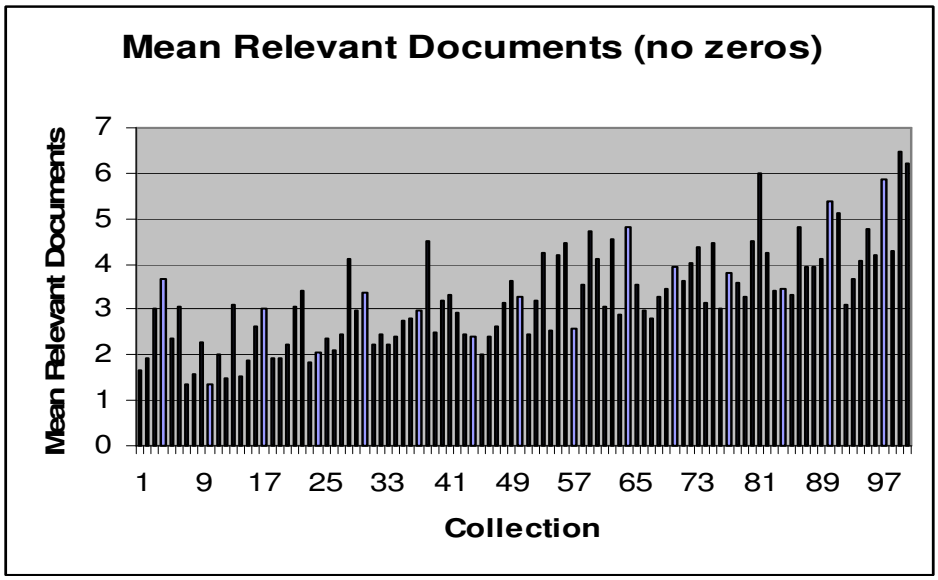

Figure 4.39: The number of relevant results for answerable queries in test bed 8

method is briefly discussed.

4.3.1 ROSCO

The offline component of ROSCO was implemented as described previously. A large set of queries from the Bibfinder system [5] was used as the training queries. Each collection in each test bed was sampled by using 10 randomly selected training queries. The samples were used to build the representative. Next, 10 size estimates were made for each collection. The final size estimate is the mean of these estimates. Finally, queries that appeared more than 5 times were used in the frequent item set computation. A support value of .05% was required during the Apriori algorithm. For the purposes of these experiments overlap meant duplicate documents.

4.3.2 COSCO

ROSCO and COSCO [18] require the same set of statistics. So these two methods shared their statistics for the experiments. ROSCO differs from

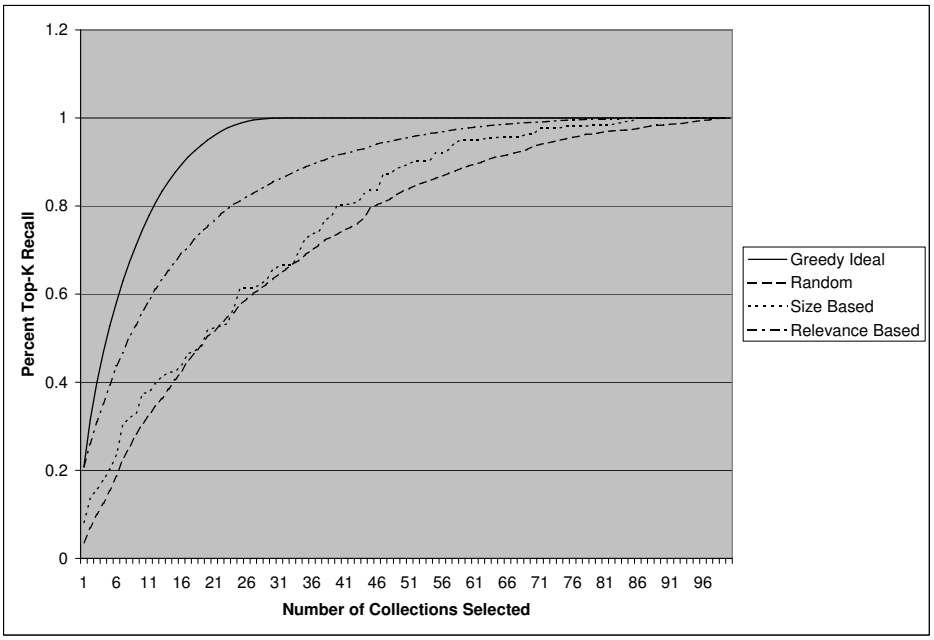

Figure 4.40: Percent recall performance in test bed 8

COSCO in that it uses top-k relevance and overlap during its first phase while COSCO always considers coverage and overlap. The second phase of ROSCO is identical to COSCO. So COSCO does not use the size estimates and relevance representative.

4.3.3 ReDDE

ReDDE [28] and ROSCO use the same set of statistics so these were also shared. ROSCO contrasts with ReDDE because it considers overlap. ReDDE does not use the overlap statistics. ReDDE has no notion of residual relevance and all calculations are done with the assumption that every document (or document class) belongs to precisely one collection.

4.3.4 CORI

Until recently, CORI [8] has been widely accepted as the best, most stable method for resource selection; hence, it was included in this study. CORI models each collection as a virtual document. It can be viewed as a *df-icf* method where *df* is the document frequency of a term within a collection and *icf* is the inverse collection frequency of a term. Single source text retrieval is performed over the collection of these virtual documents to determine the order in which the collections should be called.

CORI used the same sample as ROSCO and ReDDE. Once this sample was obtained then the document frequency or the number of documents containing each term in a collection was determined. Also, the collection frequency of a term was determined by finding the number of collections in the test bed which contain the term. CORI forms a belief network associated with each term.

$$P(t_k|C_i) = 0.4 + 0.6 \cdot T \cdot I \tag{4.2}$$

$$T = \frac{df}{df + 50 + 150 * \frac{cw}{\bar{cw}}} \tag{4.3}$$

$$I = \frac{log(\frac{|C|+0.5}{cf})}{log(|C| + 1.0)} \tag{4.4}$$

In these equations, *df* is the number of documents in collection C_i containing term t_k whereas cf is the number of collections in the test bed containing

term t_k. $|C|$ denotes the number of collections in the test bed and cw is the number of terms in collection C_i. Finally, $\bar{cw}$ is the average cw for all of the collections.

4.4 Testing

For the experiments, 100 queries which were disjoint from the training queries were sent to each method. The percent recall at each step was determined for every method. Then the results were averaged in order to provide a clear look at performance. In the next section, the results of these experiments are described in detail.

4.5 Experimental Results

The results show that ROSCO clearly outperforms the other methods when selecting a small subset of collections (5% - 25%). Figure 4.41 shows that ROSCO performs consistently higher than all of the methods except CORI which appears quite unstable. An interesting note is that in this test bed, CORI outperforms greedy ideal over a small range. Note that this is not a contradiction because it is the *greedy* ideal. It is possible for another method to outperform greedy ideal over a small range; however, over the long run this is not possible if the method is also greedy. Also note that CORI appears quite unstable over varying size subsets.

Figure 4.42 shows that in the presence of overlap that CORI's performance suffers dramatically. Furthermore, in this case ROSCO outperforms all of the other methods.

Figures 4.43 and 4.44 show the performance of the methods in test beds 3 and 4 where collection sizes vary and the distribution of relevant documents is random. This is probably not like real world scenarios since collections most likely do not have randomly assigned documents but are authoritative on certain topics. In these test beds, both ROSCO and ReDDE suffer in the beginning. COSCO performs very well but not much better than size based ranking does. Finally, CORI performs very poorly especially in the presence of overlap.

In test bed 5, figure 4.45 shows that ROSCO outperforms all of the other methods. Furthermore, CORI and COSCO suffer in this testbed presumably

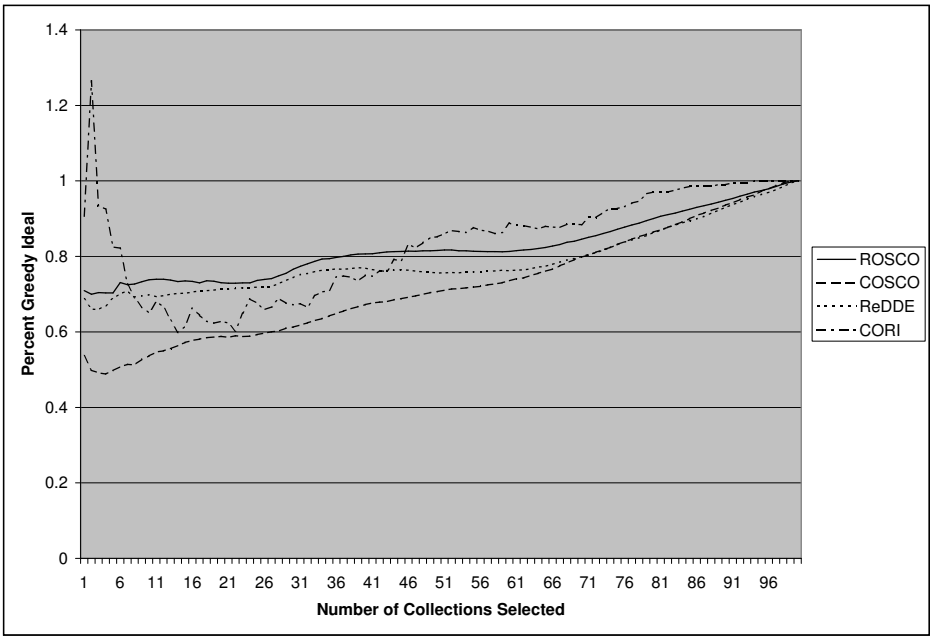

Figure 4.41: Percent of greedy ideal for test bed 1

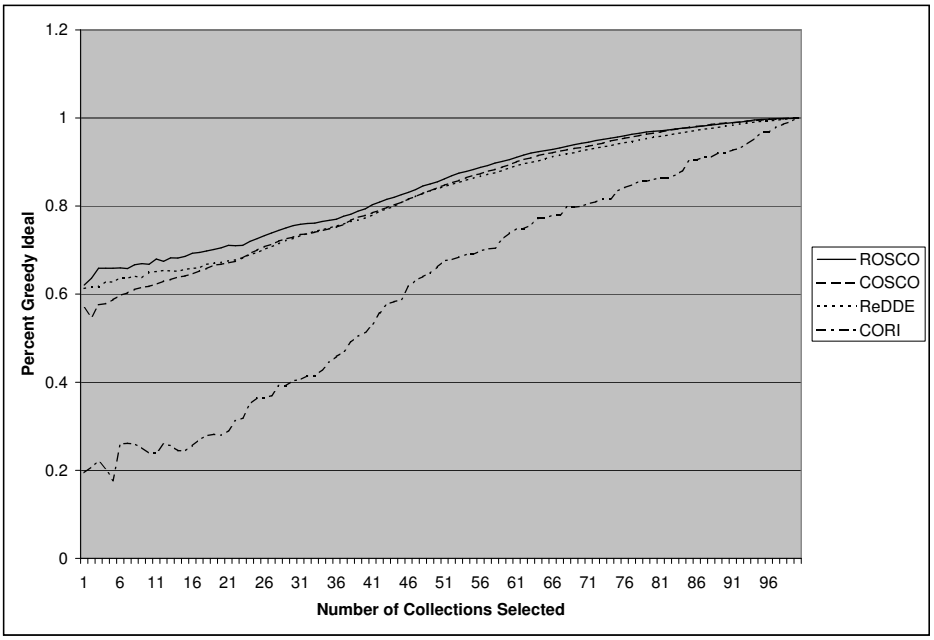

Figure 4.42: Percent of greedy ideal for test bed 2

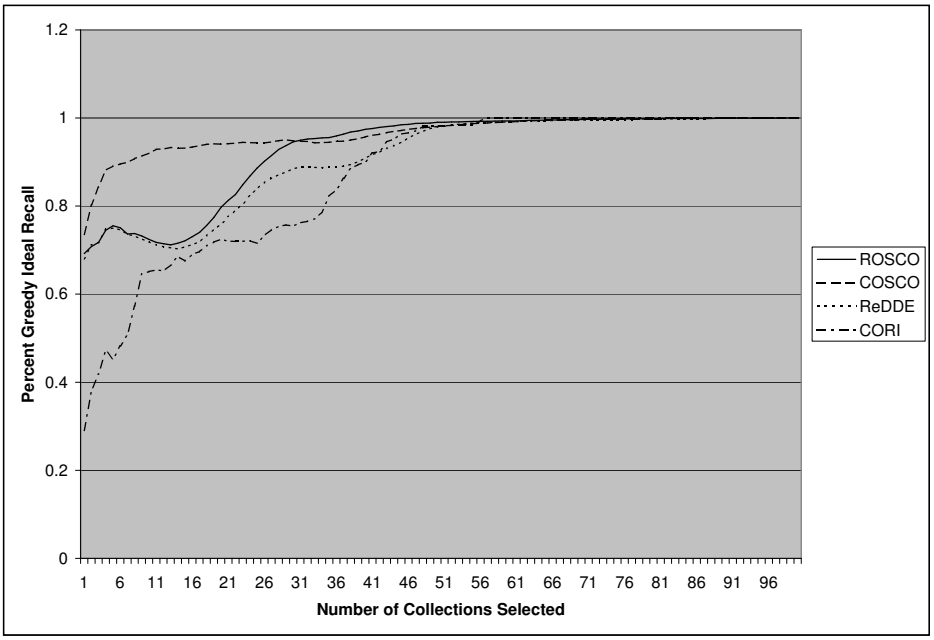

Figure 4.43: Percent of greedy ideal for test bed 3

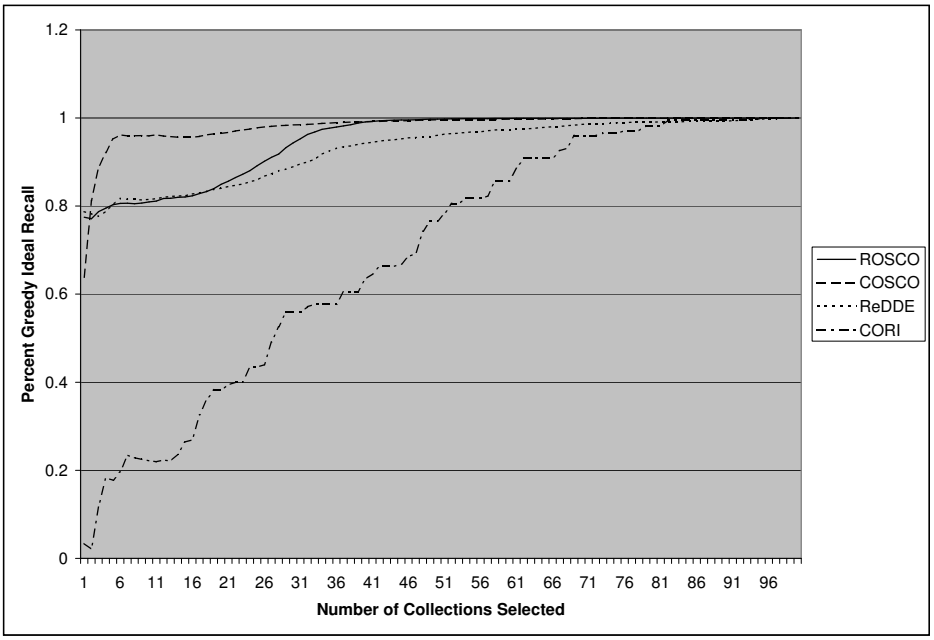

Figure 4.44: Percent of greedy ideal for test bed 4

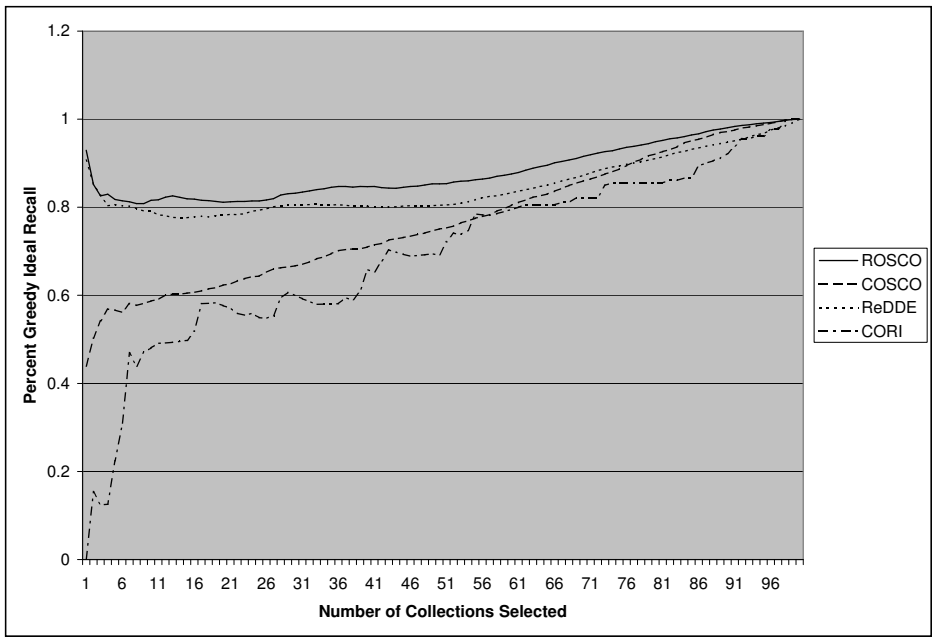

Figure 4.45: Percent of greedy ideal for test bed 5

because relevance is now not random and some collections are much more relevant than others for a given query.

Figure 4.46 again illustrates that ROSCO outperforms all of the other methods. CORI's performance degrades significantly in the presence of overlap.

Test beds 7 and 8 represent those that are most likely encountered in the real world. The collections vary in size and they have a clustered distribution. In test bed 7, ROSCO outperforms the other methods at every step except for a small range where CORI performs the best; but, CORI is very unstable in this test bed. In test bed 8, ROSCO outperforms the other methods until about half of the collections have been selected when COSCO begins to outperform ROSCO; however, since resource selection aims to select a small subset of collections, it is more important to perform well early on. Notice that CORI's performance deteriorates quickly because of the presence of overlap.

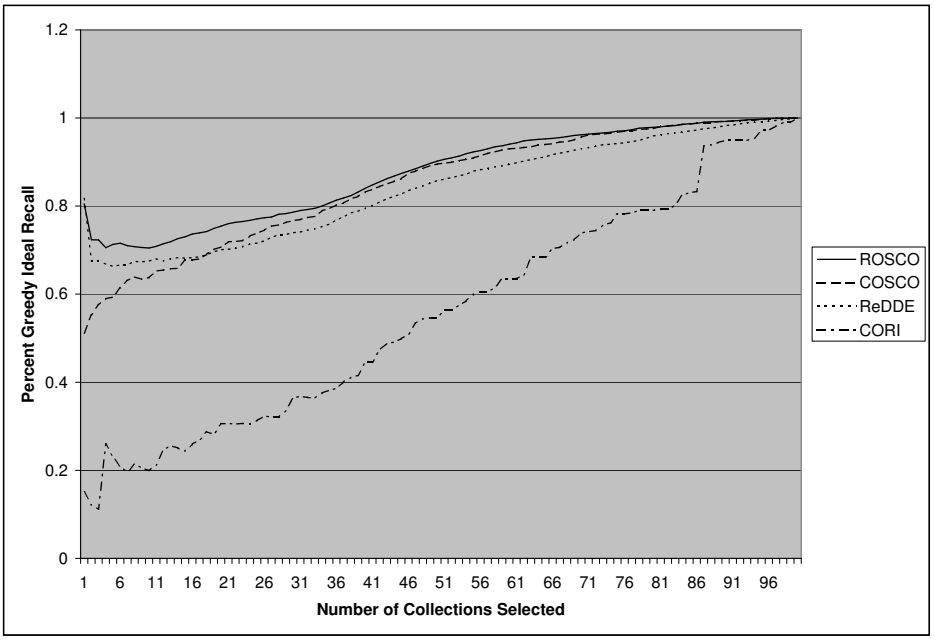

Figure 4.46: Percent of greedy ideal for test bed 6

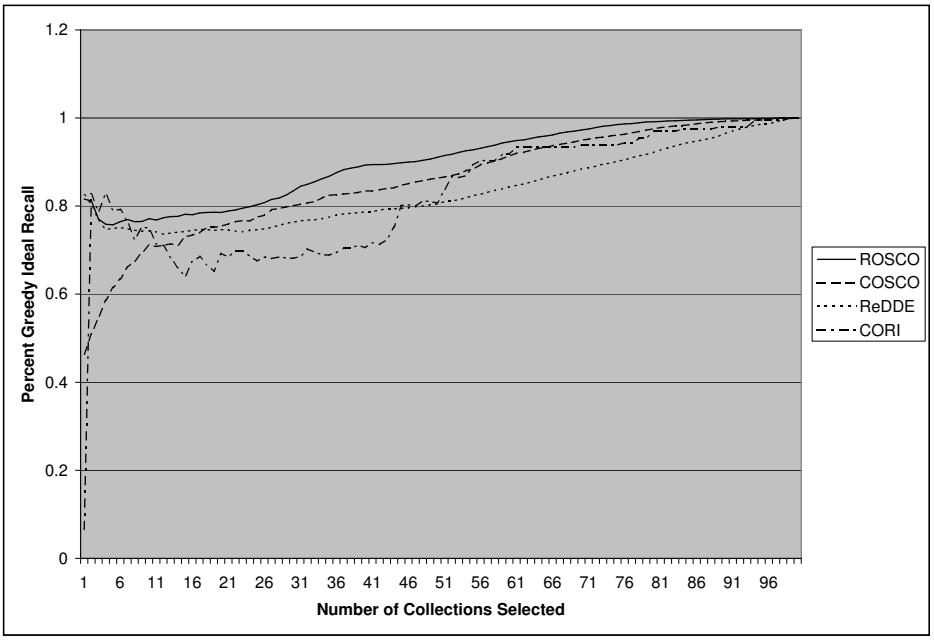

Figure 4.47: Percent of greedy ideal for test bed 7

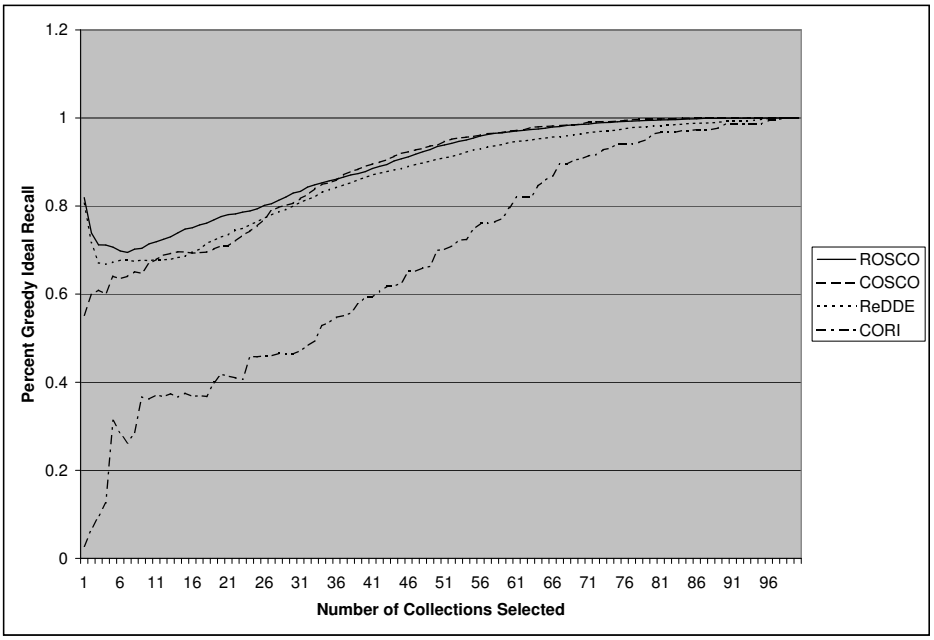

Figure 4.48: Percent of greedy ideal for test bed 8

ROSCO performs the best over all the collections with the exception of test beds 3 and 4; however, these test beds reflect scenarios that are unlikely in the real world. It should also be noted that ReDDE follows ROSCO closely but ROSCO consistently improves on ReDDE by 3%-7%. Furthermore, ROSCO is an improvement on COSCO as well. The presence of overlap does not seem to negatively affect the performance of ROSCO in relation to COSCO. CORI suffers whenever overlap is present and seems to be much less stable than the other three methods.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis presents a new method for resource selection, ROSCO, that combines and extends notions presented in COSCO and ReDDE. The method is effective and has been shown to be an improvement over both of the other methods as well as the currently accepted method, CORI. ROSCO successfully captures the ideas of relevance and overlap which are two of the most fundamental characteristics of collections in resource selection. This work also successfully created a set of eight test beds which represent a variety of conditions. Finally, a detailed evaluation of the proposed method as well as three other major methods is described.

5.2 Future Work

Several extensions to the work presented in this thesis are possible. Two such extensions are presented below: an expansion to the experiments presented in the thesis and capturing non-binary relevance.

5.2.1 Expansion of Experimental Evaluation

An expansion of the experiments performed in this thesis could lead to further discovery. It would be beneficial to study the effectiveness of each method when the number of documents within a collection is much larger and perhaps

the number of collections is greater as well. These improvements would provide a more complete view of the performance of the methods. Specifically, the creation of eight similar test beds from the TREC data would enable comparisons with other experiments to be made more easily and would increase the size of the constituent collections.

5.2.2 Capturing Non-Binary Relevance

In this thesis, relevance is binary. Either a document is relevant enough or it is not. A document is considered relevant if it belongs in the top-k overall documents for a given query. It would be interesting to consider documents of varying degrees of relevance. In this case it might be more important to find the most relevant documents even at the expense of retrieving more less relevant documents. In order to study this a new metric needs to be proposed that captures how well a resource selection method performs in this regard. Once this is done, then methods that approximate the ideal could be implemented and tested.

Bibliography

[1] The ACM Digital Library. *http://www.acm.org/dl*, 2004.

[2] The ACM Guide to Computing Literature. *http://www.acm.org/guide*, 2004.

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

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

[5] BibFinder: A Computer Science Bibliography Mediator. *http://rakaposhi.eas.asu.edu/bibfinder*, 2004.

[6] Callan, J. Distributed Information Retrieval. In W.B. Croft, editor, *Advances in Information Retrieval*, 127-150. Kluwer Academic Publishers, 2000.

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

[8] Callan, J., Lu, Z., and Croft, W. B. Searching Distributed Collections with Inference Networks. In *Proceedings of 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.

[9] Chowdhury, A., Frieder, O., Grossman, D., and McCabe, M. C. Collection statistics for fast duplicate document detection. In *ACM Transactions on Information Systems*, 20(2):171-191, 2002.

[10] CiteSeer - Computer and Information Science Papers. *http://citeseer.org*, 2004.

[11] The Collection of Computer Science Bibliographies. *http://liinwww.ira.uka.de/bibliography*, 2004.

[12] Compendex Database. *http://www.engineeringvillage2.org*, 2004.

[13] DBLP – Computer Science Bibliography. *http://www.informatik.uni-trier.de/~le*, 2004.

[14] French, J. C., Powell, et al. Comparing the Performance of Database Selection Algorithms. In *Proceedings of ACM SIGIR Conference*, 1999.

[15] Gravano, L., Chang, C., Garcia-Molina, H., and Paepcke, A. STARTS: Stanford proposal for internet metadata searching. In *Proceedings of the 20th ACM-SIGMOD Internal Conference on Management of Data*, 1997.

[16] Gravano, L., García-Molina, H., and Tomasic, A. GlOSS: text-source discovery over the Internet. In *ACM Transactions on Database Systems*, 1999.

[17] Haveliwala, T., Gionis, A., Klein, D., and Indyk, P. Evaluating Strategies for Similarity Search on the Web. In *Proceedings of the World Wide Web Conference*, 2002.

[18] Hernandez, T. Improving Text Collection Selection with Coverage and Overlap Statistics. Masters thesis. Arizona State University, 2004.

[19] IEEE Xplore. *http://ieeexplore.ieee.org*, 2004.

[20] Liu, K. L., Yu, C., Meng, W., Santos, A., and Zhang, C. Discovering the representative of a search engine. In *Proceedings of 10th ACM International Conference on Information and Knowledge Managment (CIKM)*, 2001.

[21] Network Bibliography. *http://www.cs.columbia.edu/~hgs/netbib*, 2004.

[22] Nie, Z. and Kambhampati, S. A frequency-based approach for mining coverage statistics in data integration. In *Proceedings of the International Conference on Data Engineering*, 2004.

[23] Powell, L. P. and French, J. C. Comparing the performance of collection selection algorithms. In *ACM Transations on Information Systems*, 21(4):412-456, 2003.

[24] Salton, G., Wong, A., and Yang, C. A vector space model in information retrieval. In *Communications of the ACM*, 1975.

[25] ScienceDirect. *http://www.sciencedirect.com*, 2004.

[26] Shivakumar, N. and García-Molina, H. Finding near-replicas of documents on the web. In *Proceedings of WebDB*, 1999.

[27] Shivakumar, N. andGarcía-Molina, H. SCAM: A copy detection mechanism for digital documents. In *Proceedings of the Conference on the Theory and Practice of Digital Libraries*, 1995.

[28] Si, L. and Callan, J. 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*, 2003.

[29] Si, L. and Callan, J. Using sampled data and regression to merge search engine results. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[30] Voorhees, E. M., Gupta, N. K., and Johnson-Laird, B. The collection fusion problem. In *Text REtrieval Conference, TREC*, 1994.

[31] Yu, C. T., Liu, K., Wu, W., Meng, W., and Rishe, N. Finding the most similar documents across multiple text databases. In *Advances in Digital Libraries*, pages 150-162, 1999.

[32] Yuwono, B. and Lee, D. L. Server ranking for distributed text retrieval systems on the internet. In *Database Systems for Advanced Applications*, pages 41-50, 1997.