Database Selection in Distributed Information Retrieval: A Study of Multi-Collection Information Retrieval

A Dissertation

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Computer Science

by

Allison L. Powell

January 2001

Approvals

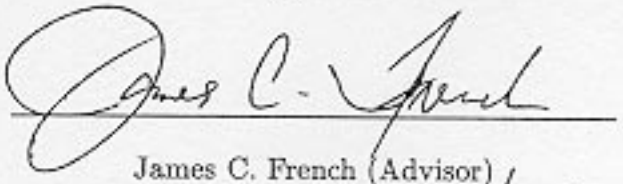
This dissertation is submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

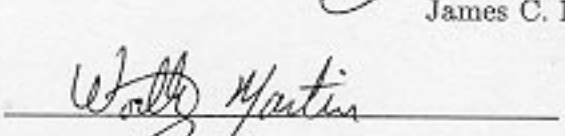
Computer Science

Allison L. Powell

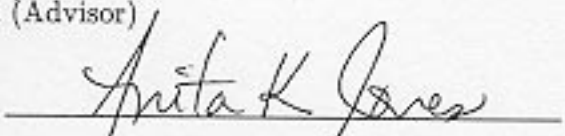
Approved:



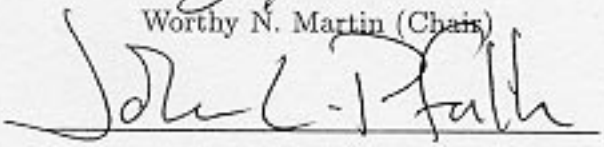
James C. French (Advisor)



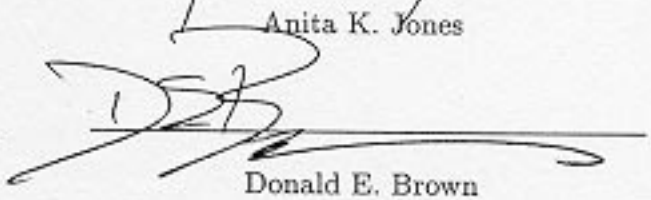
Worthy N. Martin (Chair)



Anita K. Jones



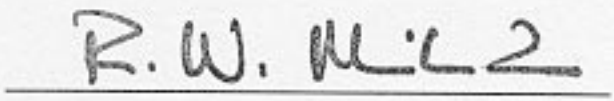
John L. Pfaltz



Donald E. Brown

(Minor Representative)

Accepted by the School of Engineering and Applied Science:



Richard W. Miksad (Dean)

January 2001

Abstract

The proliferation of online information resources increases the importance of effective and efficient information retrieval in a multi-collection environment. Multi-collection searching includes distributed searching as a special case but is more broadly defined here to incorporate searching partitioned content independently from its physical storage. It is cast in three parts: collection selection (also referred to as database selection) – decide where should a query be sent; query processing – execute the query at each selected collection; and results merging – combine the results from individual collections into a single coherent list for the searcher. We focus our attention on collection selection.

We compare a number of different collection selection approaches and examine the effect of collection selection on document retrieval performance. We consider multi-collection retrieval in six different test environments utilizing three document testbeds. Considering collection selection in isolation, we find that effective collection selection can be achieved using limited information about each collection. We then turn our attention from selection alone to data item retrieval in a multi-collection environment, considering retrieval performance in the same six test environments. First we find that good collection selection has the potential to result in better retrieval effectiveness than can be achieved in an equivalent single collection. Second we find that good performance can be achieved when only a few collections are selected and that the performance generally increases as more collections are selected. Finally we find that when collection selection is employed, it may not be necessary to maintain collection wide information (CWI), e.g., global idf. Local information can be used to achieve equivalent performance. This means that multi-collection systems can be

engineered with more autonomy and less cooperation. This work demonstrates that improvements in collection selection can lead to broader improvements in document retrieval performance.

Acknowledgements

First, thanks to my advisor, Jim French. His encouragement, support, and advice are greatly appreciated. My thanks also go to Worthy Martin, John Pfaltz, Anita Jones and Don Brown for serving on my dissertation committee.

Thanks to information retrieval research group members past and present. Special thanks go to Charlie Viles for setting a good example and for later collaborations. Special thanks also go to Travis Emmitt and Kevin Prey for assistance with testbed building and early collection selection experiments and to Eddie O'Neil and James Watson for help in collecting (and judging!) documents for the WWW metasearching experiments.

Thanks to Jamie Callan of Carnegie Mellon University and Margie Connell of the University of Massachusetts for collaborations and for CORI and Inquery expertise.

Thanks to Distributed Information Retrieval seminar class members Gary Monroe, David Mikesell, Tram Phan, Rashmi Srinivasa and Nisanti Mohanraj and Eddie O'Neil for valuable discussions about early versions of the WWW metasearching experiments.

Finally, heartfelt thanks to my family and all of my friends for providing invaluable moral support.

NASA provided financial support for this work through Graduate Student Researchers Program fellowship NGT5-50062.

Contents

List of Figures

List of Tables

List of Symbols

$\mathcal{C}$	a set of collections
$\mathcal{C}_{sel}$	the set of selected collections
C	an arbitrary collection, $C \in \mathcal{C}$
$\mathcal{D}$	the set of data items
d	an arbitrary data item, $d \in \mathcal{D}$
df_{ij}	the document frequency of term t_j in collection C_i
$\mathcal{I}$	the set of indexes associated with $\mathcal{C}$
I_i	the index associated with collection C_i
icf_j	the inverse collection frequency of term t_j
$\mathcal{J}$	the set of relevance judgements
$\mathcal{LM}$	the set of language models associated with $\mathcal{C}$
LM_i	the language model associated with collection C_i
M	the total merit in a set of collections $\mathcal{C}$ for query q
N	the total number of collections
n	the number of collections currently selected
n^*	the number of collections containing data items with non-zero merit
$\mathcal{Q}$	a set of queries
$\mathcal{Q}_s, \mathcal{Q}_l, \mathcal{Q}_{vl}$	the sets of title, long and very long query formulations used in our experiments
q	an arbitrary query, $q \in \mathcal{Q}$
$R_{\mathcal{D}\rightarrow\mathcal{C}}$	the relation specifying the data item to collection mapping
$R_{\mathcal{LM}\rightarrow\mathcal{C}}$	the relation specifying the language model to collection mapping
$\mathcal{R}_n, \widehat{\mathcal{R}}_n, \mathcal{P}_n$	collection selection evaluation measures (defined in Chapter 5)
rel_judge_{ij}	the relevance judgement of document d_j for query q_i
$(\mathcal{C}, \mathcal{Q}, \mathcal{J})$	an experimental test environment made up of collections, queries and relevance judgements

List of Abbreviations

CWI	Collection-Wide Information
RBR	Relevance Based Ranking baseline
SBR	Size Based Ranking baseline
SYM-236	Source-Year-Month testbed
TREC	Text REtrieval Conference
UBC-100	Uniform-Byte-Count testbed
UDC-236	Uniform-Document-Count testbed

1

Introduction

The growth of the internet, federated digital libraries and other information sources in which data items may be spread across a number of providers has increased attention on the problem of retrieving data items found in these environments. For reasons of efficiency, intellectual property or copyright, it may not be feasible or desirable to assemble all data items of interest at a central storage location or even to provide access through a centralized index. Consider for example online journals by different publishers and technical reports by different academic institutions or research labs. As a result, data items may be found in a large number of collections. While this situation may be necessary from the information provider's point of view, it can prove problematic for end users. Unless some intermediary is available, users must first be aware of useful collections, then issue their queries to each useful collection in turn. Fortunately, if each collection provides a means to search for and access data items, it is possible to employ an intermediary that provides a single point of access to a set of collections. The underlying collections may actively cooperate with the intermediary, as in the case of a federated digital library, or may be unaware of the intermediary, as in the case of World Wide Web (WWW) metasearch engines. Within the information retrieval community, the problem of retrieving data items from a set of collections has typically been referred to as *distributed* information retrieval. We will refer to the problem as multi-collection retrieval to emphasize the fact that collections may or may not be physically distributed. While physical distribution is a common scenario, it is

not a requirement.

1.1 Problem and Motivation

The problem of data item retrieval in a multi-collection environment can be broken down into three major sub-problems. Given a set of collections to which a user's query might be sent, the first sub-problem is to either choose the order in which the collections will be searched or alternately to choose a subset of the collections for search. We refer to this sub-problem as the *collection selection* step. This step becomes increasingly important as the number of collections grows or if the collections charge for access. The second sub-problem is to forward the user's query to the selected collections. This step can be challenging in heterogeneous environments where the underlying collections may use different query syntax. The third sub-problem is to take the individual results lists from each of the selected collections and to merge those results into a single coherent list of results to be presented to the user. We refer to this final step as *results merging*.

A great deal of work has been done on these individual sub-problems and additional research has focused on the multi-collection retrieval problem as a whole. Because research efforts have focused on different sub-problems as well as the whole, and due to the wide variety of evaluation measures and test environments, it has been difficult to directly compare different published evaluation results. Also, because evaluation has generally been performed for a sub-problem considered in isolation or for a multi-collection retrieval system as a whole, the impact of improvements for a sub-problem on overall system performance has been difficult to isolate.

Our primary goal for this work has been to enhance the understanding of the overall multi-collection retrieval problem, including the potential that introducing multiple collections when a single one is possible may become advantageous. Additional goals have been to compare competing collection selection techniques and to determine reasons for their success or failure in different environments, to determine what information about collec-

tions is necessary to enable effective collection selection, to investigate the degree to which improvements in collection selection can impact overall data item retrieval performance and to investigate the applicability of experimentally-proven techniques in an operational environment.

The thesis of this work is that collection selection in multi-collection environments can be effective even when statistical information about collections is limited, and that given a small number of well-chosen collections, effective document retrieval can be attained.

1.2 Approach

We have taken a three-pronged approach to studying the multi-collection retrieval problem. First, we have performed an in-depth comparison of competing collection selection approaches. We have analyzed the performance for a common task, using three testbeds and two query sets and have studied reasons for observed differences in performance. We have also studied the components of the approaches to gauge what information about collections is necessary for effective selection. We have studied data item retrieval performance when different collection selection approaches were employed. For the data item retrieval experiments, we used an existing collection selection approach then a best-case approach to select a small number of collections. The resulting data item retrieval was compared to gauge the retrieval performance gains that may be achieved using improved collection selection. Finally, we have applied the top-performing collection selection approach from our comparisons to a heterogeneous WWW metasearch environment to judge its broader applicability.

1.3 Contributions

We make a number of contributions with this work, discussed here in the order of their presentation. The primary two contributions, discussed in more detail below, are our comparison of collection selection approaches and our investigation of the impact of collection

selection on multi-collection retrieval.

We introduce two new testbeds and include a third in our experiments. We characterize features of all three testbeds that may affect performance results. We also introduce notation for describing multi-collection experiments such as those conducted here, and for describing multi-collection experimental environments. We perform a direct comparison of three competing collection selection approaches and analyze reasons for observed performance differences, including the testbed features mentioned above. We show that very limited information about the underlying collections is adequate for effective collection selection. Presently, two major camps of collection selection approaches exist—one that uses minimalist collection information and another that uses detailed information about the documents within the collections. Our findings help to resolve the open question of whether the detailed information is advantageous. We also find that communication among collections may not be necessary given effective collection selection. This finding plus our finding that minimalist collection information is adequate for collection selection bode well for the transfer of experimental multi-collection retrieval techniques to an operational environment. We abstract the approach that performed best in our comparison of collection selection approaches and examine the impact of its constituent components, isolating the differences that are key to its success. We collect a variety of evaluation measures, show relationships among them and discuss expected random performance under the measures. We show that on average, published collection selection approaches outperform selecting collections at random, a basic but reassuring finding. Of particular interest, we find that for all three testbeds, good collection selection of a subset of collections can yield data item retrieval results superior to those when all data items are located in a single collection. We present a new way of considering multi-collection environments that allows these techniques to be layered on top of existing single collection systems. We have deployed the multi-collection retrieval approaches studied here in a WWW metasearch environment. We report preliminary effectiveness results, plus an experience report of issues faced when applying these techniques to a heterogeneous operational environment.

1.4 Outline

Chapter 2 describes the multi-collection information retrieval problem in much more detail and includes a primer on single-collection retrieval concepts that are applicable to multi-collection retrieval. We discuss related work in the context of the broader problem description. Chapter 3 introduces more detailed notation and definitions for concepts introduced in Chapter 2. In Chapter 4 we describe the three different testbeds that we use, plus the query sets employed. In Chapter 5 we collect the evaluation measures used in the experiments reported here, discuss features of those evaluation measures and cover expected performance under three of the measures for randomly generated rankings. Chapter 6 covers early experiments that studied the *gGlOSS* [GGM95] collection selection algorithm in detail. This chapter also explains the choice of the *gGlOSS Ideal(0)* baseline as the representative for the *gGlOSS* approach in later experiments. Chapter 6 also presents the results of a preliminary comparison of the *gGlOSS* and *CORI* [CLC95] collection selection approaches. An important step in performing this comparison in a straightforward manner was the vocabulary resolution work described in Appendix A. Chapters 7 and 8 report the results of our collection selection algorithm comparisons and our study of the components of one class of collection selection approaches. Chapter 9 contains results of a comparison of multi-collection and single-collection data item retrieval and Chapter 10 studies the multi-collection retrieval problem in a WWW environment. Chapter 11 concludes.

2

Background and Related Work

Broadly speaking, information retrieval is concerned with identifying data items that have the potential to satisfy a user's information need. A great deal of research has been focused on the problem of effectively and efficiently retrieving data items from a *single* collection. This is generally referred to as *centralized* information retrieval, i.e. data items are considered to be located at a central source. However, the growth of the internet, federated digital libraries and other information sources in which data items may be spread across a number of sources has increased attention on the problem of retrieving data items from *multiple* collections. This problem has generally been referred to as *distributed* information retrieval. In this dissertation we will use the terms single-collection and multi-collection in place of centralized and distributed, respectively.

The issues faced in single-collection information retrieval still hold in a multi-collection environment. Given multiple collections from which we might wish to retrieve data items, we are obviously still concerned with high-quality retrieval from each of those collections. However, multi-collection retrieval involves additional challenges, for example, selecting which collections to search, issues due to heterogeneity in the collections, and how to present results from multiple collections to a user in a coherent fashion.

The work reported in this dissertation is concerned with the additional challenges that are present for multi-collection information retrieval. However, a quick overview of single-

collection retrieval will be useful for providing context and will introduce some applicable terminology. Therefore, we will start with a brief overview of data item retrieval from a single collection, then move on to a discussion of some of the issues specific to retrieval in a multi-collection environment. The discussion of multi-collection retrieval will be organized into a number of sections. Multi-collection retrieval can be divided into three steps, collection selection, issuing queries to the selected collections, and results merging. We will discuss collection selection and results merging individually, covering related work that focused on those specific sub-problems. We will then consider a few more detailed issues that impact the multi-collection retrieval problem as a whole. We will also discuss the related sub-problem of WWW metasearching.

In this chapter, we cover general background plus specific related work in the area of multi-collection information retrieval. The data items found in a single- or multi-collection environment might be text documents, images, sound recordings or binary files. However, in the examples and experiments presented here, all data items are text documents. As a result, when we are providing informal descriptions and examples we will refer to the data items as *documents*. However, when we present more formal notation for describing collections or experimental setup in Chapter 3, we will use the broader terminology *data items* to emphasize that that our notation is applicable to data items other than text documents.

2.1 Information Retrieval from a Single Collection

Information retrieval centers around a user with an information need. Given a collection of documents, the purpose of an information retrieval system is to identify the documents with the highest potential to satisfy that information need. This task is complicated by the often complex and unstructured nature of the documents and the frequent inability of a user to fully and accurately describe the information need.

Considered from an operational viewpoint, we start with a collection of documents.

An information retrieval system will provide the capability to preprocess, then *index* the documents. This creates an internal representation of each document that is used by the information retrieval system but is not apparent to users. Given a user with an information need, we assume that the user constructs a *query* that is a statement of that information need in a format appropriate for submission to the information retrieval system. Given the query and the internal representation of each document, a boolean information retrieval system returns the set of documents that satisfy the boolean predicate specified by the query. A similarity-based information retrieval system computes a score for each document with respect to the query and returns the documents in order of decreasing similarity. The user examines the returned documents to determine if any are *relevant* (i.e. satisfy the original information need). For information retrieval systems, the user is the final arbiter of relevance. Experimental systems are generally evaluated based their ability to locate relevant documents. If no returned documents are relevant, or if an insufficient number of relevant documents are returned, the user may choose to *reformulate*, (i.e. revise) the query and try again.

One thing that makes information retrieval challenging is that it can be difficult for users to formulate a query that fully and precisely captures the underlying information need. Users often submit queries that contain only a few terms or queries that only cover one facet of the information need. Users can also experience difficulty when the vocabulary used in a query does not match the vocabulary used in documents.

Because our research is focused on issues particular to multi-collection information retrieval, our work is only indirectly dependent upon implementation decisions at individual collections. As we mentioned earlier, we are obviously concerned that each collection provides high-quality results effectively and efficiently. Poor-quality results from the underlying collections could seriously degrade the overall performance of a multi-collection system. Aside from this obvious concern, we are also interested in the document preprocessing and indexing approaches at the underlying collections because collection selection approaches may utilize statistical information from the collections. Any heterogeneity at the underly-

ing collections may have to be accounted for when building collection representations to be used by a collection selection algorithm.

A multi-collection information retrieval approach may or may not have access to detailed information about the information retrieval systems in place at the underlying collections. As we will discuss in Section 2.2, some multi-collection information retrieval approaches are more dependent upon detailed information than others.

In the discussion that follows, we will cover document preprocessing steps, general information retrieval models and the statistical information used by systems based on these models. These information retrieval models will be referred to in Section 2.2. Some of the different multi-collection retrieval approaches that we study were originally based on specific information retrieval systems at the underlying collections. We will refer to those systems when surveying related multi-collection research. We will also cover the philosophy behind the evaluation techniques that are used for data item retrieval evaluation. We will revisit evaluation and evaluation measures in Chapter 5.

2.1.1 Document Preprocessing

Before documents in a collection are indexed they must be in a format that is compatible with the information retrieval system. For example, a WWW search engine may be able to directly handle HTML marked-up documents, while a more traditional text search engine may treat HTML tags, javascript, etc. as normal, to-be-indexed text if it is not removed in a preprocessing step. Once documents are in a compatible format, they are *tokenized* to isolate the individual terms or phrases that make up the document. For simplicity, we will assume that tokens are terms. Many information retrieval systems then employ a *stoplist* to remove extremely common terms such as articles. The stoplist terms are removed because they inflate the overall vocabulary size but are seldom useful in differentiating one document from the next. The final commonly-used preprocessing step is *stemming* in which word endings are removed. This is used to compress the vocabulary used for indexing and to cause word variants to resolve to the same token/term. For example,

"computer", "compute" and "computing" might all stem to "comput". See Salton [Sal81] for an overview of the document indexing process which covers these steps in greater detail.

2.1.2 Information Retrieval Models

Information retrieval systems can be classified in terms of how they represent documents internally, how they represent queries, and in terms of how they perform comparisons of these internal representations of documents and queries for the purpose of creating a set or list of documents to return to a user.

Indexing is the step by which an internal representation of each document is created. The actual representation and the importance of individual terms or other document components are particular to each individual retrieval system, but for text documents the importance of a term is often some function of statistical information about the occurrence of the term in the document and/or the frequency of occurrence of the term in the collection at large. Two commonly-used statistics are *term frequency (tf)* and *inverse document frequency (idf)*. The term frequency (tf_{ij}) is a count of the number of occurrences of term t_j in document d_i. The inverse document frequency (idf_j) is $\log(\frac{N}{df_j})$ where N is the number of documents in the collection and df_j is the number of those documents containing term t_j. Functions of term frequency and inverse document frequency are often used in conjunction with one another and different combinations of functions of these two components have been well-studied [SB88]. The goal of combining these two statistics is to assign higher weights to terms that occur frequently in a document but infrequently in the collection as a whole. Such terms are considered to best represent the content of a document.

As a part of the indexing process, information retrieval systems may also employ some form of *document length normalization*. Longer documents may produce higher values of some statistics, for example term frequency, due solely to the higher overall count of terms. Without normalization, some information retrieval systems tend to assign higher similarity values to longer documents, and thus make longer documents appear more relevant simply because of their length. Document length normalization combats this tendency and has

been shown to improve information retrieval system performance [SB88, SBM96]. An analogous effect for multi-collection retrieval has been observed for some collection selection approaches and is discussed in Chapters 7 and 8.

While we summarize four different classes of information retrieval systems below, we emphasize that we are not researching document indexing approaches. We summarize the approaches here because the details of these approaches have implications for some of the multi-collection retrieval approaches that we consider in Chapter 7. We also refer to different classes of information retrieval systems during our discussion of multi-collection related work in Section 2.2. In a multi-collection environment, we may not have control over the information retrieval system(s) used by the underlying collections. In fact, different information retrieval systems may be used by the different collections. This can have compatibility and performance implications for multi-collection retrieval experiments.

Boolean model. Boolean information retrieval systems maintain a simple index that, for each term in the vocabulary, is essentially a list of all documents containing at least one instance of that term. A query is expressed as a boolean predicate and the set of all documents satisfying that predicate is returned. Two major drawbacks to simple boolean systems have been noted. First, users have difficulty constructing effective queries. Second, the returned documents are not ordered. Some of the multi-collection retrieval systems discussed in Section 2.2 are based on underlying boolean collection indexes, but this approach is not used in any of the experiments reported in this dissertation.

Vector space model. In the vector space model, both documents and queries are represented as v-dimensional vectors where v is the number of terms in the collection vocabulary. For example, document d_i is represented as $(w_{i1}, w_{i2}, \ldots, w_{iv})$ where w_{ij} is the weight of term t_j in document d_i. The similarity between a document and a query (or two documents or two queries) is computed as the cosine of the angle between the two vectors. The vector space model is surveyed in Salton [Sal91]

and covered in detail in Salton & McGill [SM83]. The SMART [Sal71] information retrieval system is a vector-space system. Version 11.0 [Buc92] of the SMART system was used as the underlying information retrieval engine for the experiments reported in Chapter 6.

Probabilistic model. On a conceptual level, information retrieval systems based on the probabilistic model seek to maximize the probability that relevant documents are retrieved and minimize the probability that non-relevant documents are retrieved. Relevance properties of documents in which query terms occur are considered and used to weight query terms [RJ76]. Operationally, these relevance properties cannot be known exactly and must be estimated based on training collections, collection statistics, or relevance assessments of documents retrieved in response to previous queries.

Inference net model. The inference network model [TC90, TC91, TC92] is a kind of probabilistic model that generalizes both the boolean and probabilistic models described above. An inference network is a directed acyclic graph. A document network is used to represent a set of documents while a query network is used to represent each query. Both documents and queries are nodes in their respective networks. Concepts (often terms found in documents or in queries) are also nodes in the graph and belief values are assigned to documents based on the presence or absence of concepts. Given a query, concepts in the query network and document network are matched to compute the probability (belief) that a document is relevant to the query. The Inquery [CCH92] information retrieval system that is used as the underlying retrieval engine for the experiments reported in Chapter 9 is based on the inference net model. Inquery has been shown to perform well at TREC conferences [VH98].

2.1.3 Evaluating Retrieval Results

The overall goal for an information retrieval system is to retrieve all of the relevant documents in a collection while at the same time retrieving no non-relevant documents. In practice, this doesn't happen very often. Realistically, assuming an output of a ranked list of documents, we prefer an information retrieval system that ranks relevant documents highly.

In an operational environment, the nature of the user's information need can influence the nature of the evaluation. There is a difference between fact-based searching and in-depth searching. For example, if a user wants to know the capital of Virginia, a single document that answers the question is sufficient. Additional relevant documents may serve to lend credibility to the first but do not provide new information. Contrast this with a search for related work for a dissertation, or a lawyer's search for related case law. In the latter cases, each new relevant document is valuable. In our experiments, we assume the latter scenario—we assume that a user wants to retrieve every relevant document.

As we mentioned before, the user is the final arbiter of whether a data item is useful. When evaluating information retrieval systems based on relevant documents retrieved, researchers generally use two measures, *recall* and *precision*.

$$recall = \frac{\text{number of relevant documents retrieved}}{\text{total number of relevant documents in the collection}}$$

$$precision = \frac{\text{number of relevant documents retrieved}}{\text{number of documents retrieved}}$$

Recall and precision are usually presented in one of two ways. One common approach is to report recall-precision curves. For this approach, precision values are computed at fixed recall points. For example, precision is computed when 10%, 20%, etc. of the relevant documents have been retrieved. These results are generally presented as graphs where precision values are computed for $R = 0.1, 0.2, \ldots, 1.0$. Plots closer to $R = 1, P = 1$ denote better performance. Alternately, researchers may report recall and/or precision at fixed

values, for example when 10, 20, etc. documents have been retrieved. Reporting precision at fixed values is particularly useful for WWW applications and other environments for which recall values may not be known. This approach has also been used for reporting TREC experimental results, based on the observation that users typically examine only the most highly-ranked results. Recall values can be difficult to obtain because recall requires that data items from the collection at large be judged, not just the data items that are retrieved. This is problematic in environments with hundreds of thousands of data items.

2.1.4 Effectiveness Experiments Using Test Collections

Information retrieval systems are often tested and evaluated using *test collections*. Test collections for single-collection information retrieval experimentation are typically made up of a collection of documents, a set of queries to be used in testing, plus a fixed set of relevance judgements, generally compiled by an independent party. The relevance judgements provided allow the computation of relevance-based evaluation measures like recall and precision.

Test collections have a number of advantages. They represent an efficient use of human effort. While it is time-intensive and tedious to judge large numbers of documents for a large number of queries, once the judgements have been made for a test collection, many researchers can use this information to evaluate their systems. This allows a large number of experiments using different parameter settings or tuning changes to be tested without assembling a set of users for each experiment. Different sets of users might disagree about the relevance of a document; however, the relevance judgements of a test collections are fixed. While differences in judgements between judges have been shown to have little impact on comparisons of information retrieval systems [Voo00], it is useful to have the judgements remain stable when testing modifications to a single system. Test collections facilitate system comparisons by allowing different systems to be evaluated independently of one another using the same environment. The use of standard test collections also allows

researchers to replicate the experiments of others. Test collections also have drawbacks. For example, the types of documents or types of queries included may not be representative of the types of documents or queries an operational system may encounter. However, at the present time, test collections represent a convenient way to test and compare information retrieval systems.

For effectiveness experiments, the possible sources of test data are relatively limited. Older information retrieval test collections[1] were developed in the 1960s through the 1980s, and due to manpower constraints were necessarily very small, on the order of a few thousand data items and less than 300 queries. The small size of these traditional test collections makes them ill-suited for experiments in which the data must be further sub-divided. The introduction of the TREC/TIPSTER data represented a substantial leap in the size of available test collections; by the fourth TREC [Har95] conference, there were more than one million data items and 250 queries.

2.2 Information Retrieval from Multiple Collections

Up to this point, we have discussed issues that arise concerning information retrieval when all documents can be found in a single collection. Now we move to a discussion of retrieval in a multi-collection environment. The issues that we have been considering still apply. We still want high quality overall document retrieval results and the end user is still the judge of relevant documents. However, we now have additional considerations that are particular to multi-collection environments.

Multi-collection information retrieval consists of three major steps, illustrated in a very simplified fashion in Figure 2.1. First, given a set of collections that may be searched, the collection selection step chooses the collections to which a query q will be sent. In our example, collections C_1 and C_3 are selected. Next, the query is processed at the selected collections, producing a set of individual result-lists. Finally, those result-lists are merged into a single list of documents to be presented to a user.

[1] Available from the Glasgow IDOM server, `http://www.dcs.gla.ac.uk/idom/ir_resources/`

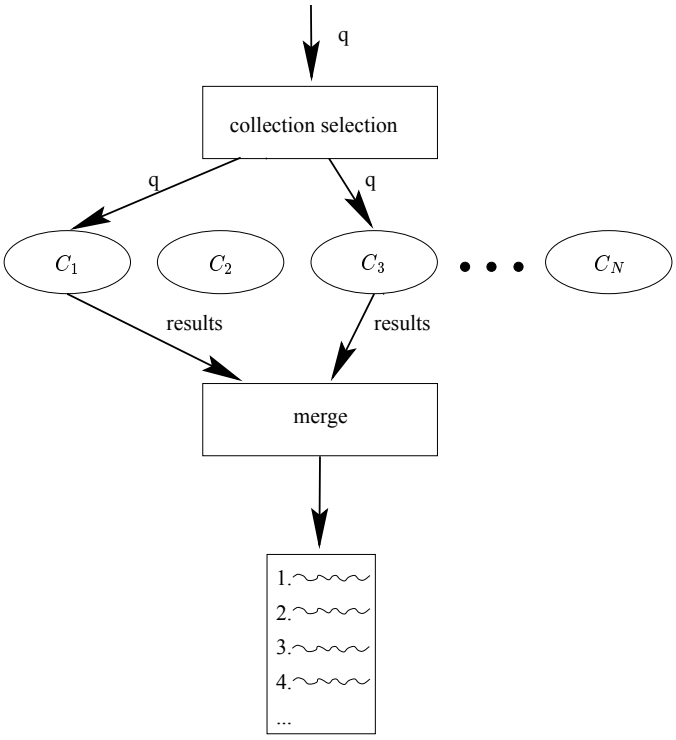


Figure 2.1: An illustration of multi-collection retrieval. The collection selection mechanism routes query q to collections C_1 and C_3. Query q is executed at those collections, then the two results lists are merged into a single, coherent list for presentation to the user.

We will cover each of the three steps of multi-collection retrieval and issues related to those steps in more detail. Within that context, we will discuss related work within the field of multi-collection retrieval. Some research has been focused on one of the individual steps; other work has addressed more than one step and will be mentioned more than once.

2.2.1 Collection Selection

Given some query q and a set of collections to which that query might be sent, the collection selection step may be viewed in two ways. Under one interpretation, the collection selection mechanism specifies the order in which the collections are searched. An alternate interpretation is that the collection selection mechanism chooses a subset of the collections to search. We use the former interpretation and assume that a collection selection mechanism seeking a subset of n collections would simply use the first n collections in the ranking.

To state the problem more formally, we have a set of N collections $\mathcal{C} = \{C_1, C_2, ..., C_N\}$ that we wish to search to satisfy some query q. We assume that each collection $C \in \mathcal{C}$ has some merit, denoted $merit(q, C)$, with respect to the query q. Merit could be defined as the number of relevant documents in a collection, the proportion of relevant documents in a collection, the number of documents in a collection, the number of documents that have a given similarity to the query or any other assignment of values. We would like to search the collections in order of decreasing merit to the query.

To isolate the selection problem, consider for example Figure 2.2. In this example, we have five collections $\mathcal{C} = \{C_1, C_2, C_3, C_4, C_5\}$ with the illustrated merits with respect to some query q. Because we wish to search collections in order of decreasing merit, for this example, we would like to visit the collections in the following order: $\langle C_3, C_1, C_4, C_2, C_5 \rangle$.

This example brings up on interesting point—in fact we only want to search the shaded collections (i.e. only the collections that have non-zero merit). Because collections C_2 and C_5 have no merit with respect to query q, we should not direct the query to them. For clarity of exposition, we have cast selection as an ordering task. The desired behavior of a selection algorithm is to place zero-merit collections at the bottom of the ranking.

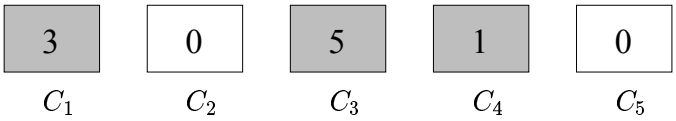


Figure 2.2: A simple collection selection example. Collections C_1–C_5 have the illustrated merits; the goal is to search collections in order of decreasing merit.

Collection selection algorithms do not know the intrinsic merit of a collection with respect to a query. Rather, they provide a means with which to *estimate* that merit. These estimates are used to produce collection rankings. Because the actual collection merits are not known, these rankings based on estimated merit may not be the same as the desired ranking based on actual merit. One approach to evaluating a collection selection technique determines the degree to which the selection technique is able to produce collection orderings that approximate the desired rankings.

A number of different approaches for database or collection selection have been proposed and individually evaluated. Direct comparisons of these collection selection approaches is complicated by the variety of experimental environments and evaluation measures that have been used by different research groups. An examination of Table 2.1 shows the variety of test environments employed by researchers and gives some insight into the difficulty of comparing findings from different research efforts. In addition, the methodology for evaluating collection selection is not yet as standardized as the methodology for evaluating document retrieval.

Comparisons are further complicated by differences in the overall goals of the different approaches. The approaches can be divided into three major classes based on their overall approach or goal. One group of approaches attempts to characterize the document-query similarities of the documents that would be returned if a query were sent to a collection C. These approaches typically have the stated goal of locating collections with a large number of similar documents or with highly-similar documents. A second group of approaches has the stated goal of identifying collections that have a large number of relevant documents

Group	Sources	Num. Colls	Queries
Gravano & García-Molina [GGM95]	news groups	53	6,800 user
Meng *et al.*[MLY+98]	news groups	3	6,597 user
Meng *et al.* [MLY+99] Liu *et al.* [LYM+99]	news groups	3	6,234 user
Yu *et al.* [YLW+99]	news groups	15	1,000 user
Voorhees [Voo95]	TREC (source, year)	10	202-250
Voorhees *et al.*[VGJL95]	TREC (by source)	5	1-200
Voorhees [Voo96]	TREC (source, month)	98	251-300
Voorhees & Tong [VT97]	TREC (by source)	5	51-200
Moffat & Zobel [MZ95]	TREC (by source, disk)	9	51-150
Walczuch *et al.*[WFPS94]	TREC (by source)	5	151-200
Viles & French [VF95a]	TREC-CatB (random)	20	201-250
Callan *et al.*[CLC95]	TREC (by source, disk)	17	51-150
Zobel [Zob97]	TREC (disk 2) TREC (disk 3)	43 91	51-150 202-250
Yuwono & Lee [YL97]	CACM, CISI, CRAN, MED	431	
Xu & Callan [XC98]	3 × TREC (20MB colls)	100, 107, 50	99 TREC

Table 2.1: Summary attributes of multi-collection test environments that have been used in a sampling of previous work.

with respect to the query. The ways in which these approaches attempt to reach that goal vary. Finally, a third group of approaches incorporates additional considerations, for example the cost to search a collection or the expected response time of a collection.

We will discuss a number of different collection selection approaches individually below, grouped by the overall goal. Three of the approaches, *CORI* [CLC95], *CVV* [YL97] and *gGlOSS*[2] [GGM95, GGMT99] were evaluated in a common environment by French *et al.* [FPV+98, FPC+99b] and Callan *et al.* [CPFC00], who found that there was significant room for improvement in all approaches, especially when very few collections were selected. Expanded versions of those experiments can be found in Chapters 6–8. When introducing those experiments, we will present a much more detailed discussion and analysis of the

[2] *gGlOSS* was later renamed *vGlOSS* in Gravano *et al.* [GGMT99], but we will continue to refer to it as *gGlOSS* for consistency with our previously published work.

CORI [CLC95], *CVV* [YL97] and *gGlOSS* [GGM95, GGMT99] algorithms; summaries are provided here to place those approaches in the context of other related work.

2.2.1.1 Collections with Matching or Highly Similar Documents

The philosophy behind approaches that attempt to characterize the query-document similarities of documents within a collection was stated by Gravano and García-Molina [GGM95] who argued that "the best we can hope for any tool like *gGlOSS* is that it predicts the answers that the databases will give when presented with a query." A number of approaches have a similar goal. We summarize some of those approaches here, then briefly discuss environments in which they are applicable and environments for which they are less well-suited.

Gravano *et al.* [GGMT94b] introduced *GlOSS*, the Glossary of Servers Server. *GlOSS* operates in an environment of Boolean information retrieval systems. *GlOSS* utilizes document frequency information for each term and assumes that terms are independently distributed. Given a conjunctive boolean query, *GlOSS* computes the probability that all query terms occur in a given document in a collection to estimate the expected result size of matching documents. Using six collections and 6,897 queries, *GlOSS* was evaluated based on the percentage of queries for which the best collection(s) were selected. The degree to which *GlOSS* ranked the remaining collections correctly was not evaluated.

GlOSS was later generalized to *gGlOSS* [GGM95] to handle the vector space information retrieval model. In the implementation and evaluation of *gGlOSS*, Gravano and García-Molina [GGM95] assumed that all of the collections in $\mathcal{C}$ employ the same algorithms to compute term weights and similarities. Given a similarity function $sim(q, d)$ that computes the similarity between a query q and document d in a collection, Gravano and García-Molina defined a notion of *goodness* for each collection. For similarity threshold l, *goodness* is defined as the sum of all document similarities in the collection where $sim(q, d) > l$. The desired behavior of *gGlOSS* is to rank collections in decreasing order of goodness. Having established the desired behavior, Gravano and García-Molina then defined two *estimators*

that estimate goodness using two assumptions of query term co-occurrence. The *Max(l)* estimator assumes the highest possible level of co-occurrence of query terms in documents while the *Sum(l)* estimator assumes that two terms appearing in the query do not appear together in a collection document. *gGlOSS* needs two vectors of information from each collection in order to make its estimates: the document frequency df_j for each term t_j and the sum of the term weights w_{ij} of each term over all documents d_i in the collection. For both estimators, it is assumed that the weight of a term is distributed uniformly over all documents that contain that term. *gGlOSS* uses the assumptions underlying *Max(l)* (or *Sum(l)*) to estimate the number of documents in a collection C having similarity to a query greater than a threshold l. This forms the basis for the *gGlOSS* estimate of the goodness of C. Gravano and García-Molina [GGM95] used a test environment of 53 collections and 6,800 queries and the $\mathcal{R}_n$ and $\mathcal{P}_n$ evaluation measures (which we discuss in Chapter 5) to evaluate the degree to which the *Max(l)* and *Sum(l)* estimators could rank collections in decreasing order of goodness. They found that both estimators perform well with respect to that evaluation criterion.

Yuwono and Lee [YL97] described the D-WISE multi-collection retrieval system, which considered collection selection, query forwarding and results merging. They referred to the collection selection portion of their work as the *Cue Validity Variance (CVV)* ranking method. *CVV* refers both to the ranking method and to a component in their calculation of collection score. *CVV* is discussed in much more detail in Chapter 7. The *CVV* ranking method employs a combination of document frequency (*df*) information and cue validity variance. Cue validity variance attempts to characterize the distribution of the density of *df* values, i.e., the variability of the fraction of documents in a collection that contain a given term. Document frequency information is used to approximate how important a term is within a collection; the goal of the *CVV* component is to estimate whether a term is useful for differentiating one collection from another. The *CVV* ranking method uses only information from (or derivable from) collection document frequency statistics. The goal of the *CVV* ranking method is to identify collections with a high concentration of query

terms. Despite the availability of relevance judgements for their test environment, Yuwono and Lee used an evaluation approach similar to that employed by Gravano and García-Molina, considering the degree to which CVV predicted the sum of the similarities of the top-ranked retrieved documents. Yuwono and Lee compared their approach to others but unfortunately used only four collections and a small number of documents per collection.

Meng *et al.* [MLY+98] proposed a collection selection approach with goals that were similar to those of Gravano and García-Molina [GGM95]. Given a multi-collection environment, their goal was to estimate the number of documents in the collection that would have a similarity to query q greater than some threshold *if a global similarity function had been employed*. While they use the information differently, Meng *et al.* require the same statistical information about each collection as Gravano and García-Molina [GGM95]—both document frequency information and average term weight information is required. The document frequency information divided by the number of documents in the collection is used as the probability that a document in the collection contains the term. In the most basic formulation of their collection selection algorithm, Meng *et al.* assumed that terms are independently distributed and that the average term weight is evenly distributed across all documents that contain the term. They used collection statistics for each query term to construct a polynomial generating function associated with each collection. The estimated merit (number of documents with similarity greater than a threshold) for the collection was extracted from the polynomial; the exponents represented threshold values while the coefficients represented the probability that a document exceeded the threshold. In essence, the polynomial generating function is a means to compute the probabilities of all combinations of the presence or absence of query terms in documents in a collection, plus the expected similarity of a document with the given combination of query terms present. Reported variations of the collection selection approach relaxed the independence assumption, then relaxed the assumption that the average term weight information is uniformly distributed. The approach was tested using a three-collection reorganization of the data used by Gravano and García-Molina [GGM95]. The approach was shown to outperform the *gGlOSS*

Max(l) and *Sum(l)* estimators for the task of identifying collections containing documents with similarity to a query that is greater than a given threshold. The evaluation focused on the number of queries correctly and incorrectly identifying a collection as useful. Whether collections could be ranked accurately was not reported.

Liu *et al.* [LYM+99] and Meng *et al.* [MLY+99] later expanded the work of Meng *et al.* [MLY+98]. The usefulness of a collection, which was previously defined as the number of documents with similarity greater than some threshold, was redefined to also include the average similarity of those documents. The authors introduced two modified collection selection approaches, the *subrange-based method* [LYM+99, MLY+99] which relaxes a previous assumption that the average term weight information is evenly distributed across documents in a collection that contain the term, and the *combining-terms* [LYM+99] method that incorporates some term dependency information. Both the subrange-based method and the combining terms method modify the collection selection approach of Meng *et al.* [MLY+98] and expand it to utilize maximum document similarity information. The authors reported collection selection results that outperformed those of Meng *et al.* [MLY+98] for the same experimental environment. No modifications to the merging approach discussed in Meng *et al.* [MLY+98] and summarized in Section 2.2.2 were reported.

Yu *et al.* [YLW+99] revisited the subrange-based and combining-terms collection selection approaches. They then provided a document retrieval plan that is guaranteed to retrieve all of the top n globally most similar documents if the collections containing those documents are ranked ahead of collections not containing those documents. Further, the authors show that their subrange-based and combined term methods fulfill this requirement for single term queries (due to the availability of the maximum term weight information). Finally, for a test environment of 9,646 documents arranged in 15 collections and 1000 queries, they compared their combined term method with the *gGlOSS Max(l)* method. The task was to retrieve the n most similar documents and to identify the collections containing those documents. The authors reported that the combined-term method outperformed *gGlOSS Max(l)* for those tasks.

Yu *et al.* [YML+99] also considered using a hierarchy of collection representations when the number of collections available is very large. As in their previous work [MLY+98, MLY+99, YLW+99], they focused on retrieving the globally most similar documents. They demonstrated that the performance of their hierarchical approach is on par with the performance of their related approach that uses a flat representation of collections.

The work of Baumgarten falls somewhat between the class of experiments described here and the class described in the next section. Baumgarten proposed a probabilistic model for multi-collection information retrieval [Bau97], assuming that the underlying collections make use of probabilistic information retrieval systems. He assumed that collections can be hierarchically partitioned. Collections are selected based on estimating the distribution of retrieval status values (RSVs) by which documents are ranked for retrieval within the collections. The collection scores are later used for scaling individual RSVs during a merging step. While document relevance is later used for evaluation [Bau99], collection selection is not based upon an estimation of relevant document distribution. The goal of the overall approach is to maintain the overall top-ranked l documents that would be retrieved in a search of a single collection containing all documents or by selecting all collections, while at the same time restricting search to the collections that actually contribute documents to the set of l documents. In the reported results, the performance of the multi-collection system and single-collection system was very similar.

The majority of our experiments evaluate the degree to which collection selection approaches can locate collections with *relevant* documents instead of highly similar documents. However, we do study the *gGlOSS* approach in detail and use it as a representative for approaches that utilize document term weight information and that attempt to locate collections with highly similar documents.

Traditionally, information retrieval performance in a single-collection environment containing the union of the documents in the multi-collection environment has been viewed as the goal/upper bound for multi-collection retrieval performance[3]. While operational

[3]We will refer to the single collection containing the union of the documents in a multi-collection envi-

multi-collection environments have yet to exceed the performance seen in equivalent single collection environments, we will show later that multi-collection retrieval performance has the potential to outperform single-collection retrieval. Approaches, such as those described above, that are designed to replicate single-collection performance have two potential weaknesses. First, these approaches may search a large subset of the collections if the globally most similar documents are widely spread across the collections. Second, depending on the degree to which they are capable of replicating single-collection performance, these approaches may also not be able to take advantage of the potential for higher multi-collection performance.

2.2.1.2 Collections with Relevant Documents

The next group of collection selection approaches were proposed with the goal of identifying collections with relevant documents. These approaches typically (but do not always) require less statistical information than collection selection approaches with the goal of identifying collections with highly similar documents.

CORI is the collection selection mechanism associated with the Inquery [CCH92] information retrieval system and was introduced by Callan *et al.* [CLC95]. In general, *CORI* treats collections as *virtual documents* using document frequency (df) and inverse collection frequency (icf) information. Collection selection can be considered as a sort of "document retrieval" over the set of virtual documents. The authors presented two general formulations of *CORI* [CLC95]. In our experiments we use the formulation that was shown to have slightly higher performance scores. We discuss *CORI* in much more detail in Chapter 7. Callan *et al.* provided a basic evaluation of *CORI* collection selection performance using a test environment containing 17 collections.

The work of Voorhees *et al.* [VGJL94, VGJL95, Voo95, VT97] is more closely associated with results-merging and will be discussed further there. However, this work also has interesting collection selection aspects. Most selection approaches do not specify the

ronment as an *equivalent single-collection environment.*

number of documents to be retrieved from a selected collection. Instead, the proportion of documents from a given collection present in the merged results list is an artifact of the merge strategy. In contrast Voorhees *et al.* defined two approaches for determining the number of documents to be retrieved from each collection. Because that number may be zero, these approaches serve as a collection selection step. The two approaches, Query Clustering (QC) and Modeling Relevant Document Distributions (MRDD), do not use statistical representations of the collections but instead rely upon information learned from training queries. While their results were promising, the necessity of training data would be a drawback in an operational system. Training queries would need to be representative of queries to be seen by the system and acquiring the training data could be expensive.

Hawking and Thistlewaite [HT99] compared an approach that they term Lightweight Probes (LWP) to the QC and MRDD approaches of Voorhees *et al.* [VGJL94, VGJL95, Voo95, VT97] as well as to two other approaches that rely upon observations of collection performance on previous queries. In contrast, the LWP approach is based upon statistical information provided by the collections in response to a set of probe queries. We will discuss the LWP approach further in a later section in which we consider collection representations used for collection selection.

Zobel [Zob97] termed the approach of using term statistics about collection vocabulary *lexicon inspection* and proposed a number of selection approaches using term statistics. He evaluated the capability of these approaches to locate both relevant documents and high-similarity documents. Overall, he found that an inner product using the query term frequency, the document frequency within a collection and the inverse document frequency within the testbed performed well.

Xu and Callan [XC98] focused on the nature of queries used for collection selection. Their premise was that queries intended for document retrieval (often containing only a few terms) are not appropriate for collection selection and that poor collection selection performance hinders multi-collection document retrieval performance. They investigated the inclusion of phrase information and the use of query expansion for collection selection.

They employed the same experimental methodology as Callan *et al.* [CLC95] but used three different testbeds and two sets of queries. On average, both phrases and query expansion produced results that outperformed the multi-collection base case and at times approached the performance of the single-collection base case. Because Xu and Callan modified the collection selection queries while employing consistent document retrieval queries, they were able to conclude that the improvements in document retrieval were due to their modified selection approaches. Xu and Callan only reported merged document retrieval results, special-purpose collection selection performance measures were not used.

2.2.1.3 Other Considerations

Most collection selection approaches are based upon statistical information about the collections and are concerned primarily with locating relevant or highly similar documents. However, there are approaches that incorporate additional information or set different goals.

An early experiment that considered the use of subdivisions of collections was based on efficiency considerations. Moffat and Zobel [MZ95] considered compression techniques plus a collection selection approach that subdivided collections into blocks of B documents. A selection step was employed to select blocks to which queries should be sent. Moffat and Zobel varied the number of documents per block and studied the impact of block size on the number of documents that must be considered in order to attain document retrieval performance approaching that of an equivalent single collection. For the initial task of retrieving 1,000 documents, they found that it was necessary to consider a large number of documents. They found the approach to be more promising when retrieving a smaller number of documents.

Given a desired number of documents (or a desired number of relevant documents) to retrieve, Fuhr [Fuh99] presented a model for minimizing the cost of retrieving those documents. Fuhr's model takes into account much more information than most of the other approaches outlined here. The model assumes that information is available about the expected number of relevant documents per collection, the cost of executing queries

and retrieving documents from a collection, plus the cost to a user of viewing relevant and non-relevant documents. All of this information is taken into account when considering the overall cost of a query.

In the context of their experiments with Lightweight probes, Hawking and Thistlewaite [HT99] also discussed the cost of accessing collections and considered the cost/benefit trade-offs of selecting additional collections.

Dolin *et al.* [DAAD97, DAAP98, DAA99] described and evaluated the Pharos system in a series of papers. Pharos differs from many other systems because it was explicitly designed to incorporate both textual and non-textual information. The Pharos architecture [DAAD97] requires hierarchical classification of collections (by subject area, date, geographic area, etc.) and utilizes two levels of collection selection intermediaries. High-level intermediaries maintain limited information about all collections while mid-level intermediaries maintain more detailed information about subsets of the classification scheme. Evaluation was based upon the degree to which upper-level intermediaries can predict the contents of mid-level intermediaries. Each collection is responsible for providing the required classification-based metadata. While Dolin *et al.* [DAAP98, DAA99] consider the impact of automatically classified documents, simplifying the task of acquiring the required metadata, Pharos still requires a higher level of cooperation from the participating collections than more minimalist term-statistics based collection selection approaches. Because the evaluation of Pharos followed a different approach than most of the evaluations of statistically-based approaches we have considered, it is difficult to determine if the additional classification information provides an advantage for environments with textual collections.

Craswell *et al.* [CBH00] argued that the retrieval performance at a collection should be incorporated into the collection selection step. They augmented the *CORI* collection selection approach with expected collection retrieval performance, but unfortunately found that the results were not significantly different from standard *CORI* results. Augmenting *CORI* with information about *actual* document retrieval performance (not available in an

operational setting) did improve performance.

Xu *et al.* [XCLN98] focused on the problem of collection selection for multi-attribute bibliographic databases. Like *GlOSS* [GGMT94b] but unlike many of the other approaches surveyed here, they attempted to estimate the overall result *size*, not the number of highly similar or relevant documents. They found that a modification of the *CVV* [YL97] approach to incorporate multi-field data performed better than a proposed approach based solely on training queries. Later related work by the same group of researchers described the design and implementation of ZBroker [LXLN99], a query-routing broker for this problem area. However, in that work, a modified version of *GlOSS* [GGMT94b] was used for collection selection.

For the NetSerf system, Chakravarthy and Haase [CH95] focused on collections organized by subject content. They used hand-crafted representations of collections, then used WordNet [Mil95] to structure, expand and disambiguate queries.

2.2.2 Results Merging

If more than one collection is selected during the collection selection step then the multiple results lists generated by those selected collections will need to be handled in some way. The simplest approach is to concatenate the results lists, delimiting them with some notation of the collection from which they came. While straightforward, this approach does not present a unified, seamless view of the multi-collection environment to the users. Results merging is an attempt to create a single results list from the individual collection's results. The goal is to rank the overall most useful documents highly, to remove any duplicates, and in general to present a coherent view of the results to the user. Results merging is made challenging by heterogeneous information retrieval systems used by the underlying collections. Even if the same information retrieval system is used at each collection, subtle differences in indexing parameters or collection statistics mean that creating a merged list of documents based solely on the similarity scores computed at the collections is rarely the best approach.

There has been attention on results merging or collection fusion from the point of view of query combination and data fusion as well as multi-collection retrieval. In multi-collection retrieval, collection selection need not have taken place; a query may have been broadcast to all collections.

Before we cover some of the merging approaches that have been suggested in the literature, we mention two simple approaches and two more heavyweight approaches to which suggested results merging approaches are often compared. The simple approaches are easiest to explain with an example. Consider a multi-collection environment with two collections, A and B. Assume that these collections return the following results lists, represented here as pairs of document IDs and scores:

$$results_A = \langle (a_1, 0.6), (a_2, 0.4), (a_3, 0.3), (a_4, 0.1) \rangle \quad \text{and}$$

$$results_B = \langle (b_1, 0.8), (b_2, 0.6), (b_3, 0.5) \rangle$$

Interleaving Results. One very simple approach is to *interleave* the documents from the two sources in a round-robin fashion. This approach is applicable to a variety of situations because the document scores provided by the collections need not be comparable. In fact, the document scores are not necessary. For our example, this approach would produce the merged list

$$results_{merge} = \langle (a_1, 0.6), (b_1, 0.8), (a_2, 0.4), (b_2, 0.6), (a_3, 0.3), (b_3, 0.5), (a_4, 0.1) \rangle.$$

This approach assumes that the documents from all collections are equally useful and collections that return more documents have more influence. However, given that document scores are sometimes not available and often not comparable, this can be a reasonable simple approach.

Raw Score Merge. A raw score merge assumes (either correctly or incorrectly) that the document scores produced by the collections are directly comparable. Results are merged on the basis of these scores alone. In our example, a raw score merge would produce the merged list

$$results_{merge} = \langle (b_1, 0.8), (a_1, 0.6), (b_2, 0.6), (b_3, 0.5), (a_2, 0.4), (a_3, 0.3), (a_4, 0.1) \rangle.$$

If the document scores are in fact comparable, a raw score merge can work well. However, if the scores are not comparable the results can be unpredictable. For collections A and B in our example, the scores appear to be on the scale of [0..1]. If that is the case, our merged results may be reasonable. If, however, the results from collection B are on the scale of [0..10], the returned results are likely to be very poor matches and the quality of our merged results list is likely to be adversely affected by ranking documents from collection B highly.

Document Re-weighting. If document reweighting is used to merge multiple results lists, all of the documents retrieved from any collection are considered to constitute a new collection. The document scores assigned by the original collections are discarded. The retrieved documents are then re-indexed and the original query is compared to them. The new document scores are used to rank the documents. While this merged results list is consistent, this approach is computationally expensive.

Normalized Merge. Normalized merge approaches assume that global collection information is available, allowing document scores from different collections to be adjusted before they are used to rank documents in a merged list. This allows the merged list to be exactly the same as the results list that would have been returned if all documents were found in a single collection. The main drawback to this approach is the cost of maintaining and disseminating the global collection information. This approach is one to which proposed merging techniques are frequently compared.

2.2.2.1 Query Combination and Data Fusion

A lot of the research on results merging has been done in the context of *query combination* or *data fusion*. *Query combination* utilizes a collection of documents indexed by a single information retrieval system. Given a single information need, multiple queries are constructed. Each of those queries is a different attempt to express the information need, or an

expression of different facets of the information need. Each query is issued to the collection, producing one results list for each query. *Data fusion* employs multiple indexes of a collection. These indexes may be produced by different information retrieval systems and/or by using different parameters of a single information retrieval system. Given an information need, a query (in the appropriate format for the information retrieval system) is issued to each index of the collection, producing one results list per index. For both query combination and data fusion, we are dealing with a single collection of documents. As a result, there is the potential for overlap among the results lists. In fact, these approaches are based upon observations that different query formulations and indexing approaches tend to produce results lists containing different relevant and non-relevant documents [KMT+82, SK88] but that documents appearing in more than one list have a higher probability of being relevant [SK88]. The task for both query combination and data fusion is to account for duplicates and to merge the individual results lists into a single result list to be presented to the user. The goal is to rank relevant documents from the individual results lists highly in the merged list.

What follows is not a comprehensive survey of data fusion and query combination research, but rather a brief summary of the research and an entry point into the literature. Our primary interest is the results merging approaches that have been employed for data fusion and query combination. We summarize some of the combination approaches suggested, the results observed, and the degree to which these combination approaches have been adopted.

For their TREC-2 experiments, Fox and Shaw [FS93] suggested six different approaches for combining multiple "individual similarity" values for a given document d. These individual similarity values may be due to retrieval by different query formulations or different document indexing approaches. These approaches were later adopted by other researchers so we summarize them below for reference:

CombMAX	maximum of individual similarities
CombMIN	minimum of individual similarities
CombMED	median of individual similarities
CombSUM	sum of individual similarities
CombANZ	CombSUM $\div$ number of nonzero similarities
CombMNZ	CombSUM $\times$ number of nonzero similarities

Fox and Shaw [FS93] found that, on average, all six approaches performed fairly well. CombSUM and CombANZ generally outperformed the best-performing of the constituent approaches, with CombSUM often outperforming CombANZ. They noted that the CombSUM performance relative to the constituent approaches varied greatly. CombSUM was adopted by Lee [Lee95] for a study of different classes of document indexing approaches. Lee [Lee97] later expanded upon the type of experiments performed by Fox and Shaw [FS93] but employed a different test environment. He studied the CombMAX, CombMIN, CombSUM, CombANZ and CombMNZ approaches, arguing that CombMNZ should be more effective given that it favors documents that are retrieved more frequently. CombMNZ was found to be more effective within Lee's test environment.

For their TREC-2 experiments, Belkin *et al.* [BKCQ93] used the simple approach of issuing multiple query formulations as a single query, automatically producing a single results list. They also used a variant of the median score rule (see CombMED above). In later experiments, Belkin *et al.* [BKFS95] also considered using training information about query performance to choose the best query formulation. Finally, a collaborative effort by Belkin, Kantor, Fox and Shaw [BKFS95] considered the problem of combining results obtained from different information retrieval systems. Variants of using the minimum, maximum and sum of scores were used.

Rajashekar and Croft [RC95] used the standard Inquery query operators to combine different representations for both queries and documents.

2.2.2.2 Merging and Multi-collection Retrieval

In a multi-collection retrieval environment, an appropriate results merging step is dependent on the information retrieval system(s) at the collections. The merging approaches proposed by different researchers depend upon whether knowledge is available about the information retrieval systems used by the collections and whether document scores from different results lists are comparable or can be normalized to make them more comparable.

Voorhees, *et al.* [VGJL95, VGJL94] proposed a merging approach in which the number of documents retrieved from a collection was based on the estimated usefulness of that collection. They used two approaches, modeling relevant document distributions (MRDD) and query clustering (QC) to determine the number of documents retrieved from a collection. Both approaches required a set of training queries. The retrieved documents were merged using a probabilistic approach that employed what they termed a *C-faced die* to choose the next collection from which a document was to be drawn. Voorhees *et al.* [VGJL95] noted that when the actual distribution of relevant documents is known and utilized, the merged performance can exceed system performance when a single collection is employed. Further experiments by Voorhees [Voo95] studied QC and MRDD using different collections and queries and discussed the efficiency of the techniques. Voorhees and Tong [VT97] later showed that results observed in Voorhees, *et al.* [VGJL95, VGJL94, Voo95] were stable over different underlying search engines.

Yager and Rybalov [YR98] considered the merging problem as stated by Voorhees, *et al.* [VGJL95, VGJL94] but enumerated several deterministic merging approaches as an alternative to the original probabilistic approach.

In the same paper in which they introduced the *CORI* collection selection algorithm Callan *et al.* [CLC95] also presented a comparison of four different results merging approaches. These approaches included interleaving, raw score merge, normalized merge and an approach they termed *weighted scores*. The weighted scores approach did not require collection-wide information, but instead used a combination of the document score and the source collection's score from the collection selection step to compute a document score for

merging. Callan *et al.* found that the weighted scores approach had similar performance to the normalized merge, but with less overhead.

In conjunction with their collection selection experiments, Meng *et al.* [MLY+98] considered the collection fusion problem with the goal of guaranteeing that they would locate all of the documents that would be most similar to some query q if a global similarity function had been employed. This necessitated computing different similarity thresholds for retrieving documents from each selected collection. Meng *et al.* presented two approaches for retrieving the globally most similar documents; however, because their their test environment did not contain relevance judgements, a relevance-based evaluation was not presented. If the global similarity function is used to order the retrieved documents, the relevance-based performance of this approach would be bounded by the single-collection performance.

Gravano and García-Molina [GGM97] also considered the problem of results merging and setting local similarity thresholds to retrieve the globally most similar data items. However, they considered collections containing structured records and examined the impact of different assumptions about the types of information available from and returned by the collections.

Craswell, *et al.* [CHT99] proposed two new merging techniques and compared their performance to other published techniques. They compared results merging approaches that used different sources of document and collection information, and found that re-weighting documents retrieved from collections was highly effective. Their two proposed re-weighting approaches, based on the position of query terms in the documents, performed well for their test environment of five TREC-based collections.

While results merging is an important step for multi-collection retrieval, we focus most of our experiments on the collection selection step. There are many different merging approaches to choose from; we chose approaches that did not obscure the impact of the collection step. The merging approach is generally held constant for our comparative experiments. For the one case where merging is a potential issue, it is clearly discussed. When

merging is necessary for our experiments, we use a raw score merge (where appropriate) or the default *CORI* merge. The experiments of Chapter 10 use a document re-weighting approach for results merging.

2.3 Issues in Multi-collection Retrieval

There are a number of issues that cut across many multi-collection retrieval approaches, although they are not always explictly mentioned. These issues include the impact of the use of different information retrieval systems at the underlying collections and information used for indexing collections. Of particular interest to us is the issue of collection representations for collection selection, the information used for those representations and how that information is acquired.

2.3.1 Heterogeneous Collections

There are many ways in which the individual collections in a multi-collection environment might differ. Collections can employ different document indexing techniques and different query processing techniques. The underlying search engine at one collection might support options not allowed by others. The acceptable query formats may differ among the search engines at the collections. The range of document scores can also vary by collection. In the context of describing the STARTS internet metasearching protocol, Gravano *et al.* [GCGMP97] discussed these issues in detail. Here, we mention a few specific issues that have implications for our experiments.

Some approaches assume homogeneous underlying search engines. The general approaches that we consider do not. In our earlier discussion of the *gGlOSS* [GGM95] collection selection approach, we noted that *gGlOSS* assumes that all collections use the same similarity function to compute query-document similarities. In practice, this assumption is only feasible when the same individual or organization controls all collections in the multi-collection environment.

With the exception of *gGlOSS*, the approaches that we study in this dissertation do not require document term weight information. In general, the approaches we study require only document frequency information (the number of documents containing each term) and other information that remains consistent even if different search engines or query formats are used at the underlying collections. However, there is still one issue that we must be aware of. Because collection selection indexes may be built from information provided by collection indexes, tokenizing, stopping and stemming can have implications for collection selection. Differences here can lead to incompatible vocabularies.

2.3.2 Indexing Issues

In addition to search engine issues, some researchers have also considered problems caused by limited indexing information at the individual collections. If collections are small or specialized, the documents in the collection may not be representative of the set of documents contained in all collections. This can result in artificially high or low similarity scores. One approach to resolving this problem has been to incorporate statistical information from other collections during indexing.

Viles and French [VF95b, FV96] studied the impact of what they termed *collection wide information (CWI)* on multi-collection information retrieval performance. Their test environment assumed that queries are broadcast to all collections and that a raw-score merge would be employed. They found that while disseminating collection wide information for use in indexing is advantageous, it is not necessary to collect statistical information from all documents in all collections. The amount of statistical information required is dependent upon characteristics of the underlying collections.

Walczuch *et al.* [WFPS94] compared their system performance using semi-local and collection-wide information. They tested using five collections indexed using the SMART information retrieval system. Results from the five collections were merged by re-ranking using either *df* information from the pooled result set or collection-wide *df* information. Walczuch *et al.* reported that they found no significant difference between the approaches;

however, given the small number of collections, it is difficult to compare their results to those of Viles and French [VF95b, FV96] or the results reported in this dissertation.

de Kretser *et al.* [dMSZ98] considered three different levels of statistical information and collection-wide information for collection selection and indexing. The simplest approach provided nothing to the collection selection mechanism except a list of collections. A more elaborate approach provided vocabulary information about the collections to the collection selection approach. That vocabulary information was used for selection and also used as collection-wide information for document retrieval. The collection block selection approach of Moffat and Zobel [MZ95] was employed as a third approach. Overall, de Kretser *et al.* reported similar results for the three approaches, with a small drop in performance when very few blocks were selected by the third approach. The similar performance of the first two approaches is of particular interest. From the description of experiments, it appears that for both approaches queries are broadcast to all collections. The lack of impact of collection-wide information implies that for the given test environment local collection statistical information was representative of the test environment as a whole.

In their work on results merging Craswell, *et al.* [CHT99] used what they termed *reference statistics*. Instead of maintaining statistical information about all documents in all collections, they maintained information about a sample of 10% of the documents. These reference statistics were employed by their top-performing merging approach.

2.3.3 Collection Representations

Collection selection is difficult partly because collection selection algorithms do not typically have access to the full contents of a collection. Instead, they utilize summary statistical information about the collections. We will use the terminology used by Xu and Croft [XC99] and Callan *et al.* [CCD99] and refer to the summary information about a collection as a *language model (LM)*. Collection selection algorithms differ in the type of information that they require in the language models. For our experiments, each collection C_i is represented by a corresponding language model LM_i. We denote the set of language models

as $\mathcal{LM} = \{LM_1, LM_2, ..., LM_N\}$. Given a query q, a formula for estimating merit, and an appropriate set of language models $\mathcal{LM}$ representing $\mathcal{C}$, a collection selection algorithm computes the estimated merit for each collection with respect to the query, then sorts and ranks collections using that estimated merit.

The language model LM_i for a collection C_i can be created in a number of ways. The administrators of a collection might choose to make all documents available, in which case a wide variety of statistical information is possible. Alternately, administrators may make only limited statistical information available.

A number of protocols for describing collections have been proposed. One well-known protocol is STARTS [GCGMP97], which was created with the input of a number of large information providers. STARTS defines a rich set of metadata that is to be provided by internet search engines that wish to cooperate in a joint metasearching project. The assumption that the search engines wish to cooperate is key—STARTS requires a large amount of metadata, some sites may not be willing or able to contribute.

Powell and Fox [PF98] defined SearchDB-ML, an application of XML (eXtensible Mark-up Language) for describing a collection. SearchDB-ML can be used to describe the search engine used at a collection, the query and results formats, and additional metadata. To enable collection selection, each SearchDB-ML description of a collection can contain a brief general description of the contents. The authors report that the simple descriptions of SearchDB-ML make it easy to add new collections to the system; however, they also report that some users expressed concern that the brief collection description would not be sufficient for effective collection selection.

Hawking and Thistlewaite [HT99] suggested that their Lightweight Probes (LWP) approach may be a useful addition to STARTS. Rather than periodically collecting metadata from collections, the LWP approach collects limited, query-specific statistical information from collections at query time. Like STARTS, LWP assumes that the underlying collections are willing and able to cooperate. The information requested by LWP includes document frequency information, the number of documents in the collections, as well as proximity

and co-occurrence information.

As we mentioned earlier, *gGlOSS* [GGM95] needs two vectors of information from each collection in order to make its estimates: the document frequency df_j for each term t_j and the sum of the term weights w_{ij} of each term over all documents d_i in the collection. This information is stored in two matrices, referred to as F (document frequency) and W (term weights) by Gravano and García-Molina.

CORI [CLC95] is built on top of Inquery [CCH92] and as defined draws its required document frequency information directly from the Inquery-indexed collections. However, the published algorithm is amenable to use with heterogeneous underlying collections if collection statistics can be provided or acquired by sampling. The use of phrases and query expansion by Xu and Callan [XC98] increased the complexity of the basic *CORI* approach. The use of phrases increases the size of the language model and also increases the complexity of gathering the statistical information required. Their query expansion approach required a training phase.

Callan *et al.* [CCD99] discussed a query-based sampling approach to generating language models. The language model for each collection is constructed using a subset of the documents in that collection. The subset of documents is gathered using a randomly selected set of probe queries. Callan *et al.* described the methodology for constructing the sample-based language models, covering the number of documents retrieved per probe query, the selection of probe queries and stopping criteria. In this work, Callan *et al.* measured the degree to which the sampled language models approximate language models built using all documents in the collections. While there is discussion of automatic stopping criteria, the reported results use samples of 300 or 500 documents per collection. Callan *et al.* [CPFC00] later studied the degree to which different collection selection approaches are affected by language models created using query-based sampling. Collection selection performance of *CORI* [CLC95], *gGlOSS* [GGM95] and *CVV* [YL97] was compared using complete and sample-based language models. Long, medium and short versions of the queries were used. With a few exceptions, Callan *et al.* found that collection selection using

sample-based language models was effective. There was often little difference between collection selection performance using complete and sample-based language models. However, they did note some issues related to normalization approaches—as currently defined, *CVV* was found to be incompatible with sample-based language models. Overall, the *CORI* collection selection approach proved to be the most stable.

Craswell *et al.* [CBH00] also presented a comparison of the impact of sample-based language models on the *CORI*, *gGlOSS* and *CVV* collection selection approaches. There were a number of methodological differences; however, the overall results are complementary with those found by Callan *et al.* [CPFC00]. For their experiments, Craswell *et al.* [CBH00] used a variant of the query-based sampling approach proposed by Callan *et al.* [CCD99]. Instead of using randomly-selected probe query terms, Craswell *et al.* used an independent set of multi-term queries. They considered the performance of the collection selection approaches when a varying number of probe queries were used to construct the language models. Craswell *et al.* found that the eventual merged document retrieval performance tended to improve as more probe queries (retrieving more documents) were used to construct the language models. They also found that reasonable performance could be obtained when selecting only ten collections and that *CORI* performed well using sampled language models.

Liu *et al.* [LYM+99] touched upon the potential unavailability of full statistical information for their approach and outlined general sampling approaches to acquire estimates for the required information. Xu *et al.* [XCLN98, LXLN99] also used a query-based sampling approach (different from that defined by Callan *et al.* [CCD99]) to acquire statistical information for their experiment using collections of bibliographic data. Of particular interest is a characterization of the overhead incurred by sampling [LXLN99].

D'Souza and Thom [DT99] suggested *n-term indexing* in which only n terms from each document in a collection are used to construct that collection's representation. D'Souza and Thom suggested a number of ways that the terms could be chosen and that the value of n could be determined. For initial experiments [DT99], they chose the first 30 unique

terms occurring in each document. They used the experimental environment proposed by Zobel [Zob97] and compared their results to Zobel's inner product results. D'Souza and Thom reported that n-term indexing performs noticeably worse than Zobel's inner product but better than selecting collections based solely on size (largest collections first). The n-term indexing approach represents a departure from many of the other language modeling approaches because it chooses terms on more of a document-centric basis. Unlike some other approaches, n-term indexing requires access to the text of all documents in a collection and also assumes that early terms are not only representative of the document but also of the collection.

Xu and Croft [XC99] considered clustering, both in the creation of collections and the construction of language models. They argued that clustering may be necessary to create multi-collection environments suitable for effective collection selection. We will revisit this argument in Chapter 9. Xu and Croft compared single-collection performance to four different approaches for constructing collections and language models. They considered a set of collections containing roughly the same number of documents per collection (their base case), a set of collections created by clustering all documents, a set of collections created by clustering within a coarse decomposition of documents, and the use of multiple language models to represent each collection. Xu and Croft reported improved performance over the base case for all three proposed approaches. Unfortunately, the overhead of clustering and the requirement that the underlying collections cooperate may impact the broad applicability of these approaches.

2.4 Early Internet Approaches

Early internet resource discovery systems, generally those in place shortly before the World Wide Web began to be widely used, often focused on specific tasks within an environment with multiple information sources. Examples include locating e-mail addresses or locating files served by anonymous FTP servers. Broader tasks included locating potentially useful

WAIS or gopher servers. Schwartz [Sch93], Schwartz *et al.* [SEKN92] and Obraczka *et al.* [ODL93] all provide excellent surveys of these approaches. Bowman *et al.* [BDMS94] also consider some of these approaches in the context of scalability. At the time that these resource discovery approaches were surveyed, the problem was one of notifying users of the existence of distributed information sources that had the potential to be useful. As a result, a basic form of collection selection (identifying potentially useful sources) was involved; however, the user was generally responsible for selecting which information sources were employed.

The Discover system [SDWG95] was implemented on top of a set of WAIS servers using an approach that the authors referred to as "content routing" [SDW+94]. A hierarchical set of content routers contained descriptions of the information sources (WAIS servers); the format of the descriptions was not constrained by the system. For the prototype system, the descriptions were the very brief WAIS server description and the WAIS catalog file (headlines for each document). A user interface was provided so that users could search or browse these descriptions to identify potential information sources to search. Once a user selected a set of collections to search, Discover provided a mechanism to send the query to all of the collections. The results list was not merged but rather delimited by the collection providing each set of documents. Discover differed from other systems of the time by providing a query reformulation feature. The authors provided an example and a qualitative performance report but reported no in-depth system evaluation.

The Harvest system [BDH+95, BDH+94] employed sets of *gatherers* and *brokers* to provide efficient access to information across multiple collections. Gatherers were responsible for exporting indexing information about information sources, while brokers provided organization, plus a search interface. Efficiency was one of the primary goals of Harvest—gatherers could provide information to multiple brokers, removing the need for each broker (search mechanism) to download documents for indexing.

2.5 Metasearching

Internet metasearching is an interesting sub-problem of multi-collection information retrieval. A metasearch engine does not maintain its own index of WWW pages but instead acts as an intermediary that forwards queries to an often predetermined set of traditional internet search engines. Metasearch engines have been around almost as long as internet search engines. One interesting aspect of metasearch engines is the very large scale on which the internet search engines themselves operate. For example, as of July 2000, Search Engine Showdown[4] estimated that ten major internet search engines indexed at least 100 million pages. Unfortunately, studies have shown that any given search engine indexes only a small fraction of the total pages available on the WWW [LG99]. Metasearching is one attempt to broaden search coverage.

The potential utility of metasearch engines is illustrated in the accessibility experiments of Lawrence and Giles [LG99]. Their experiments, conducted in February 1999, estimated eight hundred million web pages, with at most 16% of those pages indexed by any one search engine. The observed overlap among the pages indexed by any two search engines was small, suggesting that sending queries to more than one search engine may improve the comprehensiveness of the results. Taken together, the eleven engines considered by Lawrence and Giles [LG99] covered an estimated 42% of the pages.

Because metasearch engines exist in the changing WWW environment, evaluations of published metasearch engines have varied widely in methodology. In general, recall and precision-based results of the form commonly found for information retrieval experiments are not reported.

Selberg and Etzioni introduced MetaCrawler [SE95] in July 1995. A current operational version of MetaCrawler is available[5], but we will discuss the published version [SE95]. MetaCrawler accessed Galaxy, InfoSeek, Lycos, Open Text, WebCrawler and Yahoo[6], then

[4] `http://www.searchengineshowdown.com/stats/sizeest.shtml`

[5] `http://www.metacrawler.com`

[6] `http://www.galaxy.com/`, Infoseek is now `http://www.go.com/`, `http://www.lycos.com/`, Open Text no longer supports a general-purpose search engine but now provides a business-oriented site `http://pinstripe.opentext.com`, `http://www.yahoo.com`.

collated the results. For evaluation purposes, the authors kept track of which returned links were followed, which returned links were unique to each search engine, and the response time of the underlying search engines. For the purposes of evaluating effectiveness, the authors assumed that followed links were useful to the user. This is a common assumption in metasearching experiments; however, the validity of the assumption is dependent upon a number of factors. Poor summary information can influence whether or not a link is followed, and users may follow an interesting link that does not satisfy the information need. Keeping track of which returned links were followed allowed the authors to determine which search engines provided followed links. They found that on average all search engines provided followed links but that some appeared to provide more than others. They also noted low overlap among the results lists from the search engines. These two factors combined to suggest that MetaCrawler provided access to a broader array of interesting web pages. MetaCrawler loaded the pages from the results lists to elide dead links and to enable postprocessing steps like advanced query formulations. MetaCrawler was an early metasearching approach and as a result Selberg and Etzioni also focused on the overhead of the approach and the apparent acceptability to users.

SavvySearch was introduced by Dreilinger and Howe [DH97] in March 1995. SavvySearch is now available as C|Net search.com[7] but we will discuss the published version. SavvySearch accessed both general-purpose search engines and specialized resources. One interesting feature of SavvySearch was the use of search engine selection for efficiency purposes. SavvySearch employed a selection approach, referred to as a *search plan* that attempted to balance resource usage and the expected quality of the results. Given a user query, search engines were ranked based upon whether they had previously returned results for terms in the query, whether those links returned had been followed and the recent search engine response time. Dreilinger and Howe reported a number of experiments designed to determine whether the selection approach was viable and to gauge the quality of the suggested search order. Overall, they found that selection was viable (i.e. it is not

[7] `http://www.search.com`

necessary to broadcast a query to all search engines available to the metasearcher) but that modifying the search order made only a small difference, possibly due to the presence of general-purpose search engines in the search plans.

ProFusion[8] was recently acquired by IntelliSeek[9]. The version of ProFusion reported by Gauch *et al.* [GWG96] was notable for the discussion of merging issues. The published version of ProFusion had the potential to send queries to AltaVista, Excite, InfoSeek, Lycos, OpenText and WebCrawler[10]. The default of ProFusion was to send queries to InfoSeek, Lycos and Excite; alternately, a user could manually select search engines. An additional option was to enable a search engine selection step that classified the query by topic then select three search engines based on that topic classification. A major focus of ProFusion was a variety of postprocessing steps. Given results from the selected search engines, ProFusion first took steps to remove duplicate pages, then merged the results using a combination of the document score reported by the search engine and the estimated accuracy of the search engine returning the page. Pages retrieved by more than one engine were given the maximum score achieved by any instance of the page. ProFusion was compared to the six constituent search engines plus MetaCrawler[SE95] and SavvySearch[DH97] using twelve queries. ProFusion was found to outperform the constituent search engines and the other metasearchers in terms of locating relevant documents and in terms of removing duplicate pages and broken links.

MetaCrawler, SavvySearch and ProFusion were all included in a comparison of metasearch engines conducted by Repman and Carlson [RC99] and all three ranked in the top five. Repman and Carlson focused on whether metasearch engines were appropriate for use at library terminals accessible to a variety of users, and as a result did not perform an in-depth technical evaluation of the metasearch engines. However, their observations about the usability and strengths and weaknesses of different metasearch engines provide useful

[8] `http://www.profusion.com`

[9] `http://www.intelliseek.com`

[10] `http://www.altavista.com`, `http://www.excite.com`, Infoseek is now `http://www.go.com/`, `http://www.lycos.com/`, Open Text no longer supports a general-purpose search engine but now provides a business-oriented site `http://pinstripe.opentext.com`, `http://www.webcrawler.com`.

insight.

Inquirus [LG98a, LG98b] is a prototype metasearch engine with a heavy emphasis on processing documents retrieved from the individual search engines. Inquirus downloads the retrieved pages, verifies that the pages are still available and that they still contain the query terms. Because the pages are downloaded, Inquirus can re-rank the retrieved documents, taking into account query term proximity, and can construct its own document summaries. Documents producing the same summaries are declared to be duplicates. Another feature of Inquirus is "specific expressive forms" in which common question forms are rephrased in an attempt to improve recall. Inquirus begins displaying results before all documents are processed. As a result, the Inquirus response time is shown to be on par with that of other search engines, despite the additional processing. Search effectiveness results were not reported.

In addition to the general-purpose metasearch engines that we have discussed so far, there are also multisearch and metasearch systems that target specialized search engines. Search Broker [MB97] prompts users to specify the subject of a query as the first term of the query; that subject term is used to direct the query to a search engine. Search Broker does not merge results, but presents summary results delimited by responding search engine, therefore, it is classified as a multisearch system. Sugiura and Etzioni [SE00] described Q-Pilot, a multisearch system that routes queries to specialized, topic-based internet search engines. Sugiura and Etzioni compared three different methods for building collection representations and also considered query expansion.

There have also been a number of less-widely-publicized research metasearch systems. Smeaton and Crimmins [SC97] discussed a prototype metasearch system that incorporates relevance feedback and query expansion[11] functionality. This prototype system is notable because it proposes a layer of functionality that need not be available from the underlying search engines. In addition to standard metasearch, the Federated Searcher system

[11] A system that employs relevance feedback uses relevance information about initial search results provided by users to improve later results. This relevance information can be used for query expansion, in which new terms are added to the initial query.

described by Powell and Fox [PF98] addresses the problem of query translation for a multilingual federated environment.

2.6 Summary

Because there are so many facets to multi-collection retrieval research, there is a broad array of related work. In this chapter, we first gave a brief overview of single-collection information retrieval to provide context and to introduce concepts that apply to both single- and multi-collection retrieval. We then covered different facets of the multi-collection retrieval problem and work that has taken place to date. We discussed a variety of collection selection and results merging approaches, then considered additional issues that impact the multi-collection retrieval problem as a whole. We closed with a discussion of the related sub-problem of WWW metasearching.

3

Notation and Definitions

In this chapter, we formalize some of the concepts and components that were touched upon and defined somewhat loosely in Chapter 2. We also expand upon previously-introduced notation. Unfortunately, *collection* and many of the other terms that we employ have been used in other senses, and/or may suggest certain system implementation decisions. Prior exposure to these terms and any preconceived ideas about their exact meaning make it difficult to specify some of the underlying experimental details that are varied in the experiments reported in Chapters 7-9. Unclear definitions are also particularly detrimental to a shift in interpretation that will be presented in Chapter 9. As a result, we will carefully define a number of concepts, some of which are apparently trivial, then discuss assumptions that were made and relationships among the defined concepts.

In presenting more formal definitions, we will start from the bottom up. We'll begin by defining the components involved (e.g. data items, collections), then define operations that are stages in multi-collection retrieval (e.g. selection, merging) and concepts that are used in evaluation. The specifics of the actual test environment that we will use are covered in Chapter 4. While we begin defining concepts crucial to evaluation here, the evaluation measures are defined in Chapter 5.

3.1 Components

3.1.1 Data Items and Collections

Let $\mathcal{D} = \{d_1, d_2, ..., d_{|\mathcal{D}|}\}$ denote a set of data items. At this point, we make no assumptions about the organization or storage of these data items. Each d_i is a data item that may be retrieved in response to a user request. The data items might be text documents, images, sound recordings or binary files. The notation presented here can apply to data items of any format. In the examples and experiments presented in this dissertation, all data items are text documents and will be referred to as *documents*.

We will use the term *collection* to refer to a selectable, searchable group of data items. Membership in a collection is to be considered an organizational issue rather than a storage issue. In other words, data items need not be stored on the same server to be members of the same collection.

Let $\mathcal{C} = \{C_1, C_2, \ldots C_N\}$ represent a set of N collections. Each collection conceptually contains a set of data items. We assume that no duplicate data items occur within a collection; we will discuss the potential for duplicate data items across collections in a moment. $\mathcal{C}$ is defined as follows:

$$C_1 = \{d_{11}, d_{12}, \ldots d_{1|C_1|}\}$$
$$C_2 = \{d_{21}, d_{22}, \ldots d_{2|C_2|}\}$$
$$\ldots$$
$$C_N = \{d_{N1}, d_{N2}, \ldots d_{N|C_N|}\}$$

where d_{ij} is the jth data item in collection C_i and $d_{ij} \in \mathcal{D}\ \forall i, j$ where $1 \leq i \leq N$ and $1 \leq j \leq |C_i|$. $\cup_{i=1}^{N} C_i = \mathcal{D}$.

In an operational environment, duplicate data items may exist in the collections. If duplicate data items exist, they may occur in different arrangements. We first assume that duplicates are not collocated in a collection C_i, i.e., $d_{ij} \neq d_{ik} \forall i, j, k, j \neq k$. Having eliminated that case, and given a set of collections $\mathcal{C}$ there are a number of remaining possibilities.

1. at least one collection is *replicated*, that is, $\exists i, j$ such that $C_i = C_j$.

2. at least one data item is *duplicated*, i.e., found in more than one collection, that is, $\exists i, j$ such that $C_i \neq C_j$ but $C_i \cap C_j \neq \emptyset$.

3. both cases (1) and (2) occur.

In this work, we assume that each data item has a unique identifier. Data items with the same unique identifier are by definition duplicates. Given an environment where unique identifiers are not available, duplicate detection is more difficult. The general problem of duplicate detection is beyond the scope of this work; however, it will be revisited briefly in Chapter 10.

The existence of replicated collections or duplicate data items, generally out of our control in operational environments, can complicate multi-collection information retrieval. Replication and duplication are problematic during the collection selection step. Considering replication, if we have $C_i = C_j$, we want to select either C_i or C_j but not both. The presence of duplicated data items complicates both collection selection and evaluation. Assume that collections C_i and C_j both contain useful data item d_k. If collection C_i is selected and data item d_k retrieved, the usefulness to the user of data item d_k in collection C_j requires consideration. While data item d_k still satisfies the information need, the fact that it has already been retrieved may impact an evaluation of the advisability of selecting collection C_j. Buckland [Buc95] and Buckland and Plaunt [BP97] provide a thoughtful consideration of these issues in the context of searching multiple digital libraries.

Duplication or replication can also complicate results merging, the process by which retrieved data items from selected collections are integrated into a single list for presentation to a user. Once again, assume a data item d_k and two selected collections C_i and C_j where $d_k \in C_i \cap C_j$. If d_k is retrieved from both C_i and C_j, its placement in the merged list must be resolved. Due to the intrinsic replication found in the approaches, work in the areas of query combination and data fusion (see the discussion in Chapter 2) considered this issue.

The issues of duplication and replication complicate the multi-collection retrieval prob-

lem but are resolvable. In the majority of our work, we assume that $\mathcal{C}$ is a partition of $\mathcal{D}$, obviating these issues. However, in a more general federated system or a WWW environment, the assumption that $\mathcal{C}$ is a partition of documents is rarely valid. Our experiments on collection selection for metasearching, reported in Chapter 10 consider the problem of duplicate data items.

3.1.2 Indexes and Information Available for Indexing

For ease of exposition, we will at times refer to executing a query at a selected collection. In fact, the collection is just the selectable, searchable group of data items. A means of searching the collection, i.e. retrieving data items that may satisfy the user's information need, is necessary. As a result, we will assume that some information retrieval system is in place for each collection. Given a query in an appropriate format, the information retrieval system will return a set or ranked list of data items.

Put more formally, for the set of collections $\mathcal{C}$, we also assume a corresponding set of indexes $\mathcal{I} = \{I_1, I_2, \ldots I_N\}$ where I_i contains indexed representations for each data item d_{ij} in collection C_i. These indexes are produced by a potentially unknown information retrieval system. The information retrieval system used at collection C_i need not be the same as that used at collection C_j. Information used to create representations of collections for collection selection purposes may be extracted from or provided by these indexes.

Given a user query, the query representation is compared to data item representations to identify data items to be displayed to the user.

Unless otherwise noted, the index I_i for collection C_i is constructed using information gathered from the set of data items found in C_i. The information gathered is dependent upon the information retrieval system used to perform the indexing. However, additional information may sometimes be available. An external server may provide limited statistical information about other collections, for example, *df* information. Collections in a federated environment may also communicate periodically to share statistical information [VF95b]. In these cases, and for experiments reported in Chapters 9 and 10, we occasionally specify

a set of data items that contribute statistical information to index I_i. Unless otherwise noted, the set of data items that contributes statistical information to I_i is C_i.

3.1.3 Language Models

As discussed in Chapter 2, we will use the terminology *language model* for the collection representation used for collection selection purposes. For the set of collections $\mathcal{C}$, we assume a set of language models $\mathcal{LM} = \{LM_1, LM_2, \ldots LM_{|\mathcal{LM}|}\}$. In the general case, $|\mathcal{LM}|$ may be larger or smaller than N; a relation $R_{\mathcal{LM}\rightarrow\mathcal{C}}$ (discussed further in Section 3.2.2) specifies the correspondence between language models and collections. For all experiments reported in this dissertation, $|\mathcal{LM}| = N$ and $\mathcal{LM}$ corresponds directly with $\mathcal{C}$. More specifically, $\mathcal{LM} = \{LM_1, LM_2, \ldots LM_N\}$ where LM_i is the language model representing collection C_i.

Language models may be created using only a subset of the data items contained in a collection. For example, sampling techniques result in language models built from a subset of the data items. The default for our experiments is that the language model LM_i is constructed using information about all data items in collection C_i. We will note when this is not the case.

3.1.4 Queries and Relevance Judgements

Queries are either provided directly by a user or are created based on statements of a user's information need. We use the latter approach. Relevance judgements may be supplied as users view the retrieved results or in the case of a test collection may be provided by third party judgements.

We will refer to the set of queries used in experiments as $\mathcal{Q} = \{q_1, q_2, \ldots, q_{|\mathcal{Q}|}\}$. A set of statements of user's information needs may be used to produce multiple sets of queries.

In our test environments, we also have access to relevance judgements. For the TREC data, TREC relevance assessors have determined which text documents from the TREC data satisfy each stated information need. We apply those judgements to text documents retrieved in response to our queries created from those statements of information need.

The relevance judgements can be represented as three-tuples of query identifier $q_i \in \mathcal{Q}$, data item identifier $d_j \in \mathcal{D}$ and relevance judgement rel_judge_{ij}. In our case, relevance judgements are binary. If d_j satisfies the information need of q_i, then $rel\_judge_{ij} = 1$, otherwise $rel\_judge_{ij} = 0$. The set of relevance judgements $\mathcal{J}$ is a set of three-tuples where $\mathcal{J} = \{(q_i, d_j, rel\_judge_{ij})\}$.

3.1.5 Testbeds and Test Environments

Traditional information retrieval test environments are often referred to as *test collections*, encompassing the documents, queries and relevance judgements that are distributed as a set. However, we have specifically defined the term collection to refer only to a searchable group of data items. To avoid confusion, we will avoid the terminology *test collection* unless we are referring to a single-collection test environment.

We will use the terminology *test environment* to refer to the combinations of collections, queries and relevance judgements we use to evaluate collection selection approaches and multi-collection information retrieval systems.

We can think of each test environment as a three-tuple $(\mathcal{C}, \mathcal{Q}, \mathcal{J})$. We can compare test environments by noting differences in any element of the three-tuple. For all of the experiments reported in Chapters 7–9, the relevance judgements $\mathcal{J}$ will remain constant. While we vary the sets of queries $\mathcal{Q}$ used for our experiments, the major comparative performance differences seen in evaluation are due to changes in the sets of collections $\mathcal{C}$ that we use. As a result, a great deal of discussion is focused on these sets of collections. We will refer to these sets of collections as *testbeds*. Each testbed also has a mnemonic name to facilitate discussion.

The creation of a testbed for our laboratory-based experiments requires a set of data items $\mathcal{D}$ and a relation $R_{\mathcal{D}\to\mathcal{C}}$ that specifies the placement of data items in collections such that a set of collections $\mathcal{C}$ can be constructed from the data items in $\mathcal{D}$. In an operational environment, we may have the capability to retrieve data items from $\mathcal{C}$ but the data items $\mathcal{D}$ and the relation $R_{\mathcal{D}\to\mathcal{C}}$ may be unknown. $\mathcal{D}$ and $\mathcal{C}$ may also be variable. For example,

given a metasearching environment in which the collections $\mathcal{C}$ are internet search engines, the data items contained in each collection C_i are not divulged; data items may be added or removed from C_i without notification.

In Chapter 4, we will define three testbeds (*SYM-236*, *UDC-236*, *UBC-100*) that will be used in our experiments. We also define three sets of queries for which relevance judgements are available. This allows the construction of the test environments used in Chapters 6–9.

3.2 Experimental Concepts

3.2.1 Merit, Baseline Rankings and Estimated Rankings

We will refer to a collection ranking that embodies the desired collection selection behavior as a *baseline* or *baseline ranking* and the ranking produced by a collection selection algorithm as an *estimated ranking*. A collection selection algorithm may also be referred to as an *estimator*.

To begin, we assume that each collection $C \in \mathcal{C}$ has some merit, $merit(q, C)$, to a given query q. We expect the baseline to be expressed in terms of this merit. Each collection $C \in \mathcal{C}$ will also have an estimated merit $est\_merit(q, C)$ that is computed by a collection selection algorithm. We expect the estimated merit $est\_merit(q, C)$ is an attempt to implicitly or explicitly estimate the actual merit $merit(q, C)$. Different approaches to computing $est\_merit(q, C)$ were summarized in Chapter 2. The exact computations for *CORI*, *CVV* and *gGlOSS* will be detailed in Chapters 6 and 7.

Let C_{b_i} and C_{e_i} denote the collection in the i-th ranked position of the baseline and estimated rankings respectively. The baseline and estimated rankings are constructed such that $merit\,(q, C_{b_i}) \geq merit\,(q, C_{b_{i+1}})$ and $est\_merit\,(q, C_{e_i}) \geq est\_merit\,(q, C_{e_{i+1}})$. Some of the merit-based evaluation measures presented in Chapter 5 determine the extent to which $merit\,(q, C_{e_i}) \geq merit\,(q, C_{e_{i+1}})$.

3.2.2 Collection Selection

Conceptually speaking, collection selection is easy to describe. Given a query and a set of collections, we wish to either select a subset of the collections to which we will send the query, or order the collections, so that we may send the query to the collections in that order or send the query to the top-ranked n collections. We follow the latter interpretation, which affords us flexibility in the usage of the collection selection results. Given a set of collections, we view collection selection as producing a ranked list of those collections, in decreasing order of estimated merit.

Estimated rankings are constructed such that $est\_merit\,(q, C_{e_i}) \geq est\_merit\,(q, C_{e_{i+1}})$ for $1 \leq i < N$. The way in which $est\_merit(q, C)$ is calculated and the usage of language models in this calculation requires clarification. We use the notation $est\_merit(q, C)$ for broader applicability (some approaches do not use language models) and expositional clarity (we cast the problem in terms of collection selection, not language model selection). However, in the experiments described in Chapters 6-10, estimated merit is computed using the language models, $\mathcal{LM}$, associated with the set of collections $\mathcal{C}$. A correspondence is maintained so that the merit computed for a collection using its language model is associated with the collection. In most cases, the collection selection mechanism does not have access to the contents of a collection, it only has access to the language models.

$R_{\mathcal{LM}\rightarrow\mathcal{C}}$ is the relation that specifies the mapping between language models and collections. Let $R_{\mathcal{LM}\rightarrow\mathcal{C}}$ represent the set of language model, collection pairs where (LM_i, C_i) denotes the default case where the estimated merit computed using LM_i is applied to collection C_i. The default case will apply the vast majority of the time and is the case illustrated in Figure 3.1. One exception found in the literature is the work of Xu and Croft [XC99] in which a collection could be represented by multiple language models.

For our experiments, we assume that an estimator has access to a set of language models, $\mathcal{LM}$ and the mapping relation $R_{\mathcal{LM}\rightarrow\mathcal{C}}$. Given some query q, the estimator produces a set of collection, estimated merit pairs, $\{(C_i, est\_merit(q, C_i))\}$. Sorting based on $est\_merit(q, C_i)$ allows us to produce an estimated ranking $\langle C_{e_i} \rangle$ where $1 \leq i \leq N$. This

notation only specifies the general components and output of a collection selection algorithm. In Chapters 7 and 8, we will discuss a variety of algorithms and focus on the ways in which the statistics provided by the language models are employed.

The estimated ranking can be used to specify a search order or to select the top n ranked collections. The set of collection, estimated merit pairs, $\{(C_i, est\_merit(q, C_i))\}$ can be used to select all collections with an estimated merit greater than a specified threshold.

The specified search order E will be defined in Chapter 5. When either a fixed-size subset of the collections or all collections with an estimated merit greater than a specified threshold are selected, we will refer to the selected subset as $\mathcal{C}_{sel}$. If all collections are selected, $\mathcal{C}_{sel} = \mathcal{C}$.

3.2.3 Results Merging

For some experiments, we are concerned primarily with collection selection performance. For others, we are also concerned with the overall quality of the data items that are retrieved from the multicollection environment. For our experiments, evaluating the quality of the overall data items retrieved requires that a results merging step be employed.

The results merging step takes the individual results lists from each of the selected collections ($results_C$ for each $C \in \mathcal{C}_{sel}$) and produces a single results list ($results_{merge}$).

The specific results merging algorithm that we employ is defined in Chapter 9.

3.3 An Important Detail—Data Item Storage

One important terminology detail was alluded to earlier but needs to be revisited. In these experiments we use collections as a conceptual construct. Data items need not be stored on the same server to be members of the same collection. Consider Figure 3.1. For the purposes of our experiments, the important points are that ten data items can be found in collections C_1, C_2 and C_3, that an information retrieval mechanism is in place for each collection and has produced indexes I_1, I_2, I_3 and finally that language models for each

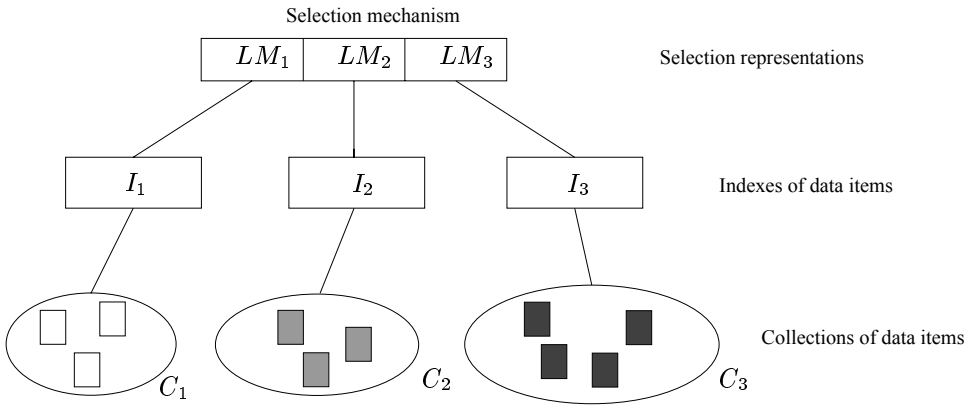


Figure 3.1: Collections, indexes and language models.

collection (LM_1, LM_2, LM_3) are available to a collection selection mechanism.

In an operational environment, it is tempting to think of data item retrieval issues in terms of the physical organization or storage of the actual data items. However, the core issues are the control of and access to the data items. The access/control and organization/storage aspects often parallel one another, but that need not be the case.

Consider Figure 3.2, which is a companion to Figure 3.1. Figure 3.2 parts (a)-(c) illustrate some possible data item organization/storage scenarios that could result in collections $\{C_1, C_2, C_3\}$ from Figure 3.1. Figure 3.2(b) represents a very simple case in which the data items are stored together at storage location L_1 but are subdivided into collections, perhaps by subject area. Figure 3.2(a) is identical to Figure 3.2(b) except that the actual data items are stored in three different physical locations instead of only one.

If the control of the data items is the same in Figures 3.2(a) and 3.2(b) (i.e. the same organization or person has control of the data items), the difference in physical location need not affect the collections, indexes or language models. However, multiple storage locations may imply that different authorities have control over the data items. This can constrain the ways in which the data items are organized into collections and can affect the amount of information available for the construction of language models. For example, consider Figure 3.2(c) and first assume that a single authority has complete access to the

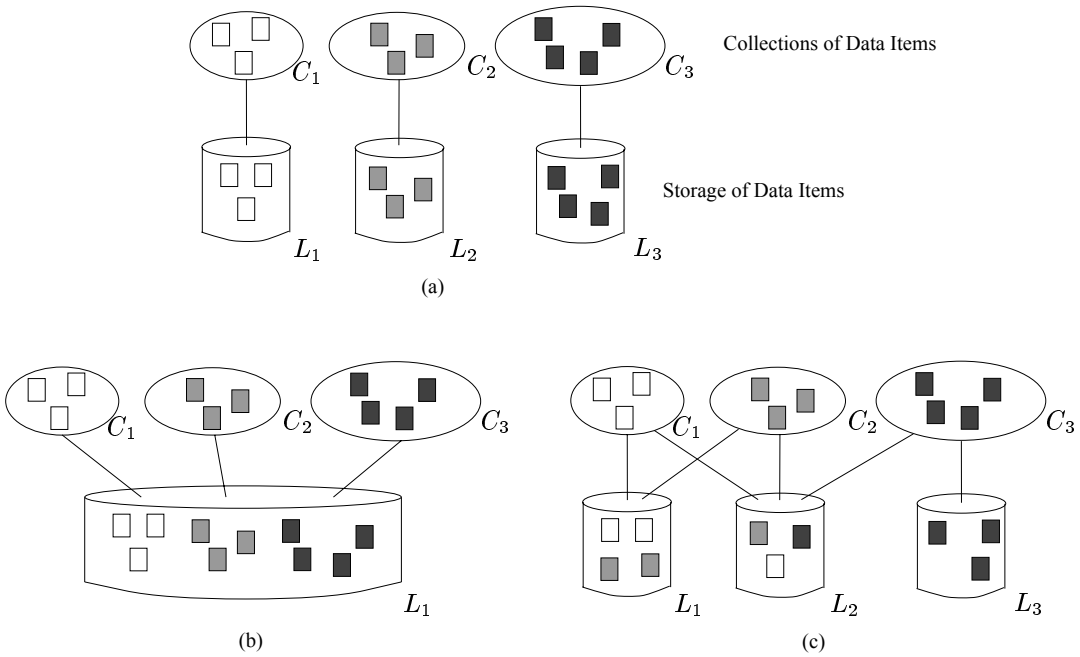


Figure 3.2: Data item storage scenarios.

data items, and that the same authority creates the collections and is responsible for the language models and selection mechanism. In this case, the possibility exists to use a wide variety of information about the data items and/or information from the indexes to create the language models. If instead we assume that three different authorities are responsible for the data items in storage locations L_1, L_2, L_3, the situation changes. First, the creation of collection C_2 would require the cooperation the two authorities responsible for the data items in L_1 and L_2. Second, once collections C_1, C_2, C_3 are indexed, the information available for language model construction may be limited to information that can be extracted from the indexes.

4

Testbeds and Queries

As we discussed in Chapter 2, traditional (single collection) information retrieval test collections usually contain a set of data items $\mathcal{D}$, a set of queries $\mathcal{Q}$ and a set of relevance judgements $\mathcal{J}$. In most widely-available information retrieval test collections, the data items are text documents. Each query represents a statement of a user's information need and for each query, the relevance judgements identify the set of data items that are relevant to the query, i.e. the data items that satisfy the information need.

Multi-collection test environments generally contain the same components as traditional information retrieval test collections, with the exception that the documents are organized into more than one collection. As we discussed in Chapter 3, the potential exists for duplicate data items within a collection. However, in most previously-reported relevance evaluation-based experiments, the collections are a partition of the documents.

Table 2.1 provides a summary of some of the test environments that have been used for multi-collection information retrieval experiments. Because many of the evaluations were relevance based, relevance judgements were needed and researchers were limited to the traditional IR test collections and TREC/TIPSTER data. The large number of documents available in the TREC data made it a popular choice as a basis for constructing multi-collection test environments. At the time that we began these experiments, most of the test environments used in published work had fewer than 100 collections. This, plus an initial

interest in a test environment with a controlled temporal component to the collections led us to construct a different test environment for our experiments. Because we were interested in both efficiency *and* effectiveness, and in evaluating systems using a large number of collections, the TREC/TIPSTER data was the only realistic starting point. Given the availability of TREC topics from which queries could be created and TREC relevance judgements, we began by constructing a testbed, referred to as *SYM-236*. Over the course of the experiments reported in Chapter 6, we noted that some collection selection algorithms have a tendency to prefer collections with a large number of documents. We created an additional testbed, *UDC-236*, to study this effect. We later added a testbed created at the University of Massachusetts, referred to as *UBC-100*.

In this chapter, we will describe and characterize the test environments that are used in the experiments presented in Chapters 6–9. We will focus mostly on the three testbeds that are components of the test environments. We start by describing the underlying TREC data from which each set of documents $\mathcal{D}$ will be drawn. The three testbeds described in this chapter contain no duplicate documents. We then discuss the TREC topics, the subset of the topics used for our experiments and the three sets of queries constructed using those topics. Finally, we describe the three testbeds (sets of collections) used in our experiments and discuss features of those testbeds.

4.1 Features of the TREC Data

The Text REtrieval Conferences (TREC) are a series of annual conferences co-sponsored by the National Institute of Standards and Technology (NIST) and DARPA. Each year, groups from industry, academia and government undertake a set of retrieval tasks, using a supplied set of documents and queries[1], then meet to discuss the results.

The *SYM-236*, *UDC-236* and *UBC-100* testbeds were all constructed using data available to participants in the TREC-4 [Har95] conference. Gross characteristics of the data

[1]The data available to TREC participants is generally referred to as TREC/TIPSTER or simply TREC collections.

Disk	Source	Size (MB)	Size (docs)
1	AP (89)	259	84,678
	DOE	186	226,087
	FR (89)	262	25,960
	WSJ (86-89)	270	98,732
	ZIFF	245	75,180
2	AP (88)	241	79,919
	FR (88)	211	19,860
	WSJ (90-92)	247	74,520
	ZIFF	178	56,920
3	AP (90)	242	78,321
	SJMN (91)	290	90,257
	PAT	245	6,711
	ZIFF	349	161,021
Totals		3,225	1,078,166

Table 4.1: Summary characteristics of TREC data on disks 1, 2, 3. ZIFF from disk 3 and DOE omitted for *SYM-236* and *UDC-236*. (From Harman [Har96])

appear in Table 4.1. To summarize, this data is approximately 3 GB of text spread over several years and from seven primary sources: AP Newswire (AP), Wall Street Journal (WSJ), Computer Select (ZIFF), the Patent Office (PAT), San Jose Mercury News (SJMN), Federal Register (FR), and Department of Energy (DOE). This data was distributed on three CD-ROMs and segments of data are sometimes referenced using the disk number on which they were distributed. Much of the TREC data is from news sources and so has easily identifiable date components. The one undated collection is the set of documents from DOE.

4.2 Queries

The TREC data is distributed along with a set of statements of information need and an accompanying set of relevance judgements. In TREC parlance, the statements of user information need are referred to as *topics*. Unlike average user queries (especially Internet search engine queries), most TREC topics are very detailed statements of information need.

Source Disk	Topic Set				
	1-50	51-100	101-150	151-200	201-250
1	X	X	X	X	
2	X	X	X	X	X
3		X	X		X
1,2	X	X	X	X	
2,3		X	X		X
1,2,3		X	X		

Table 4.2: Coverage of topics over TREC data, disks 1-3, through TREC-4. This table shows topic set, source disk pairs for which relevance judgements are available.

In many cases, they resemble detailed instructions to a professional searcher. The topics may also contain instructions to the TREC judges of what constitutes a relevant document. This information is contained in fields, all or some of which can be used to construct the query that is actually issued to a collection. As a result, the formulation of actual queries used in published results can differ widely. Later in this section, we discuss the approaches that we employed when creating queries. We will refer to our formulations as *queries* while retaining the TREC terminology of *topic* to refer to the original statement of information need. We retain the TREC topic numbering for our queries. We will apply the relevance judgements for a topic to each query generated from that topic.

In many years, the TREC conference has introduced new document sets. In every year, new topic sets have been introduced, generally in batches of 50 topics per set. Because of the evolutionary nature of the conference, relevance judgements are not available for all combinations of topic and document sets. Through TREC-4, there were a total of 250 topics with relevance judgements over some portion of the TREC documents. In Table 4.2 we summarize this coverage.

The topic coverage (Table 4.2) is important because it constrains the possible combinations of topic sets and document collections that can be used in a multi-collection retrieval experiment where relevance judgements are needed. For example, if topics 201-250 are used

for evaluation, then any collections created using documents from disk 1 should not be used. Similarly, if a researcher wants to work with the San Jose Mercury News data (found on disk 3), then only three of the five topic sets are applicable. Given these constraints, we will use only topics 51-150 in our experiments. This maximizes the number of documents available for collection creation.

In our main body of experiments, we used two query formulation strategies, producing two sets of 100 queries each. We will refer to these formulations as "short" and "long". The short queries, $\mathcal{Q}_s$, were constructed using the Title field of the TREC topics. These queries average 3.5 words per query and are a very brief description of the information need. The long queries, $\mathcal{Q}_l$, were constructed using the Concepts field of the TREC topics and average 21 words per query. The Concepts field contains words, phrases and especially proper names that might be found in relevant documents. The terms found in the Concepts field have the potential to resemble very well-thought-out user queries; however, our resulting long queries contained more terms than queries typically received from real users. For example, Spink and Saracevic [SS97] found that on average experienced searchers used approximately 15 terms per query, of which approximately 6 terms were taken from inital statements of user information need (the remaining terms were added during user interaction, using a thesaurus, etc.). For WWW queries, query length is even shorter. In analyses of different Web search engine query logs, both Jansen *et al.* [JSBS98] and Silverstein *et al.* [SHMM99] found an average query length of 2.35 terms. Abdulla *et al.* [ALSF97] found that queries from a variety of query logs rarely exceeded 4-5 terms. We chose to use both short and long queries to account for the cases of longer, more detailed queries while also considering the shorter queries typically found in operational environments.

In some early experiments, we used an even longer query formulation, which we will refer to as "very long" or $\mathcal{Q}_{vl}$. These queries use all of the text available in the TREC topics and averaged 49 terms per query. We later determined that the short and long approaches were more commonly used in other research, so switched to those approaches. Only the early experiments reported in Chapter 6 use the very long queries. As a result, plots from

the figures in Chapter 6 are not directly comparable to figures in Chapters 7–9.

4.3 Goals and Requirements for the Testbeds

We began our testbed-construction efforts with the goal of constructing a single multi-collection test environment. Given the available TREC-4 data, the main problems to address were how much of the data to use and how to partition the data into collections. We started with a number of requirements and goals:

- **A natural partition.** Earlier experiments involving parameter driven creation of document collections ([VF95b]) were illuminating, but the partitions themselves did not reflect any physical (i.e. time or source) attribute of the source data. In addition, the TREC data is often referenced in terms of *source : disk number*, and has often been subdivided using one or both of those attributes [FKS+92, WFPS94, CLC95, Voo96]. To the extent possible, a candidate partition should not obscure these other, more coarse grained possibilities.

- **At least 100 collections.** We felt realistic experiments must involve at least 100 document collections.

- **A temporal dimension.** Date and source of publication are simple criteria by which to organize a collection of documents. We wanted to study such an organization and included this in our requirements. However, while we considered time of publication for testbed creation, we do not study temporal issues in the experiments reported in this dissertation. This requirement affected the *SYM-236* testbed (the first we created) and indirectly affected the *UDC-236* testbed (both testbeds are described in more detail later).

- **Easy composition of "supercollections" from components.** As much as possible, we wanted to create a partition of the data from which easily–identifiable compositions could be created. For example, disk 3 contains documents published on

the AP newswire in 1990. If we subdivide those documents into collections based on month of publication, reconstructing a supercollection based on year of publication will be simple. This requirement affects the *SYM-236* testbed directly and *UDC-236* indirectly.

These effects of these goals can be seen primarily in the *SYM-236* testbed, the testbed we initially set out to create. Residual effects can be seen in the *UDC-236* testbed. The *UBC-100* testbed was not constructed by us and was constructed using a different set of criteria.

4.4 The *SYM-236*, *UDC-236* and *UBC-100* Testbeds

In this section, we will describe the *SYM-236*, *UDC-236* and *UBC-100* testbeds. The *SYM-236* testbed will be covered in more detail partly because its construction was heavily influenced by the goals outlined above and a number of compromises were necessary to meet those goals. *SYM-236* also has more unusual features than the other two testbeds.

Each experimental testbed is a set of N collections, $\mathcal{C} = \{C_1, C_2, \ldots C_N\}$. For the *SYM-236*, *UDC-236* and *UBC-100* testbeds, $\mathcal{C}$ is a partition of $\mathcal{D}$.

A testbed can be represented as data items, $\mathcal{D}$, plus a data item to collection map, $R_{\mathcal{D}\to\mathcal{C}}$, from which the set of collections, $\mathcal{C}$, can be constructed. The data item to collection map may be externally supplied, or constructed using some rule designed to create a testbed with some desired characteristic(s).

Each of the three testbeds described here are based upon 3 gigabytes of data available to participants in the TREC-4 [Har96] experiments.

4.4.1 *SYM-236* (Source-Year-Month)

When constructing the *SYM-236* testbed, we were working under the goals and requirements that we set forth above. As a result, there were some special cases in the selection of the documents in $\mathcal{D}$ and in the creation of the mapping relation of documents to collections,

$R_{\mathcal{D}\to\mathcal{C}}$. The general rule for creating $R_{\mathcal{D}\to\mathcal{C}}$ was to partition the documents on TREC CDs 1, 2 and 3 by publishing source, then by year and month of the publication date.

The first exceptions and special cases dealt with determining which documents from TREC CDs 1, 2 and 3 would be included in $\mathcal{D}$.

- **No DOE documents.** Because we were using publication date to create $R_{\mathcal{D}\to\mathcal{C}}$, no documents from the Department of Energy publishing source (DOE) were included in $\mathcal{D}$. The DOE data is undated.

- **No ZIFF documents from TREC disk 3.** Some of the data from the ZIFF publishing source on disk 3 overlaps temporally with data from ZIFF on disks 1 and 2. We considered placing these documents with the others from the same month, but this would have involved considerable intermingling of disk 3 documents with those from disks 1 and 2. While certainly possible, it would have violated the composability requirement given previously. As a result, ZIFF documents from disk 3 were not included in $\mathcal{D}$.

As we mentioned, the general rule for creating the mapping relation of documents to collections, $R_{\mathcal{D}\to\mathcal{C}}$, was to partition the documents on TREC CDs 1, 2 and 3 by publishing source, then by year and month of the publication date. Our next set of special cases arose when reconciling features of the documents in $\mathcal{D}$ with that rule.

- **Disambiguation of dates in ZIFF.** A small number of the documents from the ZIFF publishing source are dated as "Summer", "Winter", etc. rather than by month. In reconciling these dates, we were aided by the chronological nature of the ordering of documents on the TREC CDs. In these cases, we determined the date of the documents by looking at the dates of documents surrounding the document in question. Thus if a "Spring" document was immediately preceded by one dated "March", then we assigned the "Spring" document to the "March" collection for that source and year.

- **Disambiguation of multiple dates in PAT.** The structure of the Patent Office (PAT) documents is complex and often contains references to previously issued patents. To disambiguate, we used the first appearance of the "Application Filing Date" (AFD) field within a document as the operative date. Subsequent occurrences of this field within a document refer to other patent documents.

The net result of combining the particular attributes of the TREC data and our own requirements was a partition comprised of 236 document collections derived from some but not all of TREC disks 1, 2, and 3. Summary characteristics of this partition are given in Table 4.3.

The temporal nature of our collection construction approach led to some interesting features of the resulting *SYM-236* testbed.

- **Vast size variation.** One of the most notable features of the *SYM-236* testbed is the vast size variation (in terms of documents per collection) of the collections. There are a number of very large collections with more than 8,000 documents per collection plus several dozen very small collections 1 to 20 documents in size. The very small collections are mainly derived from early PAT data. The size variation of the *SYM-236* testbed will prove important for identifying a feature of some collection selection algorithms.

- **Temporal discontinuities.** Collections were constructed using a fixed date increment of one month. However, many sources only contain partial years, so the data is not continuous temporally. This is an attribute of the underlying TREC corpus. Coverage of the 236 document collections by source and date is given in Figure 4.1. Researchers who are interested in examining collection changes over time must be judicious in the subset of collections they use as the basis for their work. For example, there are several multi-month "holes" in the Wall Street Journal collections.

Disk	Source	Num. Coll.	Date Range	Total Coll.
1	WSJ (86-89)	29	12/86–11/89	67
	AP (89)	12	01/89–12/89	
	ZIFF	14	11/89–12/90	
	FR (89)	12	01/89–12/89	
	DOE	XX	XX	
2	WSJ (90-92)	22	04/90–03/92	54
	AP (88)	11	02/88–12/88	
	ZIFF	11	01/89–11/89	
	FR (88)	10	01/88–12/88	
3	AP (90)	12	01/90–12/90	116
	SJMN (91)	12	01/91–12/91	
	ZIFF	XX	XX	
	PAT	92	06/82–08/92	

Table 4.3: Summary characteristics of the document partition. Note: Total DB column sums to 237 because there is overlap of one collection in ZIFF between disks 1 and 2 (ZIFF.89.11).

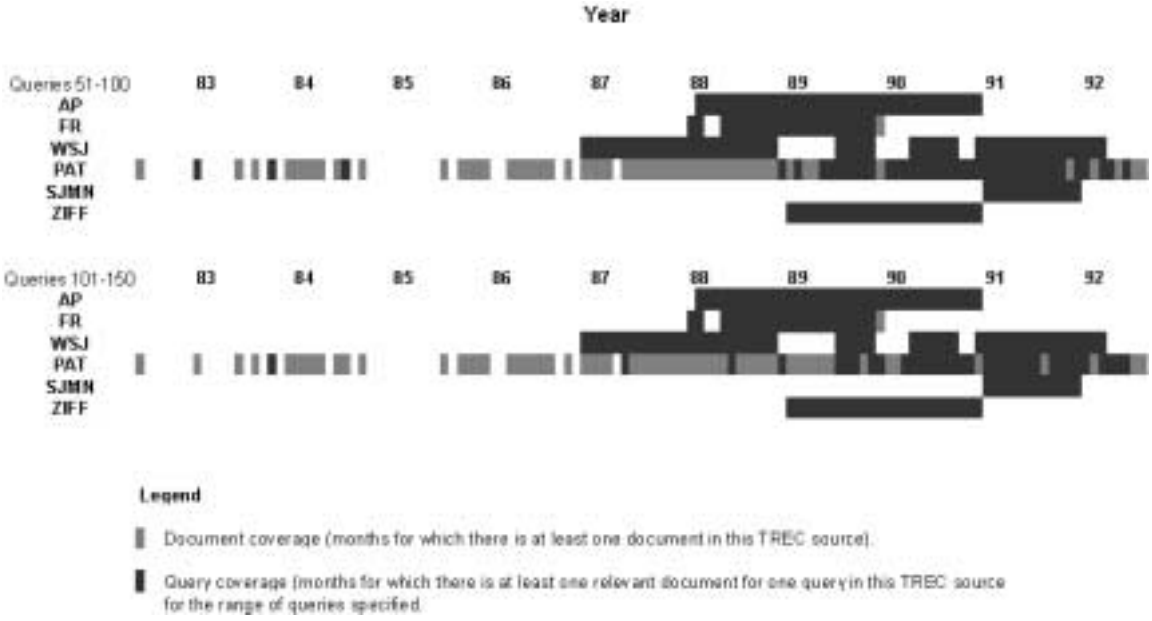


Figure 4.1: Document and query coverage for the *SYM-236* testbed (236 collections, partitioned by original source and date).

4.4.1.1 A Brief Summary of *SYM-236*

The *SYM-236* (Source-Year-Month) testbed was designed to contain a temporal component. Documents were organized into document collections based on the primary source and the month and year of publication. For example, all AP Newswire articles from February of 1988 were placed in the same collection.

$\mathcal{D}$ - Data items are text documents from TREC CDs 1, 2, and 3 minus DOE documents (documents contain no date) and ZIFF documents from disk 3 (to maintain composability requirement).

$R_{\mathcal{D}\rightarrow\mathcal{C}}$ can be found at `http://www.cs.virginia.edu/~cyberia/testbed.html` labelled trec123-236-by_source-by_month.

$|\mathcal{D}| = 691,058$ documents in the testbed subdivided into

$N = 236$ collections.

4.4.2 *UDC-236* (Uniform-Document-Count)

At the time that *SYM-236* was created, an equally viable, alternative partitioning strategy would have split the data into N equal sized collections. This partitioning approach has attractive characteristics in that 1) it is easy to control the number of collections and 2) one confounding variable, collection size, is held constant. We chose not to pursue this strategy at the time because of our interest in exploring a document organization based upon publication date and source.

However, as we noted during the description of *SYM-236*, the vast size variation of *SYM-236* proved interesting during the evaluation of collection selection algorithms that will be discussed in Chapters 7 and 8. This prompted the creation of the *UDC-236* testbed. The *UDC-236* testbed contains exactly the same documents as *SYM-236*; however, the documents were organized into collections containing roughly 2,900 documents each, ordered as they appeared on the TREC CDs, and with the restriction that all of the documents in

a collection were from the same primary source. This testbed also contains 236 collections.

4.4.2.1 A Brief Summary of *UDC-236*

The *UDC-236* (Uniform-Document-Count) testbed was designed to control for the tendency of some collection selection algorithms to prefer collections with a large number of documents. Collections for *UDC-236* contain roughly 2,900 documents each.

$\mathcal{D}$ - Data items are exactly the same as those for *SYM-236*.

$R_{\mathcal{D}\to\mathcal{C}}$ can be found at `http://www.cs.virginia.edu/~cyberia/testbed.html` labelled trec123-236-eq_doc_counts.

$|\mathcal{D}| = 691,058$ documents in the testbed subdivided into

$N = 236$ collections.

4.4.3 *UBC-100* (Uniform-Byte-Count)

The *UBC-100* testbed was constructed at the University of Massachusetts and was not influenced by the goals and requirements that we set forth earlier in this chapter. All of the documents from TREC CDs 1, 2 and 3 were included in this testbed. Data items were organized into collections of roughly 30 megabytes each, ordered as they appeared on the TREC CDs, and with the restriction that all of the data items in a collection were from the same primary source. This testbed was added during the course of our collaboration with researchers from the University of Massachusetts and Carnegie Mellon University.

$\mathcal{D}$ - Data items are text documents from TREC CDs 1, 2, and 3.

$R_{\mathcal{D}\to\mathcal{C}}$ can be found at `http://www.cs.cmu.edu/~callan/Data/` labelled trec123-100-bysource-callan99.v2a.

$|\mathcal{D}| = 1,078,166$ documents in the testbed subdivided into

$N = 100$ collections.

4.4.4 A Summary of the Testbeds

SYM-236 and *UDC-236* have been used in evaluations of database selection algorithms [FPV+98, FPC+99b]. *UBC-100* was used to study the scalability of *CORI* collection selection [FPC+99b] and the effect of sampled language models on collection selection[CPFC00].

General characteristics of the testbeds appear in Table 4.4. This table shows both features of the testbeds and the effects of particular constraints in testbed creation. The *UBC-100* and *UDC-236* testbeds are constructed to contain collections of approximately 30 MB and collections of approximately 2,900 documents[2], respectively. Depending on individual document size, fixing one of these values can still result in variability in the other. Because there was a temporal component, there was more variability in the sizes of the *SYM-236* collections. For example, there were generally few Patent Office documents in a given month, but there were often many articles from the AP Newswire.

These three testbeds represent three convenient ways to organize documents into collections or to partition a large collection into several smaller ones. Xu and Croft [XC99, p. 256] expressed concern that the distribution of relevant documents in sets of collections such as these may adversely affect the efficiency or effectiveness of multi-collection retrieval. We discuss this issue in Chapter 9. We also summarize the distribution of relevant documents in the *UBC-100*, *SYM-236*, and *UDC-236* testbeds in Figures 4.3–4.5.

Figure 4.2 provides a visual illustration of the distribution of documents in the *SYM-236*, *UDC-236* and *UBC-100* testbeds. There are a number of main features to note:

- The distribution of documents in the *SYM-236* testbed is very skewed. Not only are there a large number of PAT collections with very few documents, the AP and SJM collections tend to have twice as many documents as the FR, WSJ, and some of the ZIFF collections.

- While the same documents are used, the distribution of collections per publishing source is very different in *UDC-236* than in *SYM-236*. There are only two PAT

[2]While creating *UDC-236* we did not mix documents from different publishing sources in the same collection. As a result, there are small differences in collection size.

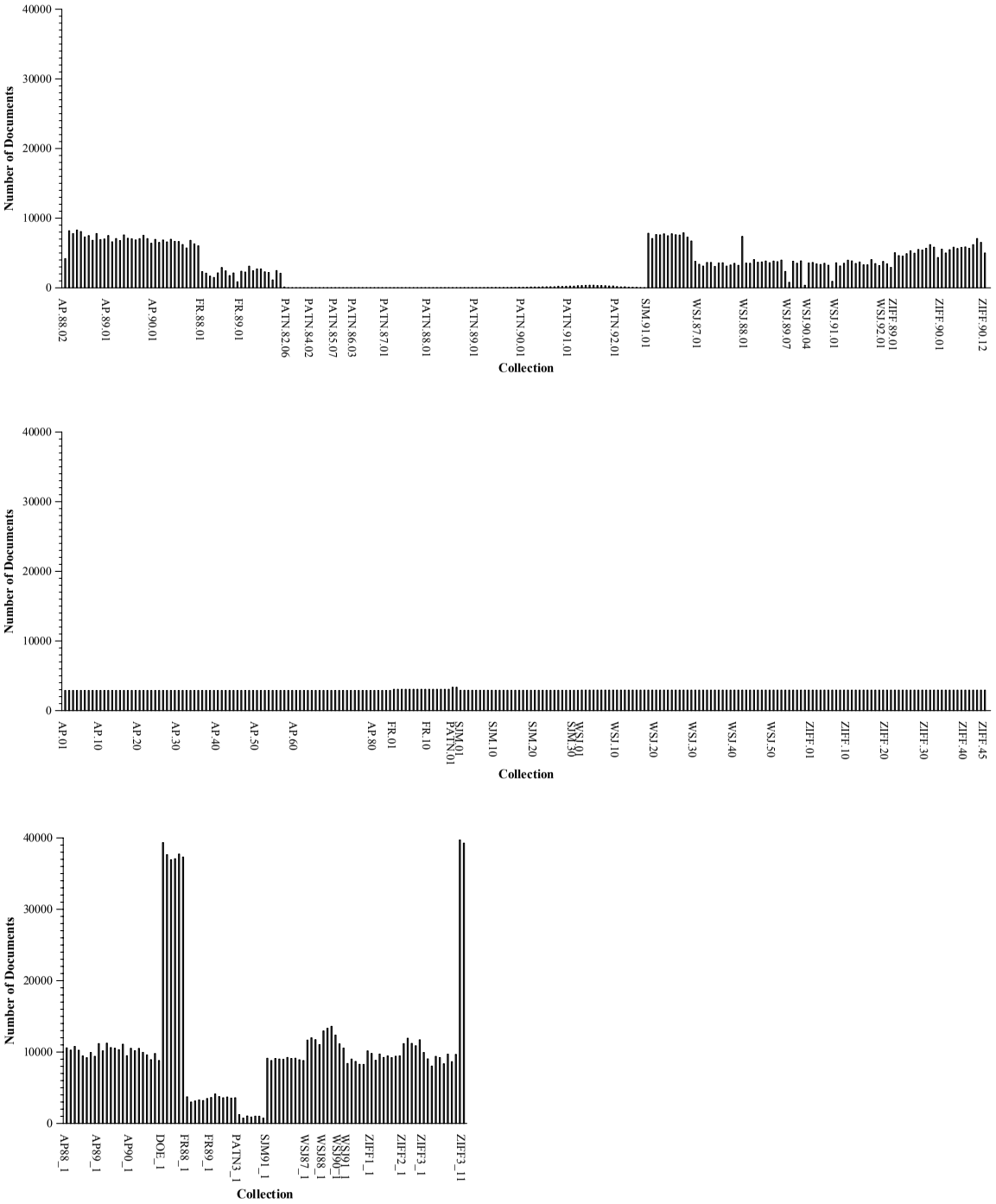


Figure 4.2: The distribution of documents in *SYM-236*, *UDC-236* and *UBC-100*.

Testbed	$\|\mathcal{D}\|$	N	Data Items per coll.			Bytes per collection		
			Min.	Avg.	Max.	Min.	Avg.	Max.
UBC-100	1,078,166	100	752	10,782	39,723	28,070,646	33,365,514	41,796,822
SYM-236	691,058	236	1	2,928	8,302	7,668	11,789,423	34,782,134
UDC-236	691,058	236	2,891	2,928	3,356	7,138,629	11,789,423	133,206,035

Table 4.4: Summary statistics for the testbeds.

collections and over fifty each of AP and WSJ collections.

- The illustrations re-emphasize that there are only 100 *UBC-100* collections as opposed to 236 *SYM-236* and *UDC-236* collections.

- The distribution of documents in the *UBC-100* testbed is skewed, but in a different way from *SYM-236*. There are a few small PAT collections, but the most striking feature are the six DOE collections and two ZIFF collections that contain a very large number of very small documents.

4.5 Document and Relevant Document Distributions

The three graphs of Figure 4.2 are repeated in Figures 4.3–4.5. They are accompanied by graphs that show the distributions of relevant documents in the collections. Figures 4.3–4.5 should be viewed sideways and the two graphs of each figure are aligned so that points and bars for each collection line up vertically. In the lower graph for each figure, we show the number of queries for which each collection contains at least one relevant document. Then, for each of those queries, we plot the mean number of relevant documents in the collection along with error bars. Taken together, these values provide a rough characterization of the distribution of relevant documents in the collections. Some observations follow:

- For all three testbeds, we find that the PAT collections contain very few relevant documents for relatively few queries.

- We also note the same feature observed by Voorhees *et al.* [VGJL95], many relevant documents are found in AP or WSJ collections. For most queries, picking AP or WSJ collections is a reasonable heuristic. We also note that SJM collections qualify for this observation.

- A more specific observation is that AP newswire articles published in 1990 tend to have a noticeably higher average number of relevant documents per query.

- Despite the very large number of documents per collection, the DOE collections of the *UBC-100* testbed tend to contain relevant documents for only a few queries. However, for those queries, the number of relevant documents in the DOE collections tends to be large but variable.

- The number of relevant documents in ZIFF collections tends to vary widely on a query-by-query basis.

We will consider the implications of these distributions of documents and relevant documents when we evaluate collection selection approaches in Chapters 7 and 8.

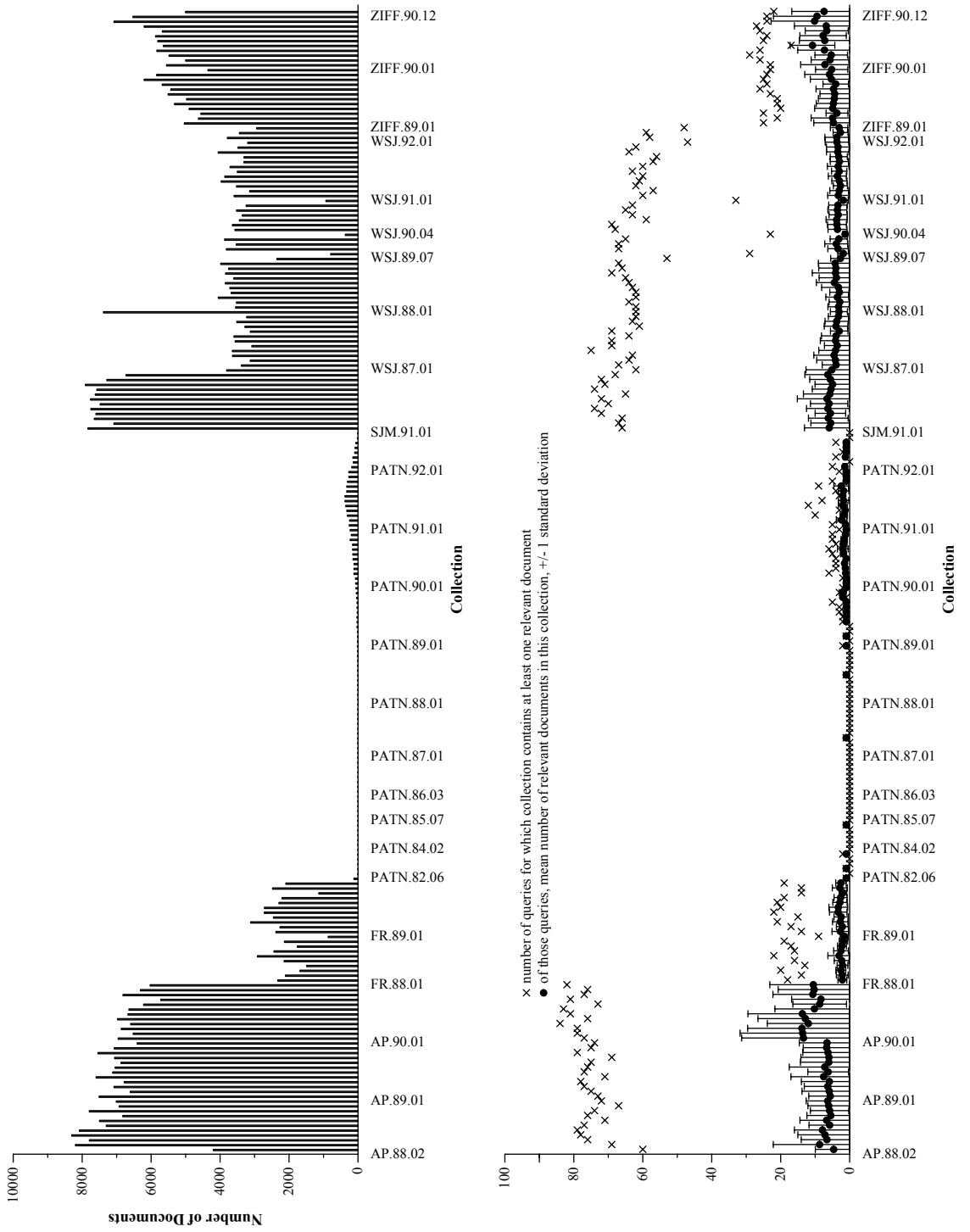


Figure 4.3: The distribution of documents and the number of queries for which a collection contains relevant documents in *SYM-236*.

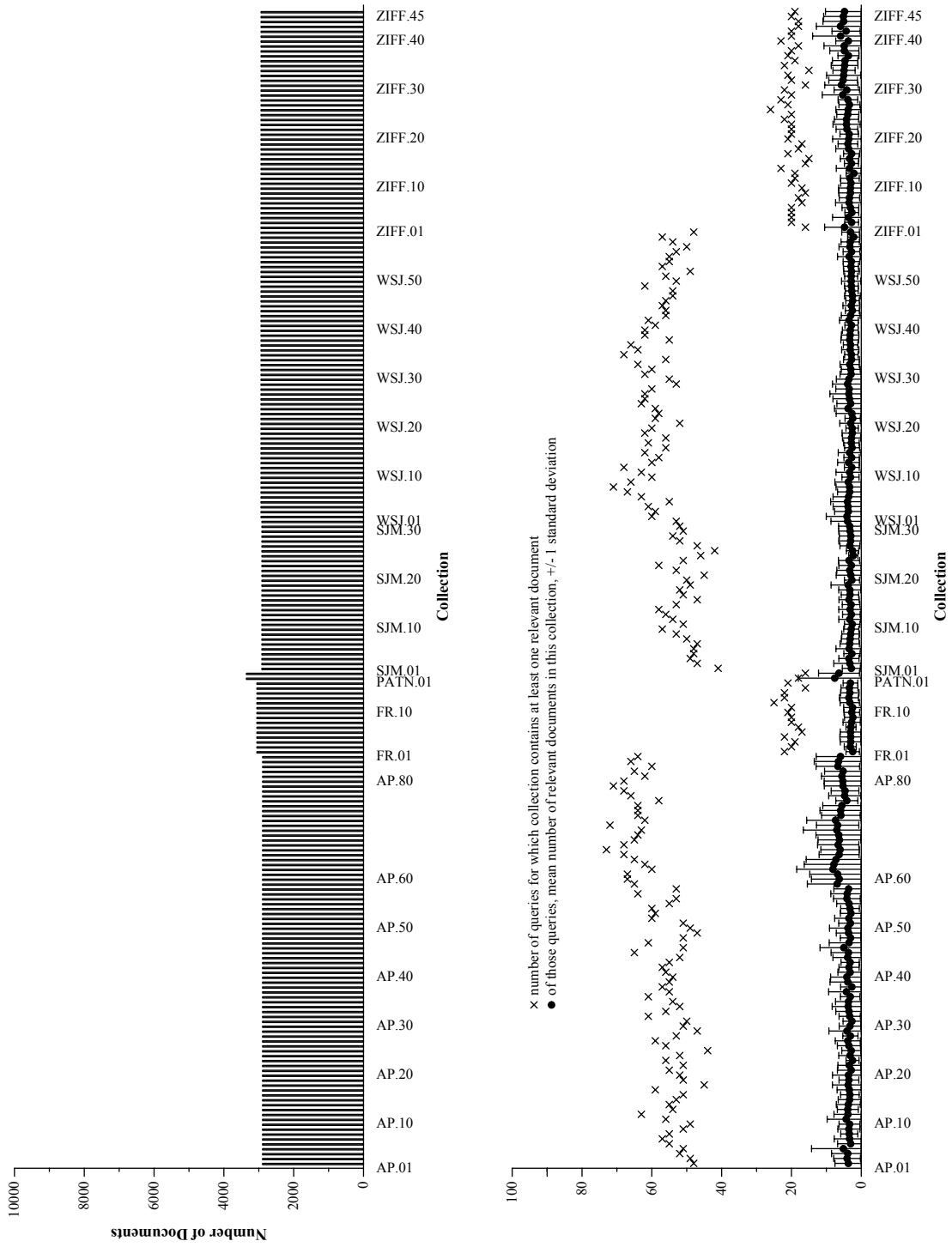


Figure 4.4: The distribution of documents and the number of queries for which a collection contains relevant documents in *UDC-236*.

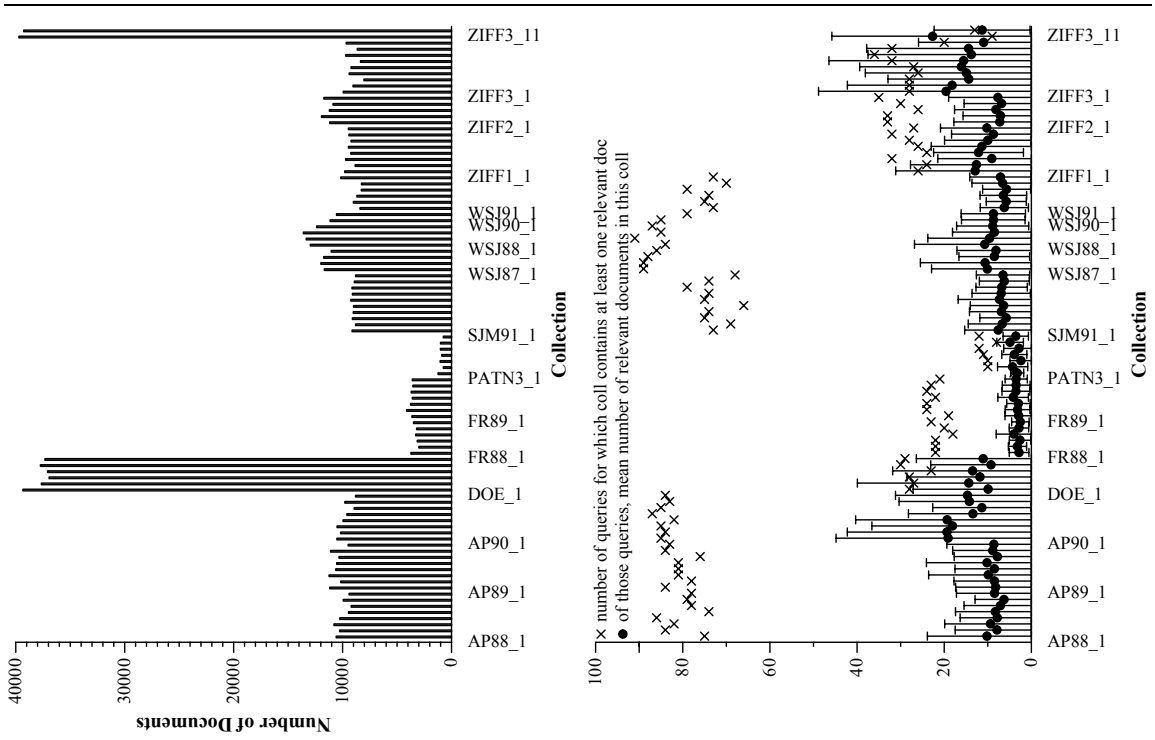


Figure 4.5: The distribution of documents and the number of queries for which a collection contains relevant documents in *UBC-100*.

5

Evaluation Measures and their Properties

Two major classes of evaluation are performed over the course of the experiments reported in this dissertation: the evaluation of collection selection approaches found in Chapters 6–8 and the evaluation of document retrieval performance found in Chapters 9. Different evaluation measures are used for these two classes of evaluation; we will define and discuss both sets of measures here.

In a multi-collection retrieval environment, collection selection and document retrieval are often performed sequentially. The effects of collection selection performance can often be seen in document retrieval performance. To give a very simplistic example, if no collections containing relevant documents are selected, final relevance-based document retrieval performance using those selected collections will obviously be very poor. However, while collection selection performance and document retrieval performance are often related, it is necessary to measure the performance of both steps individually. Excellent collection selection is ineffective if a poor document retrieval approach is used at a selected collection. A good document retrieval approach at a collection does not affect performance if the collection is not selected. The use of multiple evaluation stages helps us to determine the cause of retrieval performance successes or failures.

The evaluation methodology for document retrieval performance is more well-established in the information retrieval field than the methodology for collection selection performance.

Therefore, we will present a more thorough discussion of issues arising during collection selection performance evaluation. We will start with a discussion of collection selection evaluation measures[1] then proceed to definitions of document retrieval evaluation measures.

5.1 Evaluating Collection Selection

As we mentioned earlier, the methodology for evaluating collection selection algorithms has not yet been standardized. However, at the heart of most proposed evaluation approaches is an attempt to determine to what extent an estimated ranking produced by a collection selection approach approximates a baseline ranking that represents some desired behavior.

We will begin by giving brief definitions of the baselines and estimators used in our evaluations. The baselines and estimators are covered in much more detail in the context of the experiments in which they are used (Chapters 6–9). We will give a overview of the evaluation approach then discuss general evaluation issues. We will define the specific measures used in these experiments, then discuss the expected performance of random collection selection under these measures. Finally, we will present some features of and relationships among the collection selection evaluation measures.

5.1.1 Baselines and Estimators

Baselines and estimators are both simply rankings of collections, the difference is one of interpretation. If a collection ranking is being used to represent some desired behavior then it is a baseline. If a collection ranking is being evaluated to determine if it exhibits the desired behavior then it is an estimator. Generally speaking, baselines are created using information about collections that is not readily available in an operational setting, while estimators employ summary statistical information about the collections to create their rankings. However, the same collection ranking may be used as both a baseline and

[1]This chapter represents an expansion of our previous investigation of collection selection evaluation measures [FP99, FP00].

an estimator. For example, the *Ideal(0)* ranking was originally defined as a baseline for the *gGlOSS* collection selection algorithm [GGM95]; however, for most experiments we use *Ideal(0)* as an estimator. Chapter 6 contains a full discussion of our use of *Ideal(0)*.

There are many possible baselines that could be used for performance evaluation. We discuss four here.

Count-Based Ranking (CBR): The baseline is constructed by ordering the collections in decreasing order of the number of documents contained in the collection that satisfy a Boolean predicate.

Ideal(l): In Gravano and García-Molina [GGM95] it was assumed that (a) all collections employ the same weighting strategies and similarity algorithm; and (b) the only documents in a collection that are useful to a query q are those with a similarity greater than some threshold l. With these assumptions in mind they define

$$Goodness(l, q, C) = \sum_{\{d \in C | sim(q,d) > l\}} sim(q, d)$$

The *Ideal(l)* rank is then formed by sorting the collections with respect to their goodness to q.

Relevance-Based Ranking (RBR): The baseline is constructed by ordering the collections in decreasing order of the number of relevant records contained in the collection.

Size-based Ranking (SBR): Collections are ordered by the total number of documents they contain. Note that this ranking is constant for all queries.

There are many other possible rankings. Those listed above are representative of rankings that have been reported in the literature. For example, CBR was used by Gravano *et al.* [GGMT94b] for evaluating *GlOSS*; *Ideal(l)* was used by Gravano and García-Molina [GGM95] to report their performance evaluation of *gGlOSS*; RBR was used by Callan *et al.* [CLC95] for their evaluation of *CORI* and by French *et al.* [FPV+98, FPC+99b, FPC99a]

in their investigations; SBR was used by French *et al.* [FPC+99b] to isolate and study a bias toward large collections that is exhibited by some collection selection algorithms

Note that when evaluating a set of queries $\mathcal{Q}$, we will generally have a separate baseline instance for each $q \in \mathcal{Q}$. SBR is one exception to this; it is query-independent and simply specifies the order in which collections should be visited to satisfy any query.

The purpose of our evaluation is to gauge the performance of a set of estimators. These estimators will include:

- collection selection algorithms proposed in the literature,
- modified versions of those published algorithms intended to isolate the impact of algorithm components on performance, and
- new collection selection algorithms based on very simple collection statistics.

The actual estimators evaluated are defined in Chapters 7 and 8. These estimators employ summary statistical information about a set of collections to produce a ranking of those collections. Given the ranking, we can specify a subset of collections to be searched or a collection search order.

Before defining the specific evaluation measures we employ, we will cover some theoretical issues that arise in performing comparisons and some properties of baselines and estimators that can affect evaluation.

5.1.2 General Evaluation Strategy

Given some goal baseline B and an algorithm producing an estimate, E, of that goal, we endeavor to determine the quality of the estimate by means of some measure $m(E, B)$ comparing the estimate to the goal. To make this discussion and later definitions of evaluation measures more concrete, we present an example of baseline and estimated rankings in Figure 5.1. For six collections, $\mathcal{C} = \{A, B, C, D, E, F\}$ and three queries $\mathcal{Q} = \{q_1, q_2, q_3\}$, we show the merit in terms of the baseline, say RBR, and as estimated by some hypothetical estimator. Collections are sorted using those merit values to create the rankings shown.

Baseline Merits

	A	B	C	D	E	F
q_1	6	2	9	5	1	7
q_2	4	18	3	9	5	1
q_3	2	1	2	0	4	0

Estimated Merits

	A	B	C	D	E	F
q_1	0.7	0.4	0.6	0.2	0.1	0.5
q_2	0.2	0.8	0.4	0.7	0.9	0.1
q_3	0.2	0.0	0.3	0.4	0.5	0.1

Baseline Rankings

	1	2	3	4	5	6
q_1	C	F	A	D	B	E
q_2	B	D	E	A	C	F
q_3	E	A	C	B	D	F

Estimated Rankings

	1	2	3	4	5	6
q_1	A	C	F	B	D	E
q_2	E	B	D	C	A	F
q_3	E	D	C	A	F	*B*

Figure 5.1: An example baseline and estimate. Rankings are created using the corresponding merits.

Note that for none of the queries do the estimated rankings exactly match the baseline rankings. A given measure $m(E, B)$ is used to determine the degree and impact of the difference in the rankings.

Most evaluations are conducted over a query set $\mathcal{Q}$. The performance of an estimator with respect to a baseline is evaluated for each query. Overall results are usually presented as aggregate summary measures of the following form:

$$\frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} m(E_q, B_q).$$

Note that $m(E_q, B_q)$ might itself already be an aggregate measure, for example, mean squared error discussed below.

5.1.3 Issues in Comparison

Recall that given a baseline ranking B and estimated ranking E, we will employ a measure $m(E, B)$ to compare the estimate to the baseline.

Many of our evaluations will be of the form

$$m(E_1, B) \text{ vs. } m(E_2, B)$$

In this scenario, we are performing a direct comparison of two estimators using the same baseline and evaluation measure. These results are often presented as different plots on the same graph.

Since $m(E, B)$ is a comparison of E to B, we might also like to evaluate a particular algorithm in a variety of ways. In that case E is constant but B or $m(E, B)$ might change. This gives rise to three situations that must be considered.

1. Multiple measures, single baseline:

$$m_1(E, B) \text{ vs. } m_2(E, B).$$

   This is probably the most common situation. Here we are evaluating the estimate against the baseline using multiple yardsticks. This is often useful to expose different aspects of the phenomena under study.

2. Single measure, multiple baselines:

$$m(E, B_1) \text{ vs. } m(E, B_2).$$

   One example of this situation is when the baseline is parameterized and varying the baseline parameters while holding the measure constant is the technique being used.

3. Multiple measures, multiple baselines:

$$m_1(E, B_1) \text{ vs. } m_2(E, B_2).$$

   This situation occurs when measuring against multiple baselines and no single measure is appropriate for all baselines.

No standardization for comparisons has emerged. Evaluations reported in the literature are quite idiosyncratic, compounding the problem of comparing methods for collection selection.

5.1.4 Properties of Measures

It is often useful in an evaluation to understand the properties of the measures being employed. At the very least we want to know:

1. the value of the measure when the baseline is applied to itself, $m(B, B)$;
2. the value of the measure when the worst ranking using the merit values employed by the baseline, say $\bar{B}$, is applied to the baseline, i.e., $m(\bar{B}, B)$; and
3. whether the measure is symmetric, $m(X, Y) = m(Y, X)$.

Other properties might be important as well. To be useful a ranking measure should also have the following property:

$$m(\bar{B}, B) \leq m(E, B) \leq m(B, B).$$

$m(\bar{B}, B)$ is the operational lower bound on performance. If a ranking measure $m(X, Y) \geq 0$ and satisfies this property then it can be normalized to be in $[0, 1]$.

5.1.5 Properties of Rankings

There are two situations under which we would say that two rankings E_1 and E_2 are *equivalent*. The two scenarios are:

1. when there are equivalence classes of collections in the rankings, i.e., one or more collections have identical merit, and the only difference in the rankings is found in the ordering of items in the equivalence classes; and
2. when two rankings have the same value under the measure, i.e., for some query and measure $m(\cdot)$, $m(E_1, B) = m(E_2, B)$.

In principle these two situations might have different implications. In practice, equivalence (1) above should imply (2), but the converse need not hold.

Note that an extreme example of (1) above is the case where all collections have equal merit. In that case, every permutation of the collections is an equivalent ranking.

We need to be concerned about this issue when we compare against a baseline that has equivalent rankings. It is important that all equivalent rankings be reduced to some canonical form so that valid comparisons can be made. For example, consider the example in Figure 5.1. For query q_3, collections A and C have the same merit under the baseline. When the collections are sorted to produce the ranking, the tiebreaker was placing collections in alphabetical order. However, this is arbitrary. Because collections A and C have equivalent merit, the estimated ranking should not be penalized during evaluation for transposing the order.

5.1.6 Specific Measures for Comparison

There is no general agreement on how this type of comparison should be done. The general problem is that we are given a baseline ranking for some query and a ranking produced by some collection selection algorithm. The goal is to decide how well the candidate ranking approximates the baseline ranking and to reveal potential performance implications of the quality of the approximation. We describe some of the approaches given in the literature and discuss new measures here.

5.1.6.1 Mean Squared Error

Callan *et al.* [CLC95] reported their comparisons using mean squared error of the predicted ranks and the desired ranks. So given a set of N collections, $\mathcal{C}$, to rank for any candidate ranking we compute

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (base\_rank(C_i) - est\_rank(C_i))^2$$

where $base\_rank(C_i)$ is the baseline or desired rank and $est\_rank(C_i)$ is the predicted rank for C_i.

Mean squared error is a widely used measure of dispersion but does not give us any real intuition about ranking quality. We might benefit here from a normalization step based on comparison with worst case behavior. For any baseline B, the worst case configuration, $\bar{B}$, is simply the reverse ranking. We can calculate the sum of squared differences as follows.

$$\begin{aligned}(N-1)^2 + (N-3)^2 &+ (N-5)^2 + \cdots \\ \cdots &+ (3-N)^2 + (1-N)^2 \\ &= \sum_{i=1}^{N} [N-(2i-1)]^2 \\ &= \frac{N(N^2-1)}{3} \end{aligned} \tag{5.1}$$

From Eq. 5.1 we see that the maximum MSE is given by

$$MSE_{max} = \frac{N^2-1}{3}. \tag{5.2}$$

5.1.6.2 Spearman Coefficient of Rank Correlation

The Spearman coefficient of rank correlation [Gib76], ρ, is given by

$$\rho = 1 - \frac{6\sum_{i=1}^{N} D_i^2}{N(N^2-1)} \tag{5.3}$$

where D_i is the difference in the i-th paired ranks. We have $-1 \leq \rho \leq 1$ where $\rho = 1$ when two rankings are in perfect agreement and $\rho = -1$ when they are in perfect disagreement. Note that the computational formula for Spearman's ρ is much more involved when ties

are allowed. The formulation corrected for ties [Gib76] is given by

$$\rho = \frac{N(N^2-1) - 6\left(\sum_{i=1}^{N} D_i^2\right) - 6(u'+v')}{\left(\sqrt{N(N^2-1)-12u'}\right)\left(\sqrt{N(N^2-1)-12v'}\right)} \tag{5.4}$$

where $u' = \left(\sum u^3 - \sum u\right)/12$ and $v' = \left(\sum v^3 - \sum v\right)/12$. u is the number of observations in the first sample that are tied at a given rank; v is the corresponding value for the second sample. The sum is over all sets of u (or v) tied ranks in the sample.

In our work we use the mid-rank[2] method for assigning ranks when ties are present and we use the Spearman calculation corrected for ties.

5.1.6.3 Recall and Precision Analogs

In this section we discuss performance metrics that are analogous to the well known IR metrics of *recall* and *precision*. We begin by introducing some terminology and notation that tries to make this analysis neutral and generalizes it to include a variety of baselines.

Recall that for each query we provide a *baseline ranking* that represents a desired goal or query plan. Given some algorithm that produces an *estimated ranking*, our goal is to decide how well the estimated ranking approximates the baseline ranking.

To begin, we assume that each collection C in $\mathcal{C}$ has some merit, $merit(q, C)$, to a given query q. We expect the baseline to be expressed in terms of this merit; we expect the estimated ranking to be formulated by implicitly or explicitly estimating merit.

Let C_{b_i} and C_{e_i} denote the collection in the i-th ranked position of the baseline and estimated rankings respectively. Next we define two sequences, B and E, based on the merit of the collections in the two rankings. Let

$$B_i = merit\left(q, C_{b_i}\right) \text{ and } E_i = merit\left(q, C_{e_i}\right)$$

denote the merit associated with the i-th ranked collection in the baseline and estimated

[2]The mid-rank is simply the mean of the ranks of tied observations. For example, given four collections with merits 8, 6, 6, 3, the midranks of those four collections would be 1, 2.5, 2.5, 4 respectively.

rankings respectively. The total merit, M, is given by $M = \sum_{i=1}^{N} B_i$

We note that for viable baseline rankings it should always be the case that

$$B_i \geq B_{i+1},\ i = 1...N-1.$$

For the baselines discussed here this is always true because we assume that the baseline ranking is determined by sorting the collections in decreasing order of merit for some appropriate definition of merit. However, it is not generally the case that $E_i \geq E_{i+1}$. The performance evaluation problem discussed here is an attempt to quantify the degree to which this is true for any estimated ranking.

This point needs a bit of explanation. The estimators will rank collections in decreasing order of *estimated* merit (as calculated by the estimator). However, note that E_i is the *actual* merit associated with C_{e_i}, i.e. the merit used to create the baseline ranking. The degree to which $E_i \geq E_{i+1}$ reflects the accuracy of the algorithm's estimates of E_i.

Gravano and García-Molina [GGM95] defined $\mathcal{R}_n$ as follows.

$$\mathcal{R}_n(E, B) = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n} B_i}. \tag{5.5}$$

$\mathcal{R}_n(E, B)$ is a measure of how much of the available merit in the top n ranked collections of the baseline has been accumulated via the top n collections in the estimated ranking. This is a variant of the *normalized cumulative recall* measure defined by Tomasic *et al.* [TGL+97] and later generalized by Gravano *et al.* [GGMT94a].

We propose an alternate definition of a recall-like measure that can be used to present performance results. First we need one more definition. Let

$$n^* = k \text{ such that } B_k \neq 0 \text{ and } B_{k+1} = 0.$$

Intuitively, n^* is the ordinal position in the ranking of the last collection with non-zero merit; it is the breakpoint between the useful and useless collections. Clearly $n^* \leq N$ and,

moreover, the total merit, M, of a baseline is given by $M = \sum_{i=1}^{n^*} B_i$. With this definition we define our alternative recall metric as follows.

$$\widehat{\mathcal{R}}_n(E,B) = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n^*} B_i} = \frac{\sum_{i=1}^{n} E_i}{M} \tag{5.6}$$

The denominator is just the total merit contributed by all the collections that are useful to the query. Thus, $\widehat{\mathcal{R}}_n(E,B)$ is a measure of how much of the total merit has been accumulated via the top n collections in the estimated ranking. This measure has also been proposed by Lu *et al.* [LCC96] and was used to report results by French *et al.* [FPV+98, FPC+99b, FPC99a].

These two measures are clearly related. Since

$$\mathcal{R}_n(E,B)\sum_{i=1}^{n} B_i = \widehat{\mathcal{R}}_n(E,B)\sum_{i=1}^{n^*} B_i, \tag{5.7}$$

we have $\mathcal{R}_n(E,B) \geq \widehat{\mathcal{R}}_n(E,B)$ and $\mathcal{R}_{n^*}(E,B) = \widehat{\mathcal{R}}_{n^*}(E,B)$. (Note that we generally simplify the notation as follows: $\mathcal{R}_{n^*}(E,B) = \mathcal{R}_*$ and similarly for $\mathcal{P}_*$.)

We formalize the relationship between $\widehat{\mathcal{R}}_n(E,B)$ and $\mathcal{R}_n(E,B)$ in Theorem 1 below.

Theorem 1 $\widehat{\mathcal{R}}_n(E,B) = \mathcal{R}_n(E,B) \cdot \widehat{\mathcal{R}}_n(B,B)$

Proof: Follows directly from Equation 5.7.

From the theorem we can see that another way to interpret $\mathcal{R}_n(E,B)$ is to regard it as the rate at which the available baseline merit is being accumulated.

Gravano and García-Molina [GGM95] have also proposed a precision-related measure, $\mathcal{P}_n(E,B)$. It is defined as follows.

$$\mathcal{P}_n(E,B) = \frac{|\{C_i \in Top_n(E) | merit(q,C_i) > 0\}|}{|Top_n(E)|} = \frac{|\{C_i \in Top_n(E) | merit(q,C_i) > 0\}|}{n} \tag{5.8}$$

This gives the fraction of the top n collections in the estimated ranking that have non-zero merit. $Top_n(E)$ is just the set of collections given in the first n ranks. An alternative

interpretation for $\mathcal{P}_n(E, B)$ is that it measures the degree to which collections with zero merit have been interleaved with those having nonzero merit.

Some additional properties follow.

1. $\mathcal{R}_n(E, B) \leq \mathcal{R}_n(B, B)$ and $\mathcal{P}_n(E, B) \leq \mathcal{P}_n(B, B)$
2. $\mathcal{R}_n(B, B) = 1$ and $\mathcal{P}_n(B, B) = 1$
3. $\widehat{\mathcal{R}}_n(B, B) \leq 1$

In the remainder of the dissertation we simplify the notation by dropping all arguments to the measures when it is clear that we are referring to a specific algorithm's estimates (E) and measuring against a prespecified baseline (B).

To illustrate these recall and precision-based evaluation measures, consider the example shown in Figure 5.2. In this example, there are six collections, $\mathcal{C} = \{A, B, C, D, E, F\}$ and three queries $\mathcal{Q} = \{q_1, q_2, q_3\}$. This example is an extension of Figure 5.1 and Figure 5.1 is included as the upper portion of Figure 5.2. For each query we wish to visit the collections in order of the number of relevant documents for that query. Therefore, we will use the RBR baseline for evaluation and the merit of a collection will be the number of relevant documents for a query.

Assume that for queries q_1, q_2 and q_3 there are 30, 40 and 9 relevant documents respectively, distributed as shown in the Merits–Baseline table in the upper left of Figure 5.2. Sorting the collections using merit produces the Rankings–Baseline table. Next assume that the estimator being evaluated produces the estimates and ranking shown in the Merits–Estimate and Rankings–Estimate tables. There are two things to note about the estimated merits and rankings. First, the ranking produced by the estimator does not match the baseline ranking exactly; therefore, relevant documents will not be accumulated as quickly as they could be. Second, a more subtle point is the estimated merit of collection B for query q_3. Our hypothetical estimator has estimated that collection B has zero merit with respect to query q_3. As a result, we denote collection B as "not selected" for query q_3. Note that collection B is italicized in the Ranked Collections table of Figure 5.2. This is

meant to convey that while collection B is placed at the bottom of the ranking for rank correlation purposes, it will not be counted by some of our evaluation measures. This is not a frequently–occurring case in our experiments, but it does become an issue in Section 6.3.2 of Chapter 6, so we cover it here.

First, we consider the MSE of the baseline and estimated rankings for the three queries. The MSE values for q_1, q_2 and q_3 are 1.33, 1.33 and 2.25, respectively. The maximum MSE value for each query is 11.67. This illustrates the utility of considering the maximum MSE value. Our example represents a far different test environment than the *SYM-236* testbed, for example, which has a maximum MSE of 18,565. Note that the MSE values for q_1 and q_2 are identical. However, when we consider the merit values assigned to the collections, we find that placing the two collections with the most merit at ranks 2 and 3 has the potential for more serious performance consequences for query q_2; if we only search the top-ranked collection we have missed more relevant documents. The $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ measures utilize merit information whereas MSE does not.

The middle portion of Figure 5.2 redisplays the baseline and estimate collection rankings and illustrates how the baseline merits are used to evaluate the estimate. This was alluded to in the definition of $B_i = merit(q, C_{b_i})$ and $E_i = merit(q, C_{e_i})$. Recall that while estimated merits are used to produce the estimated rankings, the actual merits (as used to compute the baseline) are substituted in when evaluating the estimator. This is because in our evaluation we want to determine how quickly we're accumulating merit (approximating the baseline in terms of the baseline merit). The example of Figure 5.2 makes this clearer. In our example we are interested in locating relevant documents. So, given the estimated ranking for a query, we want to determine how many relevant documents are actually available in the n top ranked collections. We use the estimated ranking, but substitute in the number of relevant documents found in the collections when performing the evaluation. Note that the merit of collection B in the estimated ranking for query q_3 is left empty because B is not selected by the estimator. Also note that the use of baseline merits in the computation of $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ means that in general these measures are not

Baseline Merits

	A	B	C	D	E	F
q_1	6	2	9	5	1	7
q_2	4	18	3	9	5	1
q_3	2	1	2	0	4	0

Estimated Merits

	A	B	C	D	E	F
q_1	0.7	0.4	0.6	0.2	0.1	0.5
q_2	0.2	0.8	0.4	0.7	0.9	0.1
q_3	0.2	0.0	0.3	0.4	0.5	0.1

Baseline Rankings

	1	2	3	4	5	6
q_1	C	F	A	D	B	E
q_2	B	D	E	A	C	F
q_3	E	A	C	B	D	F

Estimated Rankings

	1	2	3	4	5	6
q_1	A	C	F	B	D	E
q_2	E	B	D	C	A	F
q_3	E	D	C	A	F	*B*

Merits as Used to Compute $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$

Ranked Collections

	q_1		q_2		q_3	
	B	**E**	**B**	**E**	**B**	**E**
1	C	A	B	E	E	E
2	F	C	D	B	A	D
3	A	F	E	D	C	C
4	D	B	A	C	B	A
5	B	D	C	A	D	F
6	E	E	F	F	F	*B*

Substitute in Merits

	q_1		q_2		q_3	
	B	**E**	**B**	**E**	**B**	**E**
1	9	6	18	5	4	4
2	7	9	9	18	2	0
3	6	7	5	9	2	2
4	5	2	4	3	1	2
5	2	5	3	4	0	0
6	1	1	1	1	0	$\emptyset$

$\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ results

		n					
		1	2	3	4	5	6
q_1	$\mathcal{R}_n$	6/9	15/16	22/22	24/27	29/29	30/30
	$\widehat{\mathcal{R}}_n$	6/30	15/30	22/30	24/30	29/30	30/30
	$\mathcal{P}_n$	1/1	2/2	3/3	4/4	5/5	6/6
q_2	$\mathcal{R}_n$	5/18	23/27	32/32	35/36	39/39	40/40
	$\widehat{\mathcal{R}}_n$	5/40	23/40	32/40	35/40	39/40	40/40
	$\mathcal{P}_n$	1/1	2/2	3/3	4/4	5/5	6/6
q_3	$\mathcal{R}_n$	4/6	4/6	6/8	8/9	8/9	8/9
	$\widehat{\mathcal{R}}_n$	4/9	4/9	6/9	8/9	8/9	8/9
	$\mathcal{P}_n$	1/1	1/2	2/3	3/4	3/5	3/6

Figure 5.2: An example evaluation using the $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ evaluation measures. The collection rankings from the example in Figure 5.1 are evaluated.

symmetric.

The lower portion Figure 5.2 illustrates how the performance of the estimated rankings for q_1, q_2 and q_3 will appear when evaluated using $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$. Again, $\mathcal{R}_n$ shows the rate at which the estimator accrues *available* relevant documents while $\widehat{\mathcal{R}}_n$ shows the rate at which the total number of relevant documents are accrued. $\mathcal{P}_n$ shows the fraction of collections that have any relevant documents. Because all collections have relevant documents for queries q_1 and q_2, the $\mathcal{P}_n$ measure doesn't shed any light on performance for those queries. However, $\mathcal{R}_n$ and $\widehat{\mathcal{R}}_n$ allow a comparison of q_1 and q_2 and show what the MSE measure did not. Despite the fact that the degree of misplacement was the same, the misplacement of collection E for query q_2 has greater performance implications than the similar misplacement of collection A for query q_1.

5.1.7 Expected Performance of Random Selection

Another approach to evaluating collection selection algorithms is to ask how they compare to randomly generated rankings. Losee has suggested evaluating IR system performance analytically [Los95]. This approach can be extended to collection selection algorithms and lets us derive the expected performance of an algorithm that generates rankings randomly. We can then use this result to examine the behavior of other algorithms to see if they have better performance than an algorithm that generates rankings randomly. We develop these ideas further in this section.

It should be noted that comparison with randomly-generated rankings is of more than theoretical interest. There are realistic situations in which we could not expect any algorithm to perform better than an algorithm that simply selects an ordering at random. As a simple example, consider the extreme case in which each collection $C \in \mathcal{C}$ has $merit(q, C) = 1$. In this situation any ordering of the collections is as good as any other; all orderings achieve the same effectiveness. More generally, this situation arises whenever all of the available merit for a query is spread approximately evenly across all or any subset of the collections. In these situations, and they do occur reasonably frequently, we can

look to the expected value of randomly-generated rankings to develop a lower bound on performance. Of course, SBR is often more useful as an operational lower bound but much can be learned by examining these random-generated rankings. In particular, we have seen instances of queries for which collection selection algorithms actually perform worse than random selection (see Table 7.1 in Chapter 7).

Definition 1 *Given a set of collections,* $\{C_1, C_2, ..., C_N\}$*, a random ranking algorithm is one in which each of the* $N!$ *permutations is equally likely.*

So by *randomly generated ranking* we mean selecting a permutation from the uniform distribution of all permutations.

Lemma 1 *Given a randomly generated ranking of* N *collections, let* X_n *denote the number of collections in the first* n *ranks having nonzero merit.* X_n *is a hypergeometric random variable with expected value given by*

$$E[X_n] = n \cdot n^*/N$$

Proof: This follows from the theorem in the Appendix C where $M = N$, and $W = n^*$.

Theorem 2 *The expected value of the precision,* $E[\mathcal{P}_n]$*, of the first* n *elements of a randomly generated ranking is given by*

$$E[\mathcal{P}_n] = \frac{n^*}{N}$$

Proof: Given a random ranking, let X_n denote the number of collections in the first n ranks having nonzero merit. Then

$$\mathcal{P}_n = \frac{|\{C_i \in Top_n(E) | merit(q, C_i) > 0\}|}{n} = \frac{X_n}{n} \text{ so that } E[\mathcal{P}_n] = \frac{E[X_n]}{n}$$

The result now follows directly from Lemma 1.

We would expect a good ranking algorithm to have significantly greater precision.

Corollary 2.1 $E[\mathcal{P}_*] = \frac{n^*}{N}$.

We have fully characterized $\mathcal{P}_n$ for randomly generated rankings. Now let's take a look at $\mathcal{R}_n$ and $\widehat{\mathcal{R}}_n$ for these rankings.

Condition 1 *The total merit M is spread evenly over all n^* collections.*

The net effect of this condition is to make all rankings with $\mathcal{P}_* = 1$ equivalent.

Theorem 3 *Let X_n denote the number of collections having nonzero merit in the first n ranks of a randomly generated ranking. If Condition 1 holds then*

1. $\mathcal{R}_n = \mathcal{P}_n$
2. $E[\mathcal{R}_n] = n^*/N$
3. $\widehat{\mathcal{R}}_n = X_n/n^*$
4. $E[\widehat{\mathcal{R}}_n] = n/N$

Proof: Condition 1 implies that each collection with nonzero merit contributes M/n^* merit to the accumulated total.

Part 1:

$$\mathcal{R}_n = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n} B_i} = \frac{X_n \cdot (M/n^*)}{n \cdot (M/n^*)} = \frac{X_n}{n} = \mathcal{P}_n$$

Part 2: Taking expectations from Part 1 we have $E[\mathcal{R}_n] = E[\mathcal{P}_n]$. The result now follows directly from Theorem 2.

Part 3:

$$\widehat{\mathcal{R}}_n = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n^*} B_i} = \frac{X_n \cdot (M/n^*)}{M} = \frac{X_n}{n^*}.$$

Part 4: From Part 3 and Lemma 1 we have

$$E[\widehat{\mathcal{R}}_n] = \frac{E[X_n]}{n^*} = \frac{n}{N}.$$

Corollary 3.1 *If Condition 1 holds then for all rankings in which* $\mathcal{P}_n = 1$*, we have* $\mathcal{R}_n = 1$ *and* $\widehat{\mathcal{R}}_n = n/n^*$*,* $n \leq n^*$*.*

Corollary 3.2 *If Condition 1 holds then*

$$\widehat{\mathcal{R}}_n = \frac{n}{n^*}\mathcal{P}_n = \frac{n}{n^*}\mathcal{R}_n$$

where $n \leq n^*$*.*

Proof: From Theorem 3(3)and Theorem 3(1).

Theorem 4 $E[\widehat{\mathcal{R}}_n] = \frac{n}{N}$

Proof:

$$\begin{aligned} E[\widehat{\mathcal{R}}_n] &= \frac{\text{expected merit}}{\text{total merit}} \\ &= \frac{n \cdot \text{expected merit per collection}}{M} \\ &= \frac{n\left(\frac{M}{N}\right)}{M} = \frac{n}{N} \end{aligned}$$

Note that this is a stronger result than Theorem 3(4) since it does not require Condition 1. So Theorem 4 can be used to determine the expected value of $\widehat{\mathcal{R}}_n$ for an arbitrary baseline.

Corollary 4.1 $E[\mathcal{R}_*] = E[\widehat{\mathcal{R}}_*] = E[\mathcal{P}_*] = \frac{n^*}{N}$.

Proof: By definition $\mathcal{R}_* = \widehat{\mathcal{R}}_*$, so by Theorem 4 $\mathcal{R}_* = \widehat{\mathcal{R}}_* = \frac{n^*}{N}$. Together with Corollary 2.1 this completes the proof.

Theorem 5 $E[\mathcal{R}_n] = \frac{M}{\sum_{i=1}^{n} B_i} E[\widehat{\mathcal{R}}_n]$

Proof: From Theorem 1 we have

$$\widehat{\mathcal{R}}_n(E, B) = \mathcal{R}_n(E, B) \cdot \widehat{\mathcal{R}}_n(B, B).$$

Taking expectations yields

$$E[\widehat{\mathcal{R}}_n(E,B)] = \widehat{\mathcal{R}}_n(B,B) \cdot E[\mathcal{R}_n(E,B)]$$

since $E[cX] = cE[X]$. The desired result follows from $\widehat{\mathcal{R}}_n(B,B) = \frac{\sum_{i=1}^{n} B_i}{M}$.

The following corollary to Theorem 5 is immediate from Theorem 4.

Corollary 5.1 $E[\mathcal{R}_n] = \frac{nM}{N\sum_{i=1}^{n} B_i}$

5.2 Evaluating Data Item Retrieval

Data item retrieval is generally evaluated in the same way for both single-collection and multi-collection environments. Data item retrieval evaluation in a single-collection environment was discussed in more detail in Chapter 2. Given a list of retrieved data items, recall and precision or precision at fixed points can be reported.

5.2.1 Recall and Precision

Recall and precision are two commonly used data item retrieval evaluation measures. These measures were initially presented in Chapter 2; we repeat them here for reference.

$$recall = \frac{\text{number of relevant documents retrieved}}{\text{total number of relevant documents in the collection}}$$

$$precision = \frac{\text{number of relevant documents retrieved}}{\text{number of documents retrieved}}$$

When considered in conjunction, recall and precision are usually presented as recall-precision curves. For this approach, precision values are computed at fixed recall points. These results are generally presented as graphs where precision values are computed for $R = 0.1, 0.2, \ldots, 1.0$. Plots closer to $R = 1, P = 1$ denote better performance.

5.2.2 Precision at Fixed Points

For the data item retrieval experiments reported in Chapter 9 we use the approach that has been used for reporting TREC experimental results. We report precision at fixed values, when 5, 10, 15, 20, 50 and 100 documents have been retrieved.

6

Early *gGlOSS* and *CORI* Experiments

Our early comparative collection selection experiments began with an evaluation of the *gGlOSS* [GGM95, GGMT99] collection selection approach. Our initial evaluation of *gGlOSS* is covered in greater detail in French *et al.* [FPV+98]. We summarize the evaluation results and provide some additional analysis here.

The original *gGlOSS* work [GGM95] defined two parameterized estimators, *Max(l)* and *Sum(l)*, plus a parameterized baseline *Ideal(l)* representing the desired performance of *Max(l)* and *Sum(l)*. *Ideal(l)* is based upon similarities of data items in a collection to a query. The original *gGlOSS* evaluation compared the *gGlOSS* rankings produced by *Max(l)* and *Sum(l)* to the *Ideal(l)* rankings. Gravano and García-Molina [GGM95] argued that "ranks based on end-user relevance are not appropriate for evaluating schemes like *gGlOSS*." They further claimed that "the best we can hope for any tool like *gGlOSS* is that it predicts the answers that the databases will give when presented with a query." We strongly agreed with the latter assertion, but felt that there was something to be learned by evaluating against end-user relevance; in the end that is the only interesting metric from a user's standpoint. We conducted a study of *gGlOSS*, focusing on two major questions.

1. How well do the *gGlOSS* estimators predict the *Ideal(l)* baseline?

2. How well does *Ideal(l)* predict a relevance-based ranking (RBR as defined in Chapter 5)?

After our initial study of *gGlOSS*, we moved to a direct comparison of *gGlOSS* and the *CORI* collection selection algorithm. Some steps were required to allow a direct comparison. First, we selected a single representative for *gGlOSS*, the *Ideal(0)* baseline. Second, we undertook vocabulary resolution steps to allow a straightforward comparison. These vocabulary resolution steps are detailed in Appendix A. Having completed those steps, we performed the initial comparison that is reported here. Additional collection selection algorithm comparisons using more algorithms, alternate query formulations and additional testbeds are reported in Chapter 7.

6.1 Test Environment

We used a single test environment for the experiments reported in this chapter. The test environment can be specified (*SYM-236*, $\mathcal{Q}_{vl}$, $\mathcal{J}_{TREC4}$). We used the *SYM-236* testbed as defined in Chapter 4, the very long query formulations created using TREC topics 51-150 and the relevance judgements supplied with the TREC-4 data. TREC topics 51-150 were chosen for reasons outlined in Chapter 4. The very long query formulations, $\mathcal{Q}_{vl}$, are only used for the experiments reported in this chapter and in Appendix A. For the experiments of Chapter 7–9 we moved to the short and long query formulations, $\mathcal{Q}_s$ and $\mathcal{Q}_l$, to be more compatible with other reported results.

We prepared each collection by using version 11.0 of the SMART information retrieval system [Buc92]. Statistical information extracted from the SMART indexes is the input to *gGlOSS*. SMART is freely-available research software and is a vector-space model retrieval system (see Chapter 2 for a discussion of different types of retrieval systems). We used the following SMART parameters to be consistent with published *gGlOSS* evaluation [GGM95].

- Documents were indexed using SMART indexing parameter *ntc*: document term weights were formed using a combination of term frequency (tf) and inverse document frequency (idf), normalized by vector length.

- Queries were indexed using SMART indexing parameter *nnn*: query term weights

were based on query term frequency (tf).

- Query-document similarity was computed using the dot product of the document and query vectors.

Note that for these experiments each of the 236 collections used the same parameters and search engine to process queries.

The next step was to prepare a union vocabulary incorporating all of the terms appearing at any of the separate collections. This gave us a canonical global vocabulary with which to store the document frequencies and weight sums required by *gGlOSS* to make its estimates.

Next, TREC topics 51-150 were indexed by SMART to convert them into term lists in the global vocabulary for use by *gGlOSS*. Finally, we computed the *Ideal(0)* and *Ideal(0.2)* baselines and produced *gGlOSS* rankings for each of the queries using the *Max(0)*, *Max(0.2)*, *Sum(0)* and *Sum(0.2)* estimators. These are the same threshold values used by Gravano and García-Molina [GGM95] in their experiments.

Before presenting results of the comparison of the estimators to the baselines, we will provide a more detailed description of the *gGlOSS Max(l)*, *Sum(l)* and *Ideal(l)* computations.

6.2 *gGlOSS*

Gravano *et al.* [GGMT94b] proposed *GlOSS*, the *Glossary-of-Servers Server*, as an approach to the collection selection problem. *GlOSS* originally assumed a Boolean retrieval model but was later generalized to *gGlOSS*[GGM95] to handle the vector space information retrieval model.

gGlOSS assumes that a group of collections can be characterized according to their *goodness* with respect to any particular query. *gGlOSS*'s job is then to estimate the goodness of each candidate collection with respect to a particular query and then suggest a ranking of the collections according to the estimated goodness.

6.2.1 Goodness and Ideal Ranks

In [GGM95], Gravano and García-Molina made the following two assumptions:

1. all of the collections in a testbed, i.e. group of collections, employ the same algorithms to compute term weights and similarities; and
2. given a query q and a document d, d is only useful for q if $sim(q, d) > l$ for a given similarity threshold l and similarity measure $sim(q, d)$.

They then defined a notion of *goodness* for each collection C_i as follows.

$$Goodness(l, q, C_i) = \sum_{\{d \in C_i | sim(q,d) > l\}} sim(q, d) \tag{6.1}$$

Once $Goodness(l, q, C_i)$ has been calculated for each collection C_i with respect to q at threshold l, the ideal rank for the query at threshold l, *Ideal(l)* can be formed by sorting the collections in descending order of their goodness.

Note that *gGlOSS* does not compute *Ideal(l)*, rather *Ideal(l)* is advanced as the goal to which *gGlOSS* ranks will be compared. The strategy employed by *gGlOSS* is to attempt to estimate the goodness of each collection and thereby create a ranking. Since by hypothesis the collection utility to the query is expressed by goodness, it is reasonable to measure the performance of *gGlOSS* by how well it estimates this goodness. That leaves open the question of how well goodness correlates to relevance.

6.2.2 *gGlOSS* Estimators

gGlOSS needs two vectors of information from each collection C_i in order to make its estimates. These vectors are stored as two matrices, F and W.

1. $F = [df_{ij}]$ where df_{ij} is the document frequency (the number of documents in the collection containing the term) for each term t_j in C_i; and
2. $W = [w_{ij}]$ where w_{ij} is the sum of the weights of each term t_j over all documents in C_i.

This information is gathered periodically by unspecified means from all collections in the group of collections and used to formulate estimates.

As we have already noted, *gGlOSS* creates rankings for each query by estimating the goodness of each collection with respect to that query. There are many ways that one might make these estimates; two estimates, *Max(l)* and *Sum(l)* were reported in [GGM95]. Complete details for calculating the *Max(l)* and *Sum(l)* estimators are given in Gravano *et al.* [GGM95, GGMT99]. The equations are reproduced here for reference; the *gGlOSS* papers contain more discussion and examples. Our reproduction of the equations uses the notation of this dissertation. As a result there are some notational differences between the *gGlOSS* papers and this dissertation.

6.2.2.1 High-Correlation Scenario

The *Max(l)* estimator is based on what Gravano and García-Molina [GGM95] refer to as a high-correlation scenario. If two query terms t_1 and t_2 appear in C_i and $df_{i1} \leq df_{i2}$ then it is assumed that every document in C_i that contains t_1 also contains t_2. For this scenario, it is assumed that terms are ordered such that $df_{ij} \leq df_{ik}$ for all $j \leq k$.

For the computation of *Max(l)* over collection C_i, *gGlOSS* first defines

$$sim_k = \sum_{j=k}^{n} q_j \times \frac{w_{ij}}{df_{ij}}$$

The *Max(l)* ranking is produced by sorting collections by their estimated merit. For *Max(l)*, the estimated merit is computed by $Estimate(l, q, C_i)$ as defined below. To compute $Estimate(l, q, C_i)$, *gGlOSS* determines the value p such that $sim_p > l$ but $sim_{p+1} \leq l$.

Given the value of p,

$$Estimate(l, q, C_i) = \sum_{j=1}^{p} (df_{ij} - df_{i(j-1)}) \times sim_j \tag{6.2}$$

6.2.2.2 Disjoint Scenario

The *Sum(l)* estimator is based on what Gravano and García-Molina [GGM95] refer to as the disjoint scenario. For any two query terms, the set of documents in collection C_i containing term t_1 is assumed to be disjoint with the set of documents containing term t_2. Like the *Max(l)* estimator, the *Sum(l)* estimator is produced by sorting collections based on their estimated merit. In the case of *Sum(l)*, the estimated merit is computed as follows.

$$Estimate(l, q, C_i) = \sum_{\left\{j=1..n \middle| \left((df_{ij}>0) \wedge \left(q_j \times \frac{w_{ij}}{df_{ij}} > l\right)\right)\right\}} df_{ij} \times \left(q_j \times \frac{w_{ij}}{df_{ij}}\right) \tag{6.3}$$

In both *Max(l)* and *Sum(l)* it is further assumed that the weight of a term is distributed uniformly over all documents that contain the word.

gGlOSS uses the assumptions underlying *Max(l)* or *Sum(l)* to estimate the number of documents in a collection C_i having similarity to a query greater than a threshold l. This forms the basis for the *gGlOSS* estimate of the *goodness* of C_i.

6.2.2.3 *Max(0) = Sum(0) = Ideal(0)*

Gravano and García-Molina [GGM95] note that the definitions of $Estimate(l, q, C_i)$ are identical for *Max(l)* and *Sum(l)* when $l = 0$.

$$Estimate(0, q, C_i) = \sum_{j=1}^{n} q_j \times w_{ij} \tag{6.4}$$

We note that *Max(0)* and *Sum(0)* are also identical to *Ideal(0)*, that is, at threshold $l = 0$ both estimators give identically the *Ideal(0)* ranking of collections for all queries. This is an artifact of the summing approach of the computation of the estimators and *Ideal(0)*. When $l = 0$, all documents contribute to $Goodness(l, q, C_i)$ and to $Estimate(l, q, C_i)$.

6.3 Comparing the *Max(l)* and *Sum(l)* Estimators to the *Ideal(l)* Baseline

Our first experiments were essentially a confirmation of the results presented by Gravano and García-Molina [GGM95], who found that the *Max(l)* and *Sum(l)* estimators approximated the *Ideal(l)* baseline very accurately. Their experiments were conducted using 53 collections; we verified that *Max(l)* and *Sum(l)* performed well in our 236 collection test environment before proceeding with experiments to determine how well *gGlOSS* approximates relevance.

6.3.1 Mean Squared Error

We first evaluated the *gGlOSS* rankings against the *Ideal(l)* baseline using the mean squared error (MSE) of the ranks. Our usage of mean squared error is described in Chapter 5. Because *Max(0)* = *Sum(0)* = *Ideal(0)*, we simply verified this case. For all other cases we produced a plot of MSE by query ID to help get a sense of the error distribution. The plots of these comparisons are shown in Figure 6.1. The plot labels in the graphs are of the form E.B where E is the estimate ranking and B is the baseline ranking. This labelling convention will be employed for many of the figures in the remainder of the dissertation.

Because *Max(0)* = *Sum(0)* = *Ideal(0)*, *Max(0)* and *Sum(0)* are not plotted in the *Ideal(0)* graph of Figure 6.1. Also, *Max(0)* and *Sum(0)* are represented by the same plot in the *Ideal(0.2)* graph of Figure 6.1. Overall, the *Max(0)* and *Max(0.2)* estimators had a very small mean squared error when compared to both the *Ideal(0)* and *Ideal(0.2)* baselines. *Sum(0.2)* had considerably more variability and higher MSE values. Our initial MSE analysis suggests that the *gGlOSS* estimators are reasonably accurate models of *Ideal(l)*.

Of particular interest here is the observation that ranking differences vary on a query-by-query basis, for example *Sum(l)* estimator is much more accurate for some queries than for others.

Unfortunately, while MSE can identify variability, it is not very useful for resolving

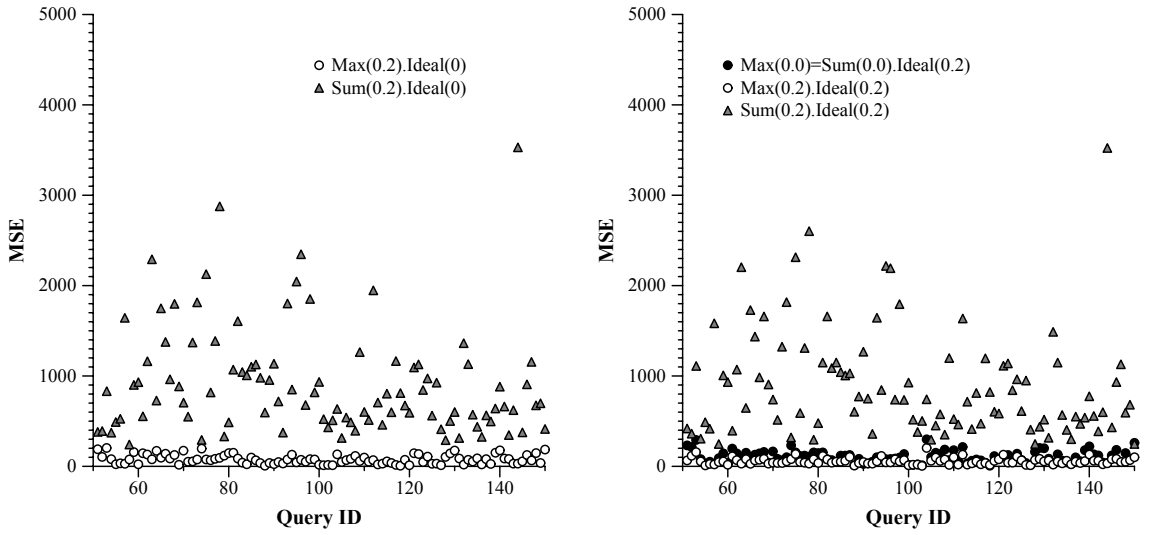


Figure 6.1: MSE comparison of *gGlOSS Max(l)* and *Sum(l)* estimators to *Ideal(0)* and *Ideal(0.2)* baselines.

the source of variability. A high MSE value could be caused by a few very badly ranked collections or a large number of smaller ranking errors. In addition, MSE cannot reveal if the ranking differences have broader performance implications. We probe this further and utilize additional performance measures in the next section.

6.3.2 Recall and Precision Analogs

Our continuing analysis used the $\mathcal{R}_n$ and $\mathcal{P}_n$ measures proposed by Gravano and García-Molina [GGM95], plus our $\widehat{\mathcal{R}}_n$ measure. These measures are covered in detail in Chapter 5. We continued to study the degree to which the *Max(l)* and *Sum(l)* estimators approximated the *Ideal(l)* baseline.

Our analysis involved several different representations and interpretations of the results using these evaluation measures. We plotted $(n, \mathcal{R}_n)$, $(n, \widehat{\mathcal{R}}_n)$ and $(n, \mathcal{P}_n)$ averaged over all queries for $n = 1...N$ where N is the total number of collections in the testbed. $(n, \mathcal{R}_n)$, $(n, \widehat{\mathcal{R}}_n)$ and $(n, \mathcal{P}_n)$ plots are used most frequently in the remainder of the dissertation to present average estimator performance, so plots from French *et al.* [FPV+98] are reproduced

here to show *Max(l)* and *Sum(l)* performance.

Our original analysis also contained a number of scatterplots on the $(\mathcal{R}, \mathcal{P})$ plane. Each point on the plane represented the $\mathcal{R}_n$ and $\mathcal{P}_n$ values measured for a particular query when compared against the operative baseline metric. 100 points, one for each query, were plotted. We considered two fixed values of n, 5 and 10, as well as n^*[1]. The two fixed values were chosen to determine how well an estimated ranking would perform when selecting a small number of collections. The scatterplots can be found in French *et al.* [FPV+98] but are not reproduced here. The $(n, \mathcal{R}_n)$, $(n, \widehat{\mathcal{R}}_n)$ and $(n, \mathcal{P}_n)$ plots are sufficient to demonstrate that the *Max(l)* and *Sum(l)* estimators approximate the *Ideal(l)* baseline very well. The larger question, which we will address in Section 6.4 is whether or not the overall *gGlOSS* approach can predict the presence of relevant documents.

The $(n, \mathcal{R}_n)$, $(n, \widehat{\mathcal{R}}_n)$ and $(n, \mathcal{P}_n)$ plots are found in Figure 6.2. Only baseline *Ideal(0)* comparisons are shown here. The primary observation to make is that these results corroborate the MSE conclusion that the *gGlOSS* estimators model *Ideal(0)* well.

In addition, a comparison of the $\mathcal{R}_n$ and $\widehat{\mathcal{R}}_n$ graphs in Figure 6.2 illustrates the differences in interpretation of $\mathcal{R}_n$ and $\widehat{\mathcal{R}}_n$, which provide different points of view. The $\mathcal{R}_n$ plots track how much of the available merit in the top n collections of the baseline has been accumulated by the top n collections of the estimate. In contrast, the $\mathcal{R}_n$ plots track the fraction of *total* merit accumulated. It is apparent that the two measures can produce drastically different numeric results; however, the two measures are mathematically related (see Chapter 5).

In addition to a comparison of estimator performance, the $\widehat{\mathcal{R}}_n$ graph of Figure 6.2 provides insight into the distribution of merit (goodness) in the *SYM-236* testbed. In general, the merit with respect to a query is not evenly distributed. Recall from Figure 4.3 that there are a number of collections with very few documents. These collections tend to result in very low merit and are usually (correctly) placed at the bottom of the rankings. The "knee" in the plots on the $\widehat{\mathcal{R}}_n$ graph represents a transition from high- or medium-merit

[1]Recall that n^* is the number of collections with non-zero merit with respect to a query.

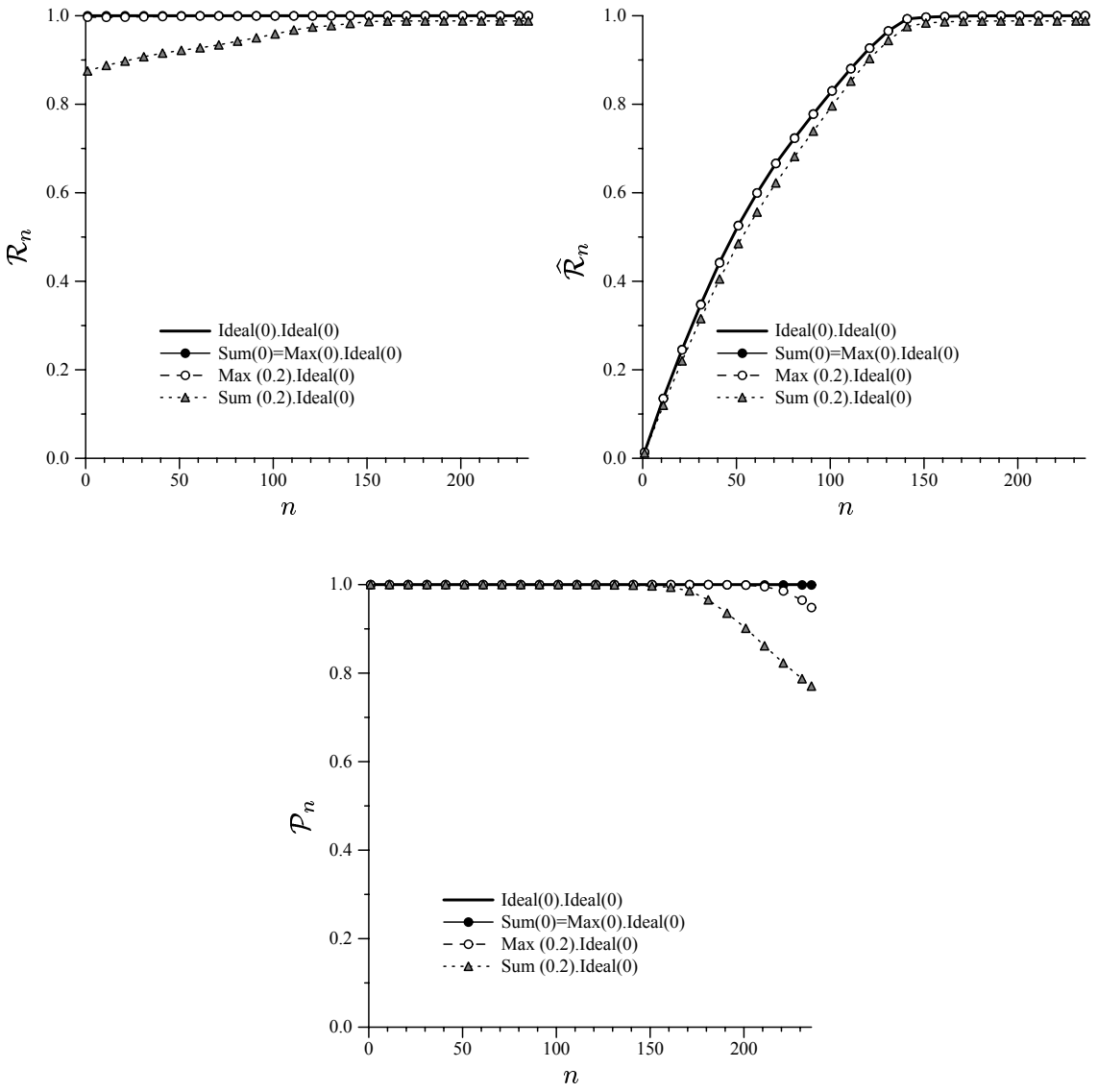


Figure 6.2: Comparison of *gGlOSS Max(l)* and *Sum(l)* estimators to the *Ideal(0)* baseline using $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ measures.

collections to low-merit collections. In general, nearly all of the total merit is accounted for when 60% of the collections have been selected.

Also of interest is the $\mathcal{P}_n$ graph of Figure 6.2. The first thing to note is that for values of n less than approximately 150, all of the estimators exhibit what appears to be *very* good performance under the $\mathcal{P}_n$ measure. However, this is due to a feature of the baseline rather than to features of the estimators. Under the *Ideal(0)* baseline, nearly all collections have non-zero merit for all queries. As a result, there are very few zero-merit collections to be interleaved in the ranking, virtually guaranteeing good $\mathcal{P}_n$ performance for any estimator.

Given this, the performance for values of n greater than approximately 150 in the $\mathcal{P}_n$ graph of Figure 6.2 is surprising at first glance. The *Max(0.2).Ideal(0)* and *Sum(0.2).Ideal(0)* plots exhibit behavior that is not found in other $\mathcal{P}_n$ graphs that we present. Namely, there is a drop in precision for high values of n. This is due to the non-zero value of l used for those estimators. For the *SYM-236* testbed, there exist queries and collections for which the estimated merit is zero. However, the actual merit (goodness) is very small but non-zero. Due to the estimated merit of zero, we denote those collections as "not selected". However, our evaluation methodology continues to evaluate selection performance as long as a collection with non-zero merit has not yet been selected. The dip in $\mathcal{P}_n$ is a combination of these two experimental decisions and can be interpreted as a penalty for incorrectly assigning zero merit to a collection. A milder version of this effect can also be seen in the plots on the $\mathcal{R}_n$ graph where a close observation reveals that the *Sum(0.2).Ideal(0)* curve approaches but does not reach 1.0. The effect there is less noticeable because the missed collections have very small actual merit. The effect on the $\mathcal{R}_n$ measure is smaller than that on the $\mathcal{P}_n$ measure. The $\mathcal{P}_n$ measure reveals that a collection with merit was missed while $\mathcal{R}_n$ takes into account the amount of merit missed. This effect is not seen in other plots because most of the measures we evaluate rarely assign zero merit.

6.4 Comparing *gGlOSS* to the RBR baseline

In the previous section, we confirmed that the *gGlOSS Max(l)* and *Sum(l)* estimators approximate the *Ideal(l)* baseline well for our *SYM-236* testbed. This verifies the findings of Gravano and García-Molina [GGM95]. Our next step was to study the extent to which the *gGlOSS* estimators and baselines are able to predict relevance.

6.4.1 Mean Squared Error

We first examined the mean squared error of the RBR and *gGlOSS* rankings. While the rankings of the *gGlOSS* estimators and baselines are generally self-similar, the MSE analysis suggests that the *gGlOSS* rankings are not strong predictors of RBR. The MSE values for comparisons of all *gGlOSS* measures to the RBR baseline tend to be high. Due to the self-similarity of the *gGlOSS* measures and their general dissimilarity to RBR, there is very little to differentiate the different MSE comparisons of the *gGlOSS* measures to RBR.

As we noted earlier, MSE is useful for determining the degree of difference in two rankings but not the cause or impact of that difference. This shortcoming of MSE makes the measure less useful than the $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ measures for these comparisons.

The MSE comparison of *Ideal(0)* to the RBR baseline is representative of the *gGlOSS* comparisons. We include it as Figure 6.3 as an example.

6.4.2 Recall and Precision Analogs

Our next step was to compare the *gGlOSS* rankings to the RBR baseline using the $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ recall and precision analogs. The plots of these measures for $n = 1...N$ are shown in Figure 6.4. We would like to emphasize that the difference between Figure 6.4 and Figure 6.2 is the baseline to which the estimators are being compared. The measures and their interpretation are the same.

Overall, neither the *gGlOSS* baselines nor estimators approximate the RBR baseline closely, especially for small values of n. We will discuss each of the graphs of Figure 6.4 in

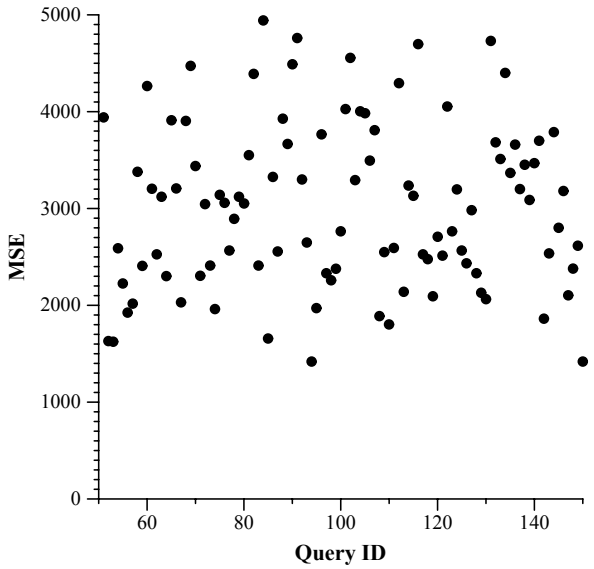


Figure 6.3: Mean squared error when the *gGlOSS Ideal(0)* ranks are compared to the relevance-based ranking (RBR).

more detail. In all graphs, the plot RBR.RBR represents the baseline's estimation of itself, i.e. the best possible performance for an estimator compared to the baseline under a given performance measure.

The $\mathcal{R}_n$ graph of Figure 6.4 reveals that the collections with the most merit (in terms of relevant documents) tend not to be ranked extremely highly by the *gGlOSS* estimates. Over the top twenty ranked collections, only about 40-55% of the potential relevant documents are located by the estimators. This is due to a combination of factors. First, it is often the case that the estimators rank highly collections with average, but non-zero, merit. Unfortunately, it is also the case that the estimates rank highly collections with no merit with respect to the query. The $\mathcal{P}_n$ graph of Figure 6.4 reveals the degree to which the *gGlOSS* approaches interleave zero-merit collections into the rankings.

The $\widehat{\mathcal{R}}_n$ graph of Figure 6.4 illustrates why ranking highly collections with small or zero merit is problematic. The RBR.RBR curve in the $\widehat{\mathcal{R}}_n$ graph shows the distribution of merit for the *SYM-236* testbed, averaged over 100 queries. Overall, the distribution of relevant

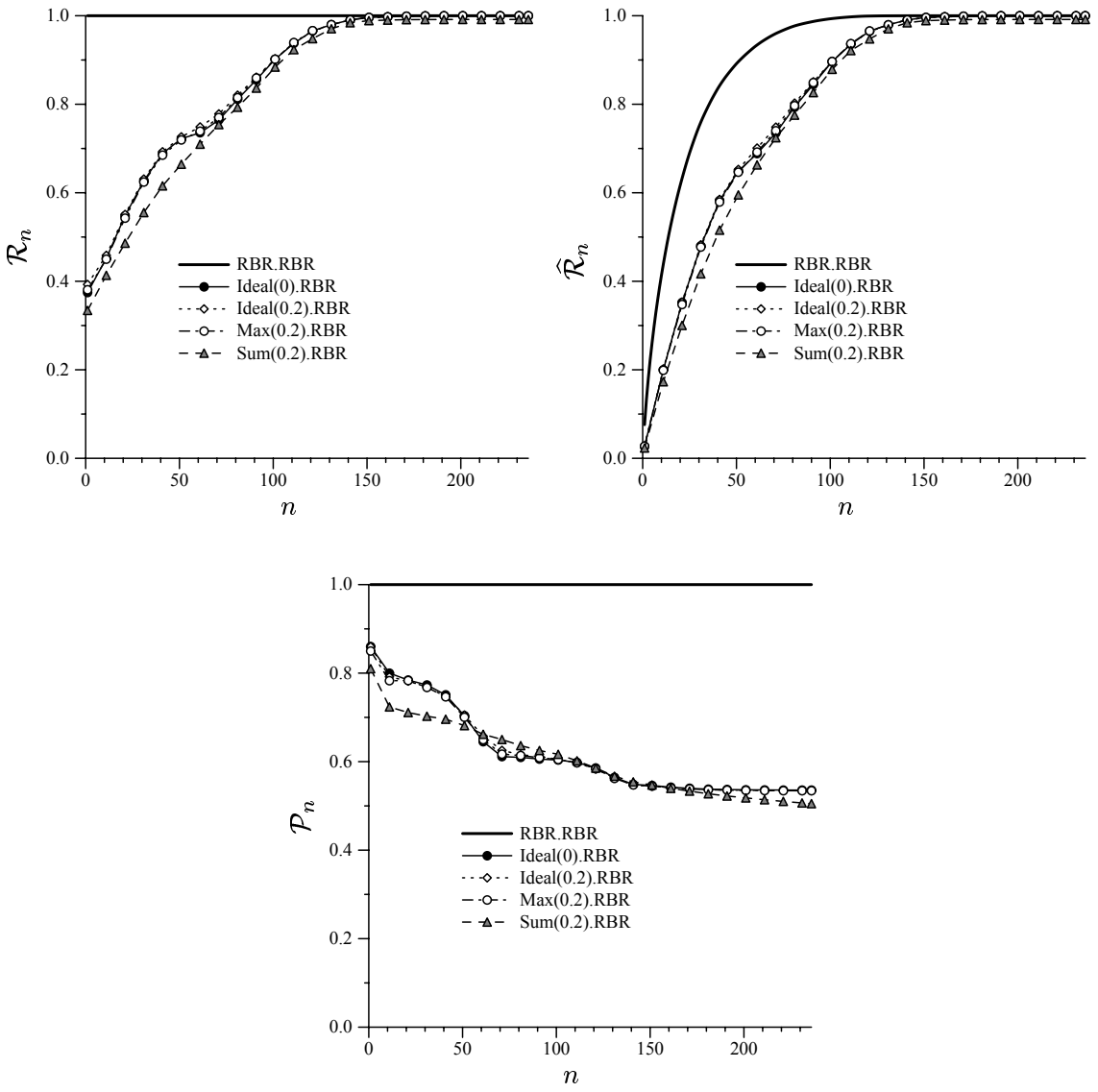


Figure 6.4: Comparison of *gGlOSS Max(l)* and *Sum(l)* estimators to the RBR baseline using $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ measures.

documents is very skewed. For an average query, relevant documents are found in less than half of the collections with some collections containing many more relevant documents than others. In a testbed with characteristics like this, errors in ranking have the potential to produce noticeable degradation in measured selection performance.

The "knee" in the curves that can be observed in the $\mathcal{R}_n$ and $\widehat{\mathcal{R}}_n$ graphs of Figure 6.4 is indicative of interesting features of both the *SYM-236* testbed and the *gGlOSS* estimators and *Ideal(l)* baseline. In the *SYM-236* testbed, there are a number of collections that have no merit for most queries. The *gGlOSS* estimators and *Ideal(l)* baseline all do well at identifying those special-case collections. However, *gGlOSS* is not infallible at separating useful and useless collections, as evidenced by the $\mathcal{P}_n$ plots.

6.4.2.1 $(\mathcal{R}_*, \mathcal{P}_*)$ Scatterplots

We mentioned is Section 6.3.2 that our original analysis included a number of scatterplots on the (R, P) plane. Each point represents a single query and shows the $\mathcal{R}_n$ and $\mathcal{P}_n$ values measured for that query when the estimated ranking was compared to the operative baseline metric. Most of the scatterplots found in French *et al.* [FPV+98] are not reproduced; however, we include one here both because it raises an interesting point and to illustrate the strengths and weaknesses of that representation.

Figure 6.5 is a plot of $(\mathcal{R}_{n^*}, \mathcal{P}_{n^*})$ for the performance of $Ideal(0)$ with respect to the RBR baseline. This figure is typical of all the comparisons that we made between *gGlOSS* estimators and the RBR baseline. We want to emphasize that n^* is potentially different for every query and this affects the way that the plots are interpreted. To make this distinction clear we will write $\mathcal{R}_{n^*}$ as $\mathcal{R}_*$. and $\mathcal{P}_{n^*}$ as $\mathcal{P}_*$.

To help visualize the distribution of points in Figure 6.5, we show arcs centered at $(1, 1)$ in increments of 0.25 as well as the line $\mathcal{P}_* = \mathcal{R}_*$. Note that by definition $\mathcal{R}_*$ and $\mathcal{P}_*$ evaluate to one when applied to the baseline ranking.

The $(\mathcal{R}_*, \mathcal{P}_*)$ scatterplot is interesting and useful because it allows us to view performance on a query by query basis. It is easy to see that some queries perform far better

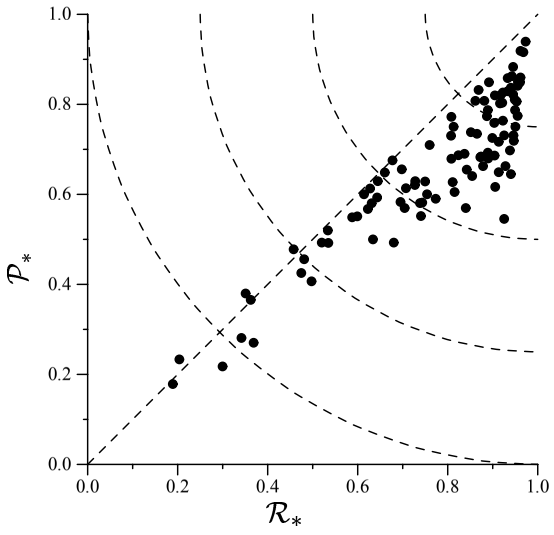


Figure 6.5: $(\mathcal{R}_*, \mathcal{P}_*)$ for *Ideal(0)* compared to RBR.

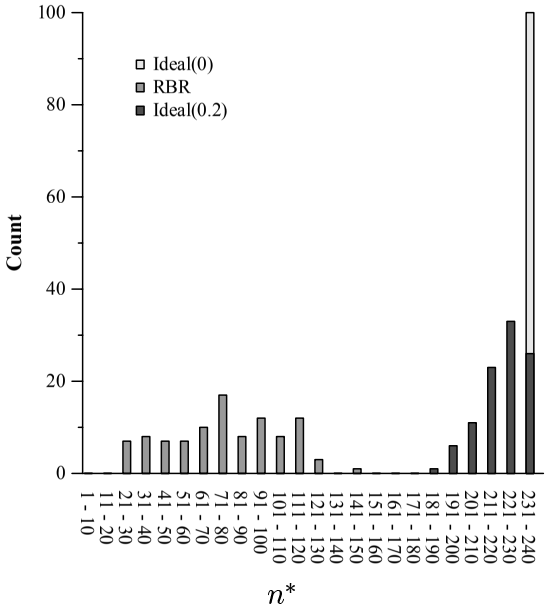


Figure 6.6: Distribution of n^* for *Ideal(l)* and RBR.

than others, something that is obscured in the average plots of Figure 6.4.

Unfortunately, the $(\mathcal{R}_*, \mathcal{P}_*)$ scatterplots can be visually misleading. A histogram of distances from $(1,1)$ shows that 73% of the points in Figure 6.5 lie within a radius of 0.5. $\mathcal{R}_* > .65$ for these points with $\mathcal{P}_* > .5$. These values appear more favorable in the scatterplot than they do in $(n, \mathcal{R}_n)$ and $(n, \mathcal{P}_n)$ plots.

The basic interpretation of $(\mathcal{R}_*, \mathcal{P}_*)$ is "Assuming that there are n^* collections with non-zero merit, how well did the estimator perform when exactly n^* collections are selected?". While this is an interesting question, it is unfortunately vulnerable to differences in baselines and differences in distributions of n^*. Figure 6.6 shows the distribution of n^* values for the RBR, *Ideal(0)* and *Ideal(0.2)* baselines. For *Ideal(0)* and *Ideal(0.2)*, most collections have non-zero merit, virtually guaranteeing that $(\mathcal{R}_*, \mathcal{P}_*)$ scatterplots will cluster around $(1,1)$. Figure 6.6 also shows that even for RBR, n^* ranges from 21 to 150 (9% to 64% of the collections in the *SYM-236* testbed). By the time an estimated rank includes this many collections it can accumulate a significant amount of merit by chance alone.

These factors combine to make $(\mathcal{R}_*, \mathcal{P}_*)$ scatterplots difficult to interpret. At best, they are useful for providing an overview of query-by-query performance. We will rely primarily on average $(n, \mathcal{R}_n)$, $(n, \widehat{\mathcal{R}}_n)$ and $(n, \mathcal{P}_n)$ plots for experiments reported in the body of the dissertation.

6.5 Using *Ideal(0)* to Represent *gGlOSS*

The *gGlOSS* approach includes a variety of parameterized baselines and estimators. In the early experiments presented in this chapter, we have shown that the *gGlOSS* estimators approximate the *gGlOSS* baselines very well. We have shown that both the estimators and baselines have very similar performance when used to approximate the RBR baseline. For the remainder of the dissertation, we will use the *Ideal(0)* baseline to represent *gGlOSS* in comparisons with other collection selection techniques. Here, we motivate that decision. There were two other options, using *Max(l)* or *Sum(l)*, or using *Ideal(l)* for some $l > 0$. We cover the two cases separately.

6.5.1 Why not use *Max(l)* or *Sum(l)*?

An immediate question is why we chose a *gGlOSS* baseline instead of one of the estimators as the representative. After all, it was the estimators that Gravano and García-Molina proposed for actual use. However, recall that *Max(0)* = *Sum(0)* = *Ideal(0)*, that is, at threshold $l = 0$ both estimators and the *Ideal(0)* baseline give identical rankings of the collections for all queries.

Given this equivalence, using the *Ideal(0)* formulation is simpler from an implementation standpoint. *gGlOSS* requires two vectors of information from each collection C_i in order to make its *Max(l)* and *Sum(l)* estimates,

1. the document frequency df_{ij} for each term t_j in C_i; and

2. the sum of the weight of each term t_j over all documents in C_i.

If the underlying collection cannot be made to divulge this information directly, it is in principle still possible to compute the estimates. However, the two vectors of information must be recovered in some way, possibly by issuing a single-term query for each vocabulary term. Our choice of *Ideal(0)* obviates this; if the information required to compute *Ideal(0)* is not readily available, we can compute *Ideal(0)* directly from the collections by simply issuing the test queries.

Finally, when used to approximate the RBR baseline, *Ideal(0)* consistently achieves good performance relative to the *Max(l)* and *Sum(l)* estimators. The upper portion of Figure 6.7 shows the performance of *Ideal(0)* plus *Max(l)* and *Sum(l)* for $l = 0.1$ and $l = 0.2$. The lower portion shows pairwise significance comparisons of the plots for different values of n. For each comparison, the baseline is no significant difference (NSD); the plot shifts to one of the approaches when that approach is significantly better than the approach with which it is paired. For most values of n, there is no significant difference between *Ideal(0)* and the *Max(l)* and *Sum(l)* plots. *Ideal(0)* often significantly outperforms the others, especially for small values of n. Figure 6.7 also reveals that *Sum(l)* rarely outperforms the other approaches.

Given the general performance equivalence and the operational advantages of *Ideal(0)*, we chose *Ideal(0)* over *Max(l)* or *Sum(l)* as the *gGlOSS* representative.

6.5.2 Why not use *Ideal(l)*, $l > 0$?

A secondary question is whether an alternate choice of l for *Ideal(l)* would be more appropriate than *Ideal(0)*. For some experimental environments, this may very well be the case; however, the choice of $l > 0$ is problematic for two reasons.

Figure 6.8 illustrates one difficulty with choosing the value of l. Choosing a value of l that is too high will incorrectly estimate zero merit for many collections, seriously degrading average performance. Note in Figure 6.8 that while *Ideal(0.5)* and *Ideal(1.0)* produce slightly better average performance than *Ideal(0)*, the *Ideal(5.0)* performance is very poor. Using *Ideal(0)* avoids this difficulty.

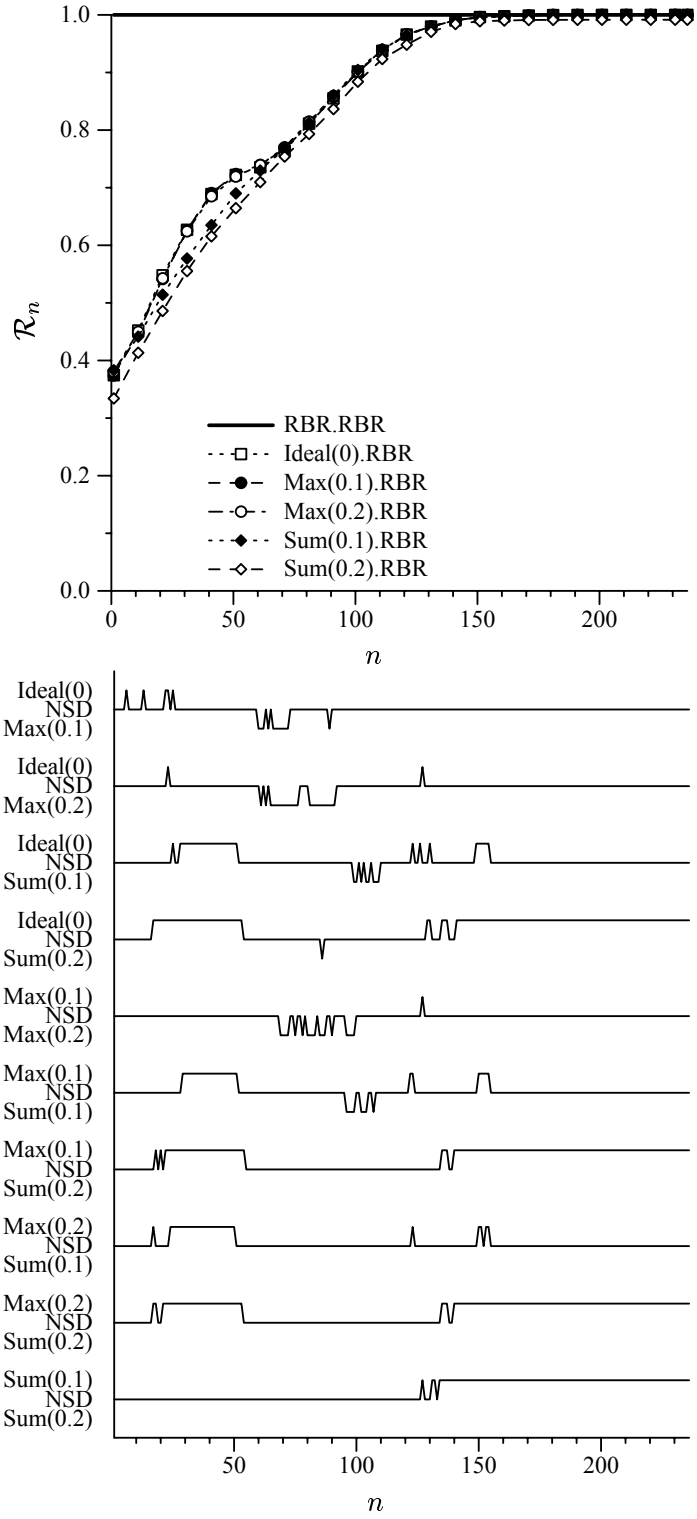


Figure 6.7: Why represent *gGlOSS* with *Ideal(0)* instead of *Max(l)* or *Sum(l)*?

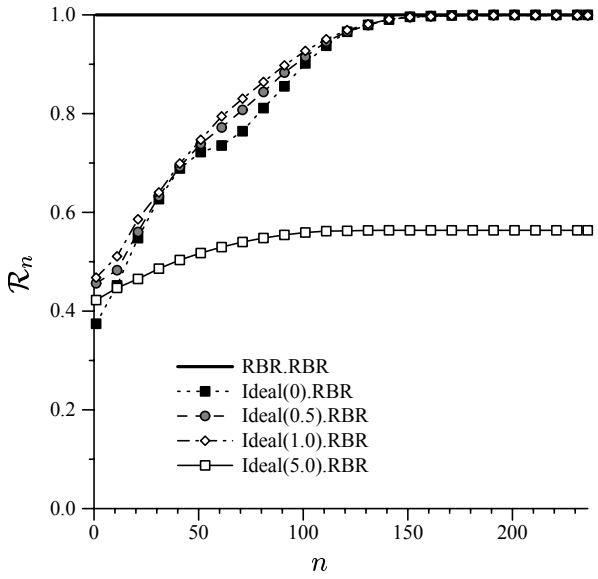


Figure 6.8: The effect of increasing l for *Ideal(l)*.

A second difficulty is that as *gGlOSS* is defined, the value of l is a constant across all collections. When all collections use the same indexing scheme with the same ranges of possible similarity values, the only difficulty is the one noted above—choosing an appropriate value of l. However, in an operational environment it may be the case that the underlying collections use different information retrieval systems that produce differently scaled similarity values (i.e. $sim(q, d)$ in Equation 6.1). Consider collection C_1 with potential similarity values of 1 to 100 and a second collection C_2 with potential similarity values of 1 to 10. A threshold value of $l = 5$ would require that documents have a much higher relative similarity to contribute to the goodness of C_2 than to C_1. This makes ranking collections based on these goodness values difficult.

When $l = 0$ is used as a threshold, all documents with non-zero similarity to the query contribute to the goodness of the collection. This allows a consistent comparison of collections with different underlying retrieval systems. Note that while the comparison is consistent, it may still not be straightforward. Generally speaking, sums of values from 1 to 100 will grow much faster than sums of 1 to 10.

For these reasons, $l = 0$ represents the simplest and generally most reliable choice for a threshold. We use *Ideal(0)* to represent *gGlOSS*.

6.6 A Preliminary Comparison of *Ideal(0)* and *CORI*

Our initial experiments with *gGlOSS*, originally reported in French *et al.* [FPV+98] and summarized above, confirmed that the *gGlOSS Max(l)* and *Sum(l)* estimators approximate the *Ideal(l)* baseline well. We also showed that the *Ideal(0)* baseline is not a very strong predictor of relevance, i.e. the RBR baseline. However, the question of how *gGlOSS* compared to other collection selection algorithms still remained. We began to address that question in French *et al.* [FPC+99b]. We summarize the initial results here; the issue of comparing different collection selection algorithms is covered in much more detail in Chapter 7.

In the results that follow, we report a comparison of *gGlOSS* and *CORI* [CLC95]. As discussed in Section 6.5, we use the baseline *Ideal(0)* to represent *gGlOSS*. A brief description of *CORI* can be found in Appendix A; a more detailed description is given in Chapter 7. For both algorithms, the evaluation was conducted using the full 236 collections of the *SYM-236* testbed. We used the "very long" query formulations of TREC topics 51-150 as the test query set. As a result, these experimental results are directly comparable to those presented in Appendix A. However, the experimental environment differs from those used in Chapters 7–8. While the overall conclusions are comparable, the numerical results may be on different scales.

Our first step in this comparison was the rather involved process of insuring that the two approaches used the same vocabulary when performing the collection selection computation. Minor differences in vocabulary existed due to differences in the underlying indexing technology as defined by Gravano and García-Molina [GGM95] and by Callan *et al.* [CLC95]. The steps we took to achieve a uniform vocabulary are outlined in Appendix A. We also confirmed that the steps taken had very little impact on algorithm

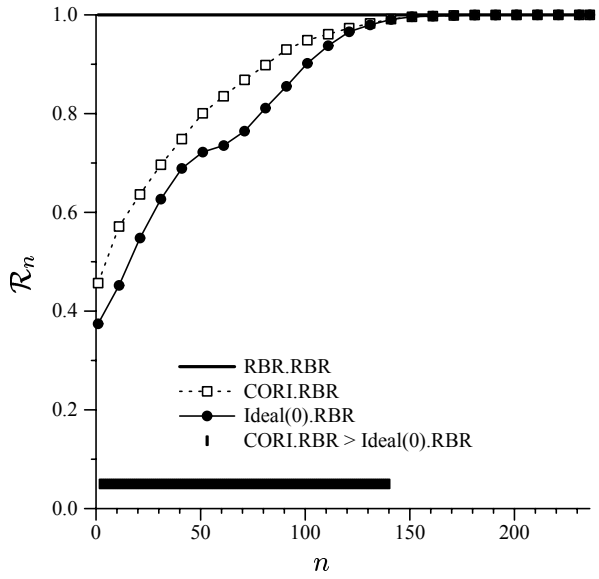


Figure 6.9: Comparing *Ideal(0)* and *CORI* to the RBR baseline.

performance.

Figure 6.9 shows a preliminary comparison of *Ideal(0)* and *CORI* performance approximating the RBR baseline using the $\mathcal{R}_n$ evaluation measure. The plot labeled *Ideal(0).RBR* is exactly the same as plots with the same label in the $\mathcal{R}_n$ graph of Figure 6.4 and with the plot *Ideal(0)-smart.RBR* of Figure A.3 in Appendix A. The plot labeled *CORI.RBR* is the same as *CORI-UVA.RBR* in Figure A.4.

Figure 6.9 reveals that on average *CORI* visibly outperforms *Ideal(0)* for n less than approximately 120. We performed a paired Wilcoxon significance test for $p = 0.05$. Because only two approaches were compared, significance is shown on the same graph. Values of n for which *CORI* significantly outperforms *gGlOSS* are marked at the bottom of the graph. Both approaches did equally well identifying the special case collections mentioned in Section 6.4.2. In Chapter 7, we examine potential reasons for the difference in performance, including an *Ideal(0)* preference for collections with a large number of documents.

One thing to note is that while *CORI* outperforms *gGlOSS*, there is still a lot of room for improvement, especially for small values of n.

7

Collection Selection — Comparative Experiments

This chapter presents a comparative evaluation of three collection selection approaches, *CORI* [CLC95], *gGlOSS* [GGM95, GGMT99] and *CVV* [YL97]. When they were proposed, these approaches were independently evaluated; however, the evaluations were conducted using a variety of test environments. It was not possible to compare these algorithms reliably based solely on the published evaluations. Prior to our early comparative experiments [FPV+98, FPC+99b], there had been only one extremely limited comparison of these approaches[1]. This chapter represents an expansion of our early comparative experiments.

Here, we present a direct comparison of the *CORI*, *gGlOSS* and *CVV* approaches using the *SYM-236*, *UDC-236* and *UBC-100* testbeds and both the title and long query formulations of TREC topics 51-150. For this comparison, we are concerned only with the *collection selection* performance of these approaches. The impact that collection selection can have on data item retrieval in a multi-collection environment will be covered in Chapter 9. Our goal for these experiments was to perform a direct comparison of the approaches in a variety of test environments to determine relative performance, the effect of test environment on performance, and whether the additional statistical information used by *gGlOSS* is beneficial.

In this chapter we will first describe the *CORI* and *CVV* collection selection approaches.

[1] Yuwono and Lee [YL97] compared the ability of *CORI*, *gGlOSS* and *CVV* to identify collections with highly similar documents using a very small test environment.

gGlOSS was covered in detail in Chapter 6. We then describe the test environments and cover our experimental setup. We will present results of our direct comparison of the three approaches using the $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ measures defined in Chapter 5. Finally, we will discuss the correlation of these approaches with the SBR baseline and compare the performance of these approaches to the expected performance of random selection.

7.1 Approaches Considered

We compare the performance of three collection selection approaches in a variety of test environments. Here we provide the details of the collection selection algorithms; summaries of the original experiments performed by the researchers who proposed these techniques are provided in Chapter 2.

7.1.1 *gGlOSS*

gGlOSS [GGM95, GGMT99] was described and examined in detail in Chapter 6. Here, we summarize the *Ideal(0)* baseline that we use as a representative for *gGlOSS* in all remaining experiments. The choice of *Ideal(0)* as a representative for *gGlOSS* was covered in Section 6.5.

From the original definition of *Ideal(l)*, *Ideal(0)* can be computed by sorting collections in decreasing order of Goodness when $l = 0$ (from Equation 6.1).

$$Goodness(0, q, C_i) = \sum_{\{d \in C_i | sim(q,d) > 0\}} sim(q, d)$$

Because *Max(0)* = *Sum(0)* = *Ideal(0)*, *Ideal(0)* can also be computed using only the W matrix required by *gGlOSS*, using the following computation,

$$Estimate(0, q, C_i) = \sum_{j=1}^{n} q_j \times w_{ij}$$

repeated here from Equation 6.4.

7.1.2 *CORI*

Given a set of collections to search, the *CORI* collection selection approach creates a *collection selection index* in which each collection is represented by its terms and their document frequencies *df*. Collections are ranked for a query q by a variant of the Inquery document ranking algorithm. The belief $p(r_k|C_i)$ in collection C_i due to observing query term r_k is determined by:

$$\begin{aligned} T &= \frac{df}{df + 50 + 150 \cdot cw/\overline{cw}} \\ I &= \frac{\log\left(\frac{N+0.5}{cf}\right)}{\log\left(N + 1.0\right)} \\ p(r_k|C_i) &= 0.4 + 0.6 \cdot T \cdot I \end{aligned} \tag{7.1}$$

where:

- df is the number of documents in C_i containing r_k,
- cf is the number of collections containing r_k,
- N is the number of collections being ranked,
- cw is the number of words in C_i, and
- $\overline{cw}$ is the mean cw of the collections being ranked.

In the general case, the belief in a collection depends upon the query structure; for our experiments it is the average of the $p(r_k|C_i)$ values for each query term [CLC95].

The constants found in the formulation of T in Equation 7.1 are explained by a parametric study reported in Callan *et al.* [CLC95]. T was originally defined as

$$T = d_t + (1 - d_t) \cdot \frac{df}{df + K}$$

where $K = k \cdot ((1 - b) + b \cdot cw/\overline{cw})$ and k and b are constants. A parametric study [CLC95] found that for the 17 collection test environment used the best combination of values was

$k = 200$, $b = 0.75$. For those values and $d_t = 0$, the original definition of T yields the constants found in Equation 7.1. We use those constants for our experiments.

The *CORI* approach to ranking collections can be summarized as $df.icf$, where icf is inverse collection frequency. Given a set of collections to search, the *CORI collection selection index* essentially represents each collection as a virtual document made up of a list of terms and their document frequencies in the underlying collections. The virtual documents are indexed by the Inquery information retrieval system[CCH92]. A query q is applied to the collection selection index to rank the virtual documents. The resulting virtual document ranking is the *CORI* ranking of the collections.

7.1.3 CVV

Yuwono and Lee [YL97] proposed an approach to the broader problem of distributed search, considering collection selection, query forwarding and results merging. They referred to the collection selection portion of their work as the *Cue Validity Variance (CVV)* ranking method. *CVV* refers both to the ranking method and to a component in their calculation of a collection's estimated merit or score.

The *CVV* approach is based loosely on the concept of *cue validity* as used in cognitive science. A summary description of cue validity can be found in Smith and Medin [SM81]. In short, it has been observed that a feature F_i is useful for categorizing a concept X_j if the feature is not also associated with a contrasting concept. The traditional definition of cue validity of F_i for X_j is

$$P(X_j|F_i) = \frac{P(F_i|X_j)}{P(F_i|X_j) + P(F_i|X_k)}$$

where X_k is some contrasting concept. The cue validity increases with the probability that feature F_i describes concept X_j but decreases if F_i is also representative of some other concept X_k. Yuwono and Lee use a variant of the traditional cue validity definition, suggested by Goldberg [Gol96]. In this variant a concept is contrasted with the union of

all other concepts.

The CVV ranking method employs a combination of document frequency (df) information and cue validity variance. Cue validity variance, as defined by Yuwono and Lee [YL97], attempts to characterize the distribution of the density of df values, i.e., the variability of the fraction of documents in a collection that contain a given term. Document frequency information is used to approximate how important a term is within a collection; the goal of the CVV component is to estimate whether a term is useful for differentiating one collection from another.

The CVV estimated merit computation is summarized below. Note that there are some notational differences between our summary and the version presented by Yuwono and Lee [YL97]. We have modified the notation to be conformant with the notation used in the rest of this dissertation.

$$est\_merit(C_i, q) = \sum_{\{t_j \in q\}} CVV_j \cdot df_{ij} \tag{7.2}$$

where t_j is a term in query q, df_{ij} is the document frequency of t_j in collection C_i, and

$$CVV_j = \frac{\sum_{i=1}^{N} (CV_{ij} - \overline{CV_j})^2)}{N}$$

where

$$CV_{i,j} = \frac{\frac{df_{ij}}{|C_i|}}{\frac{df_{ij}}{|C_i|} + \frac{\sum_{k \neq i}^{N} df_{kj}}{\sum_{k \neq i}^{N} |C_k|}}$$

and

$$\overline{CV_j} = \frac{\sum_{i=1}^{N} CV_{ij}}{N}$$

The CVV ranking method uses only information from (or derivable from) the matrix F used by $gGlOSS$. The goal of the CVV merit estimation method is to identify collections

with a high concentration of query terms.

7.2 Test Environments and Experimental Setup

We used a six test environments for the experiments reported in this chapter. As we mentioned earlier, we used the *SYM-236*, *UDC-236* and *UBC-100* testbeds and both the long and short query formulations of TREC topics 51-150. The six test environments can be specified as:

- ($SYM\text{-}236$, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$), ($SYM\text{-}236$, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$),
- ($UDC\text{-}236$, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$), ($UDC\text{-}236$, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$),
- ($UBC\text{-}100$, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$) and ($UBC\text{-}100$, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$).

Note that the relevance judgements for all six test collections are those provided with the TREC-4 data. When describing the results, we will refer to the test environments using testbed and the query set names. We find that collection selection results vary more on a testbed-by-testbed basis than on a query set basis. Therefore, we present results by testbed, with long and short query results displayed side-by-side for easier comparison.

For each test environment and all experiments, the default configurations of $\mathcal{LM}$ and $R_{\mathcal{LM}\rightarrow\mathcal{C}}$ apply. Specifically, for each test environment, LM_i is constructed using all data items from C_i and the merit computed using LM_i is applied to collection C_i.

We prepared each collection in each testbed using version 11.0 of the SMART information retrieval system [Buc92]. Statistical information extracted from the SMART indexes is used to create the F and W matrices that are the input to *gGlOSS*. In Appendix A we discuss the steps that were taken to ensure a fair comparison between *gGlOSS* and *CORI* in Chapter 6. To ensure a fair comparison here, the document frequency (*df*) information required by *CORI* and *CVV* was taken from the F matrix used by *gGlOSS*'s *Ideal(0)*. To facilitate the use of external *df* information, we use the UVA implementation of *CORI*

described in Appendix A. The versions of *Ideal(0)* and *CVV* that we use are also UVA implementations of the published algorithms.

Queries were processed using SMART to convert them into term lists compatible with the vocabularies used by the collection selection approaches. However, SMART was not used for the implementations of the collection selection approaches or for actually issuing the queries.

For these experiments, we are concerned with the ability of these collection selection approaches to locate collections containing relevant documents. As a result, we use the RBR baseline for comparison. We also utilize the SBR baseline to illustrate a simple lower bound on performance.

7.3 Correlation with the SBR Baseline

Before we present the results of our comparative experiments, we need to point out a feature that the three collection selection approaches share to varying degrees. We previously introduced the SBR baseline, which orders collections simply in decreasing order of the number of documents in the collection. We presented SBR as a simple heuristic that could also serve as a useful lower bound on performance. Over the course of our experiments we found that SBR is also useful for explaining a feature of collection selection algorithm performance.

Our interest in SBR started with a detailed analysis of the *Ideal(0)* results shown in Chapter 6. We were attempting to isolate the cause of the observed performance difference between *Ideal(0)* and *CORI*[2]. A detailed examination of the collection rankings revealed that many of the same collections were highly ranked by *Ideal(0)* for a startling number of the queries. Further examination revealed that the highly-ranked collections tended to be the collections with the largest number of documents. The SBR baseline provides a means to examine the tendency of *Ideal(0)* and other algorithms to prefer larger collections. Plotting SBR.RBR on Figure 6.9 to produce Figure 7.1 makes a circumstantial case for a correlation

[2]Recall that these experiments used the very long query formulations.

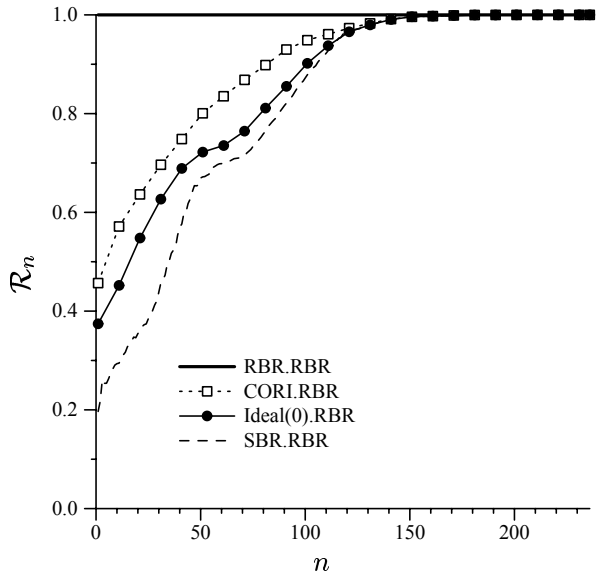


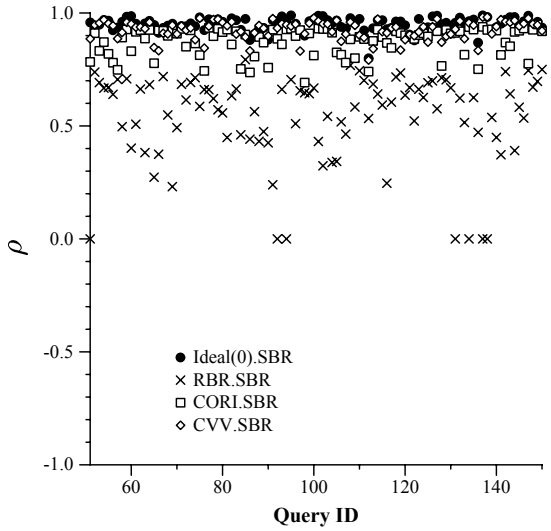
Figure 7.1: Comparing *Ideal(0)*, *CORI* and SBR to the RBR baseline.

between *Ideal(0)* and SBR. Note that because Figure 6.9 reports early experimental results, the very long query set was used here.

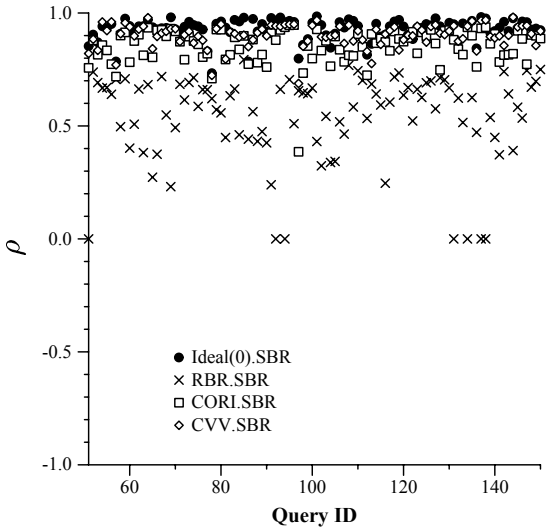
Turning our attention to our current set of six experimental environments, we use Spearman's ρ as defined in Chapter 5 to measure the correlation between the SBR rankings and the rankings produced by RBR and the three collection selection approaches. Figure 7.2 presents the Spearman correlation coefficient values for the short and long queries for the *SYM-236* and *UBC-100* testbeds[3]. Each graph of Figure 7.2 is a scatterplot of ρ values for a testbed, query set pair. The values of ρ are computed for each of the 100 short queries and each of the 100 long queries and presented as a scatterplot. Each point represents a query under the labeled approach.

First, consider the Spearman values for the *SYM-236* testbed, the testbed for which a correlation between *Ideal(0)* and SBR was originally suspected. The correlations between

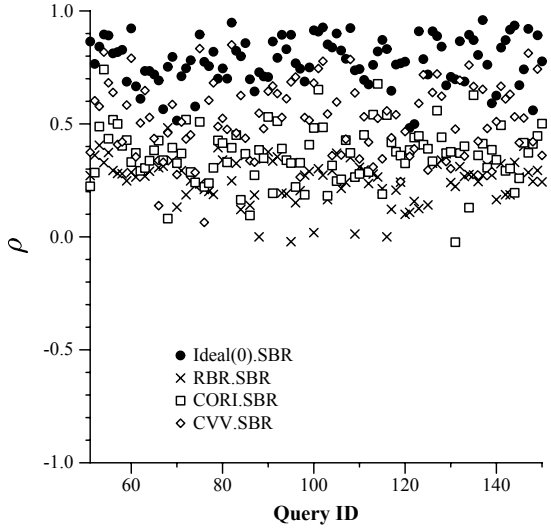
[3]Because the *UDC-236* testbed was constructed to contain collections with roughly the same number of documents, the SBR baseline is not applicable for this testbed. In fact, SBR essentially devolves to alphabetical order (our tiebreaker) and comparisons with it are not illuminating. There are no correlations with SBR for Figure 7.2 and no SBR.RBR plot presented for any of the later *UDC-236* testbed results.



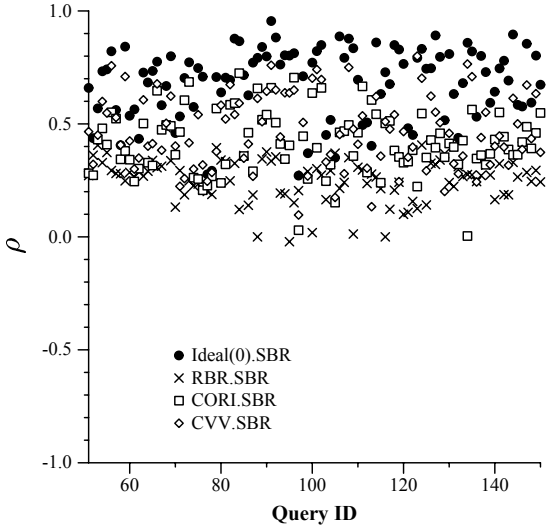
SYM-236 testbed, long queries



SYM-236 testbed, short queries



UBC-100 testbed, long queries



UBC-100 testbed, short queries

Figure 7.2: Spearman correlation of selection approaches with SBR baseline.

Ideal(0) and SBR and between *CVV* and SBR are very strong for both long and short queries. The correlation between *CORI* and SBR is still positive but not as dramatic. For later reference, note that correlation between RBR and SBR is positive, suggesting that for many queries, collections containing a large number of documents also tend to contain relevant documents. The correlations between all of the approaches and SBR for the *UBC-100* testbed are not as pronounced, but are still positive on the whole.

We note that a preference for large collections is not necessarily a liability. If large collections tend to have high merit, selection based on this heuristic and selection approaches correlated with collection size can be effective. However, SBR and approaches highly correlated with it will generally perform poorly if the largest collections have little or no merit.

7.4 Results

This section presents the initial results comparing the *CORI*, *CVV* and *gGlOSS* collection selection approaches in the test environments described above. We consider the results on a testbed by testbed basis, using the $\widehat{\mathcal{R}}_n$, $\mathcal{R}_n$ and $\mathcal{P}_n$ measures together in an attempt to present a comprehensive view of performance. We will discuss the results for each testbed individually, drawing out features of the testbeds that affect performance. In a later section, we will discuss comparisons with the expected performance of $\widehat{\mathcal{R}}_n$, $\mathcal{R}_n$ and $\mathcal{P}_n$ for random selection. Finally, in Section 7.6, we will present a more unified discussion of the overall results.

7.4.1 *SYM-236*

The first two test environments we consider utilize the *SYM-236* testbed and both short and long queries. These test environments can be specified as (*SYM-236*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$) and (*SYM-236*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$). The collection selection evaluation results for these environments are presented in Figure 7.3. Figure 7.3 contains six graphs and presents results from two

test environments using three different evaluation measures. Each graph is labeled with the testbed, evaluation measure and query set it represents. The results for test environments utilizing the *UDC-236* and *UBC-100* testbeds will follow the same organizational approach.

The first thing to note is that the *SYM-236* testbed is the same testbed used for the experiments reported in Chapter 6. The difference between these graphs and the graphs reported in Chapter 6 is that different query formulations were used. While the query formulations are different, the distribution of relevant documents is the same. As we discussed in Chapter 6, the *SYM-236* testbed contains a number of collections that have no merit for most queries. The "knee" in the RBR.RBR plots on the $\widehat{\mathcal{R}}_n$ graphs represents the transition from high- and medium-merit collections to very low-merit collections. The plateaus in the plots for *CORI*, *CVV*, *Ideal(0)* and SBR at approximately $n = 140$ show that all of these approaches do a similarly good job at placing the very low-merit collections at the bottom of their rankings.

The steep climb of the RBR.RBR curve under the $\widehat{\mathcal{R}}_n$ evaluation measure reveals that for most queries, relevant documents are not evenly distributed across the collections in the *SYM-236* testbed. In other words, for many queries, there are a few collections with a large number of relevant documents as well as collections with no relevant documents. Any approach that is able to correctly identify the collections with a large number of relevant documents will perform very well under the $\widehat{\mathcal{R}}_n$ and $\mathcal{R}_n$ measures. Conversely, collection selection approaches that do not identify high-merit collections will perform poorly. One of the hallmarks of the *SYM-236* testbed is that small perturbations in ranks can have larger effects under our evaluation measures.

The performance of *CORI*, *CVV* and *Ideal(0)* is best revealed by examining the $\mathcal{R}_n$ and $\mathcal{P}_n$ graphs together. The graphs reveal that, especially for $n < 50$, while the approaches tend to identify collections with some relevant documents (revealed by the $\mathcal{P}_n$ measure), they do not identify the collections with the *most* relevant documents (revealed by the $\mathcal{R}_n$ measure).

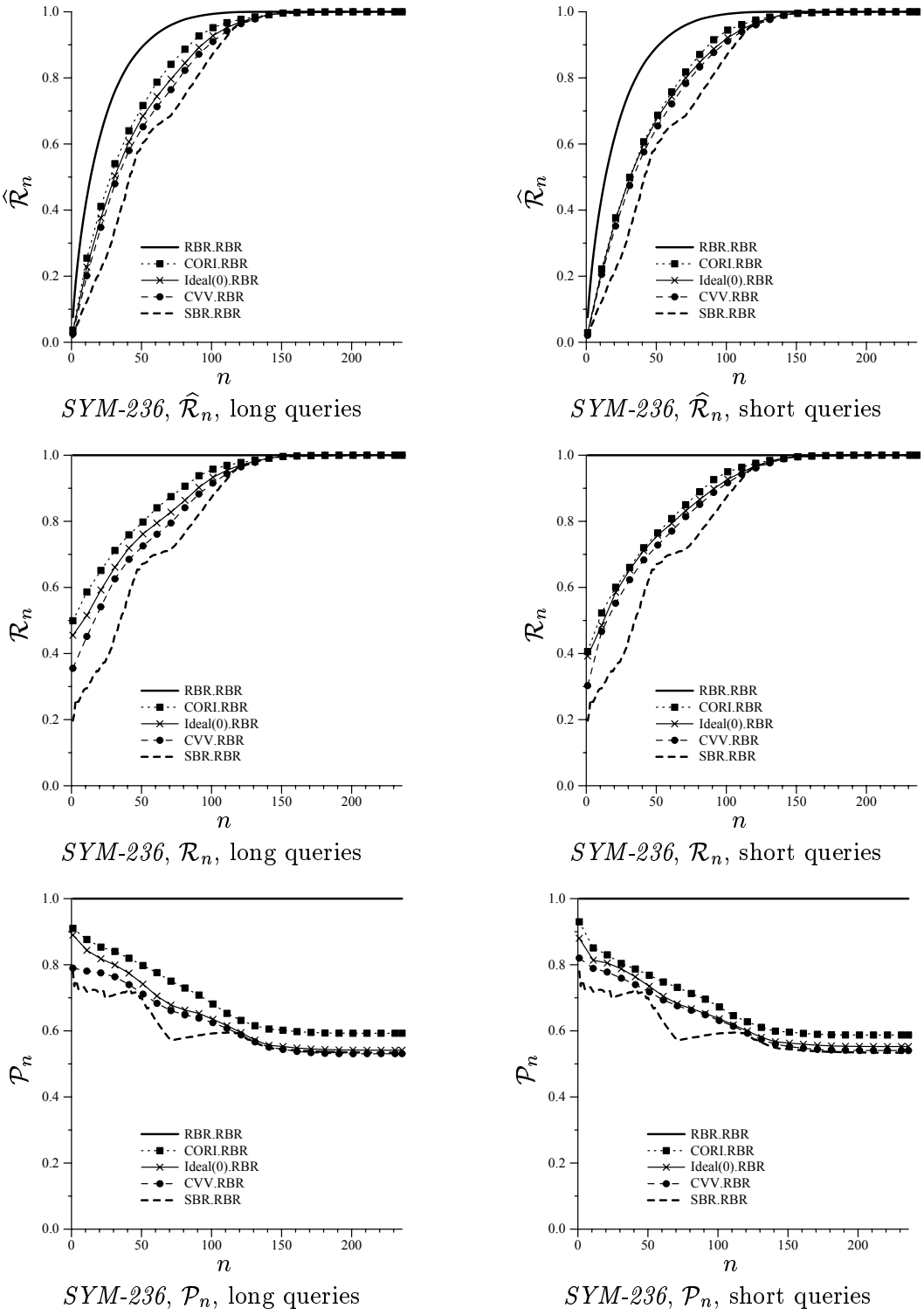
Comparisons between the performance of the approaches using the two different query

formulations are facilitated by the RBR.RBR and SBR.RBR curves. These curves are not affected by the query formulation and are the same for each pair of $\widehat{\mathcal{R}}_n$, $\mathcal{R}_n$ and $\mathcal{P}_n$ graphs. Overall, we note that the relative performance of the three approaches is similar under both the long and short query formulations, but that the overall performance of the approaches is slightly worse for the short queries. The differences among the approaches are smaller under the short queries; however, on average, *CORI* appears to be affected more by the short query effect.

Overall, the performance curves for *CORI*, *Ideal(0)* and *CVV* appear to be similar under all three evaluation measures. The tendency of the plots to follow a similar path can obscure the difference between the curves; the differences between the curves need to be observed vertically. The vertical differences between the points are greater than they appear when only the track of the curves is considered. For example, consider the *SYM-236*, $\mathcal{R}_n$, long queries graph of Figure 7.3. Averaged over the values $1 \leq n \leq 50$, we find that *CORI* performs 10% better than *Ideal(0)* and 20% better than *CVV*.

Figure 7.4 presents a pairwise comparison of approaches using a paired Wilcoxon (p=0.05) significance test for the evaluation results under the $\mathcal{R}_n$ measure. Figure 7.4 reveals that the pairwise differences between the approaches are significant for most values of n less than approximately 140. For the *SYM-236* testbed, all approaches are equally effective at placing the very low-merit collections (which also contain a very small number of documents) at the bottom of the rankings. As a result, there is little visible (or significant) difference between the approaches for n greater than approximately 140. We also note that the drop in the performance of *CORI* for the short queries results in no significant difference between *CORI* and *Ideal(0)* for some of the lower values of n.

One additional thing to note about Figure 7.3 is the slight shift in SBR performance for $50 < n < 70$. This is due to the presence of a consecutively-ranked group of ZIFF collections. Note in Figure 4.3 that the ZIFF collections contain a fairly large number of documents. However, there are relatively few queries for which these documents are relevant. These large, but often non-relevant collections adversely affect the performance



SYM-236, $\widehat{\mathcal{R}}_n$, long queries

SYM-236, $\widehat{\mathcal{R}}_n$, short queries

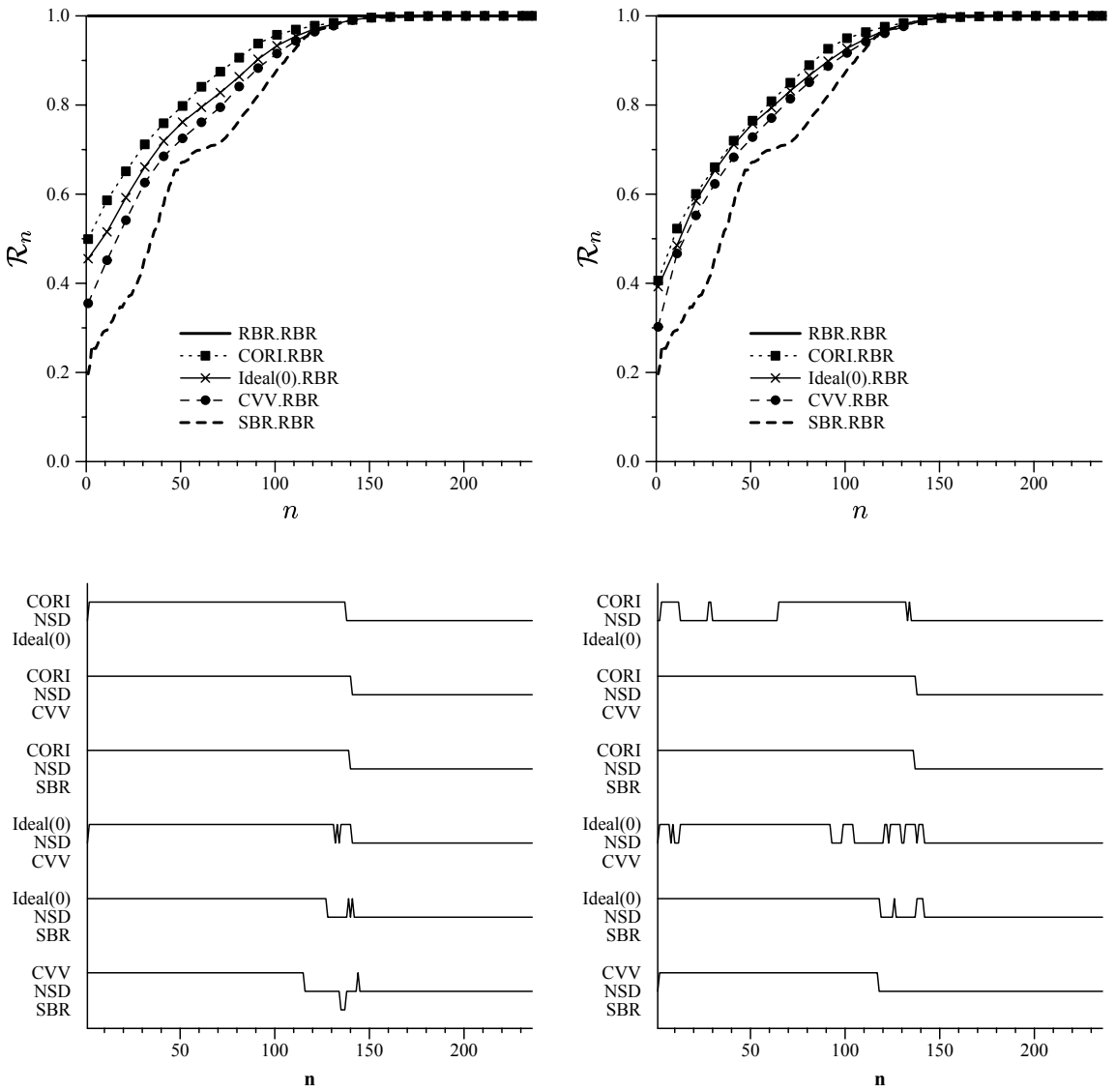
SYM-236, $\mathcal{R}_n$, long queries

SYM-236, $\mathcal{R}_n$, short queries

SYM-236, $\mathcal{P}_n$, long queries

SYM-236, $\mathcal{P}_n$, short queries

Figure 7.3: Collection selection results for the *SYM-236* testbed, long and short queries, $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ evaluation measures.



SYM-236, long queries

SYM-236, short queries

Figure 7.4: Comparison of approaches using the $\mathcal{R}_n$ measure, *SYM-236* testbed, long and short queries, plus significance of comparison.

of SBR.

A comparison of Figure 7.3 and Figure 7.1 leads to an obvious question. In Section 7.3 we noted a pronounced visible "tracking" of SBR by *Ideal(0)* for very long queries. However, that tracking is much less evident for the long queries and not seen for the short queries. A detailed examination of the placement of ZIFF collections within the collection rankings reveals that for many very long queries ZIFF collections are ranked within the $50 < n < 70$ range, as they are for SBR. This behavior is still evident but less common for the short and long queries. While Figure 7.2 shows that *Ideal(0)* is still very highly positively correlated with SBR for short and long queries, the ZIFF collections are less tightly grouped, lessening the visible effect.

7.4.2 *UDC-236*

The next pair of test environments that we consider utilize the *UDC-236* testbed and both long and short queries. These test environments can be specified as (*UDC-236*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$) and (*UDC-236*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$). Results are presented in Figure 7.5.

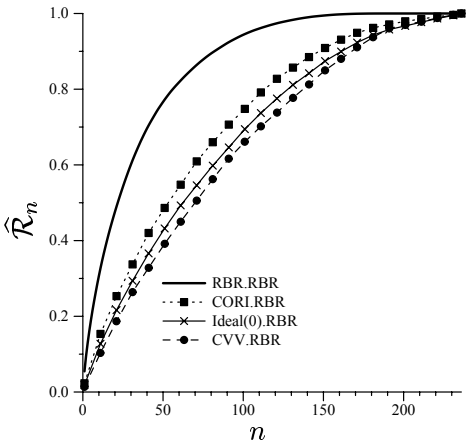
Recall that the *UDC-236* testbed uses the same document set $\mathcal{D}$ and the same number of collections N as the *SYM-236* testbed, the documents are merely organized into collections differently. *UDC-236* was designed so that the number of documents per collection is roughly uniform. A related feature, although a result that was not explicitly designed into the testbed, is that the number of relevant documents per collection is more uniform for *UDC-236* than for *SYM-236*. As a result, in Figure 7.5 we don't see the pronounced plateaus in the performance curves for *CORI*, *CVV* and *Ideal(0)* that we saw for the *SYM-236* testbed. However, the shape of the RBR.RBR curve under the $\widehat{\mathcal{R}}_n$ evaluation measure reveals that for most queries there are collections for which some query has no relevant documents. The RBR.RBR curve is not as steep as that seen in the *SYM-236* testbed, however it is still much steeper than the curves for the three collection selection approaches. Similar to what we observed for *SYM-236*, *UDC-236* contains collections with large numbers of relevant documents with respect to the queries. The differences

in relevant documents per collection simply are not as pronounced as they were in the *SYM-236* testbed.

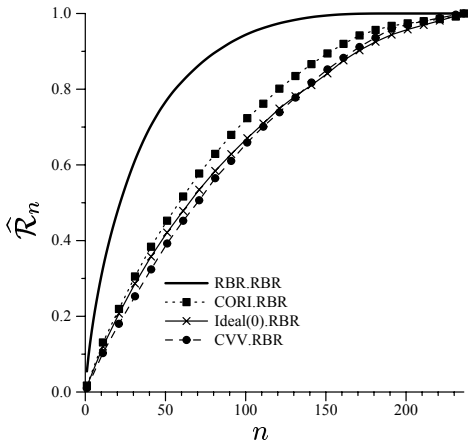
The $\mathcal{R}_n$ and $\mathcal{P}_n$ graphs of Figure 7.5 reveal comparative performance of *CORI*, *CVV* and *Ideal(0)* that is similar to that we saw for the *SYM-236* testbed. All three approaches tend to identify collections with some relevant documents but not the collections containing the highest number of relevant documents. Comparisons between the performance of the approaches using the two different query formulations are more difficult because of the lack of an SBR.RBR curve. The difference between the performance of short and long queries is most obvious for the $\mathcal{R}_n$ measure. Overall, we note that the relative performance of the three approaches is similar under both the long and short query formulations, but that the overall performance is slightly worse for the short queries. The differences among the approaches are smaller under the short queries.

Figure 7.6 presents a pairwise comparison of the approaches using a paired Wilcoxon (p=0.05) significance test for the evaluation results under the $\mathcal{R}_n$ measure. For most values of n, the differences between the three approaches are significant for the long queries. For the short queries, there is no significant difference more often, most notably between *CORI* and *Ideal(0)* for very small values of n. However, for most comparisons and most values of n the pairwise differences are significant. It is interesting to note that the *CVV* and *Ideal(0)* performance curves for short queries cross over at approximately $n = 140$ and *CVV* becomes significantly better at approximately $n = 180$. However, this is unlikely to have a large effect due to the very high value of n.

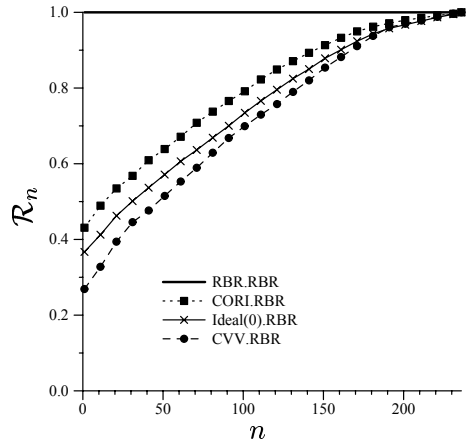
One final thing to note about Figure 7.5 is that *UDC-236* is a more difficult testbed than *SYM-236*. As we noted earlier, the relevant documents are more evenly distributed. While the performance curves for all three approaches are similar to those seen for *SYM-236* the overall values under the performance measures are lower.



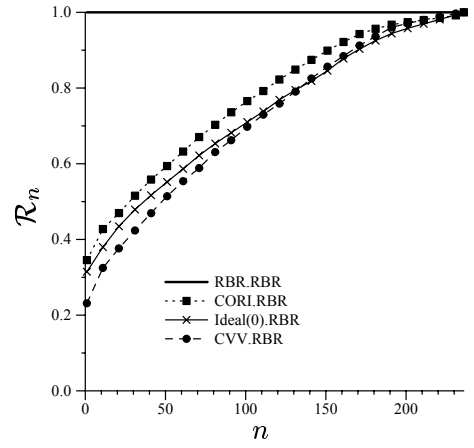
UDC-236, $\widehat{\mathcal{R}}_n$, long queries



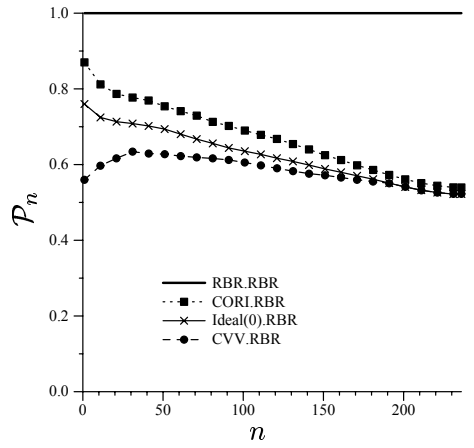
UDC-236, $\widehat{\mathcal{R}}_n$, short queries



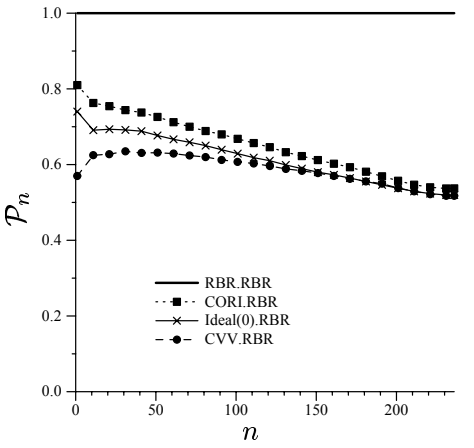
UDC-236, $\mathcal{R}_n$, long queries



UDC-236, $\mathcal{R}_n$, short queries

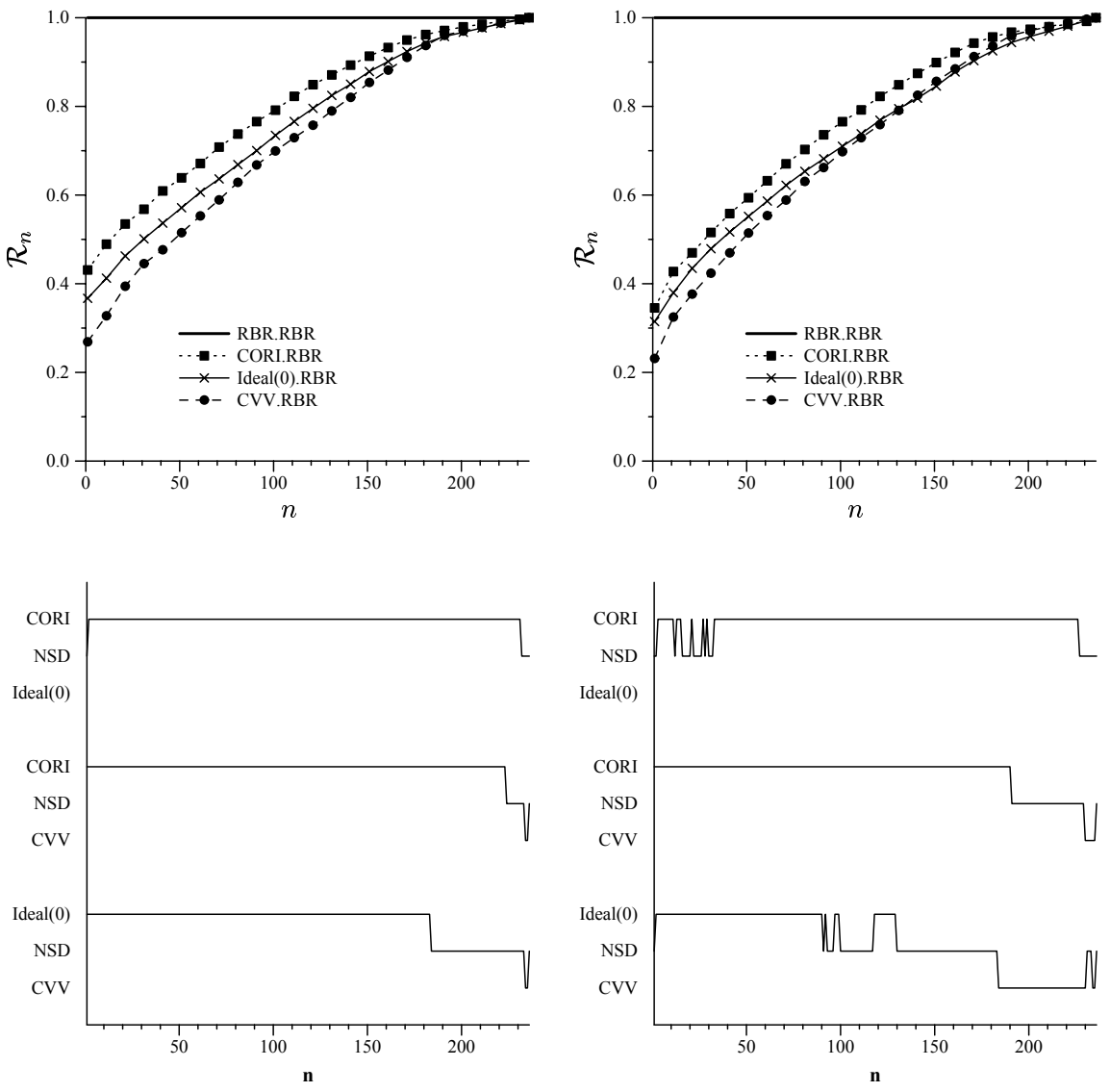


UDC-236, $\mathcal{P}_n$, long queries



UDC-236, $\mathcal{P}_n$, short queries

Figure 7.5: Collection selection results for the *UDC-236* testbed, long and short queries, $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ evaluation measures.



UDC-236, long queries

UDC-236, short queries

Figure 7.6: Comparison of approaches using the $\mathcal{R}_n$ measure, *UDC-236* testbed, long and short queries, plus significance of comparison.

7.4.3 *UBC-100*

The third pair of test environments that we consider utilize the *UBC-100* testbed and both long and short queries. These test environments can be specified as (*UBC-100*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$) and (*UBC-100*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$). The results for these test environments using the $\widehat{\mathcal{R}}_n$, $\mathcal{R}_n$ and $\mathcal{P}_n$ evaluation measures are presented in Figure 7.7.

When examining the results in Figure 7.7, recall that there are a few differences between the *UBC-100* testbed and the *SYM-236* and *UDC-236* testbeds. The *UBC-100* testbed contains more documents spread over fewer collections. Despite these differences, the shape of the RBR.RBR curve under the $\widehat{\mathcal{R}}_n$ measure is similar to that seen for the *UDC-236* testbed.

The most striking thing about the results of Figure 7.7 is the performance of SBR and *Ideal(0)*. The performance of SBR is much poorer than that seen for the *SYM-236* testbed—while SBR tracked the other approaches closely in *SYM-236*, the performance in *UBC-100* is obviously much poorer. The $\mathcal{P}_n$ graphs provide a first clue as to the cause of the poor performance. The $\mathcal{P}_n$ graphs reveal that for very small values of n, SBR chooses collections with zero merit approximately 80% of the time. The source of this difficulty is illustrated by Figure 4.5. Six DOE and two ZIFF collections are far larger than the other collections but contain relevant documents for few queries[4].

While *Ideal(0)* is somewhat less correlated with SBR in this testbed than in *SYM-236* (see Figure 7.2), that correlation is enough to adversely affect performance. For the *UBC-100* testbed, *Ideal(0)* performs significantly more poorly than *CVV* whereas in *SYM-236* and *UDC-236*, *Ideal(0)* performs better than *CVV*. Under the evaluation measures that we use, very poor early performance can have ramifications for values of n other than those for which poor selection occurred. The *UBC-100* testbed reveals the potential downfall of collection selection approaches that prefer collections with a large number of documents. The presence of large collections with few relevant documents will often confound these

[4]Recall that ZIFF collections as a whole proved problematic for SBR within the *SYM-236* testbed. For *UBC-100*, the other ZIFF collections are sufficiently scattered through the SBR ranking that they have no visible impact.

approaches.

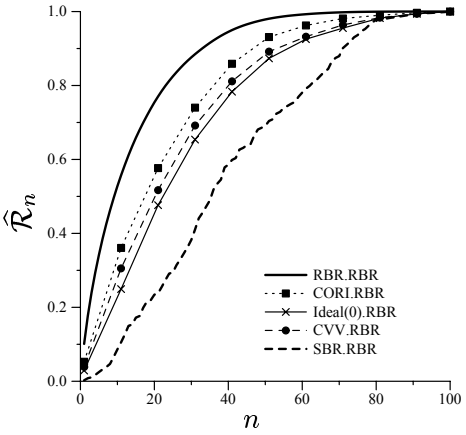
It is interesting to note that with the exception of *Ideal(0)* and SBR, the overall performance trends presented in Figure 7.7 are similar to those seen for the *SYM-236* and *UDC-236* testbeds. The pairwise Wilcoxon (p=0.05) significance tests presented in Figure 7.8 reveal that for most values of n, the differences between the three approaches are significant for the long and short queries.

7.5 Comparison with Performance of Random Selection

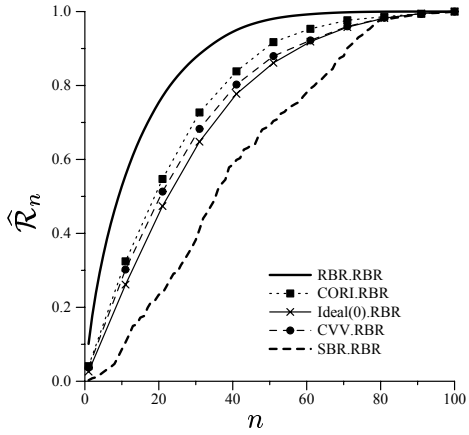
The graphs of Figure 7.9 show the expected performance of random selection under the $\mathcal{R}_n$ and $\mathcal{P}_n$ measures. $E[\widehat{\mathcal{R}}_n]$ is defined as n/N and as a result is the same for all testbeds (the line $x = y$), so is not plotted here. It is interesting to note that the $\widehat{\mathcal{R}}_n$ performance observed by Hawking and Thistlewaite [HT99] when collections were experimentally selected at random is very close to the expected values that we predict.

Both evaluation measures are plotted for the *SYM-236*, *UDC-236* and *UBC-100* testbeds. The short query results first shown in Figures 7.3, 7.5 and 7.7 are included for reference. $\mathcal{P}_n$ is the only measure for which a substantial difference can be seen between the testbeds. This is due to the distribution of collections containing at least one relevant document for queries. The more skewed distribution of relevant documents for *SYM-236* makes it more difficult to select a collection with non-zero merit at random, reducing the expected $\mathcal{P}_n$ performance.

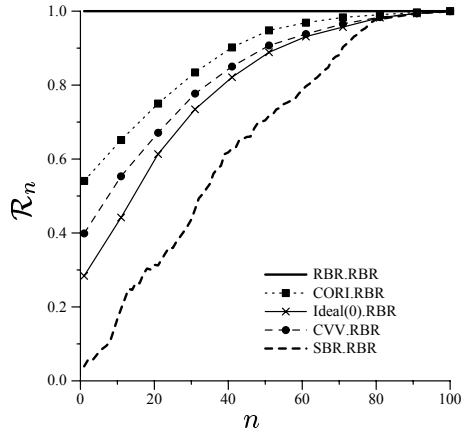
With the exception of SBR selection for testbed *UBC-100*, all collection selection approaches tend to perform much better than random selection when performance over 100 queries is averaged. However, it is also illuminating to consider performance on a query-by-query basis. Using the same data that was used to generate Figure 7.9, we compared the *short query performance* of the three collection selection algorithms to the expected value of $\mathcal{R}_n$ given randomly generated rankings. We chose the short query results because they tended to perform slightly worse than long query results. For each n, Table 7.1 records the



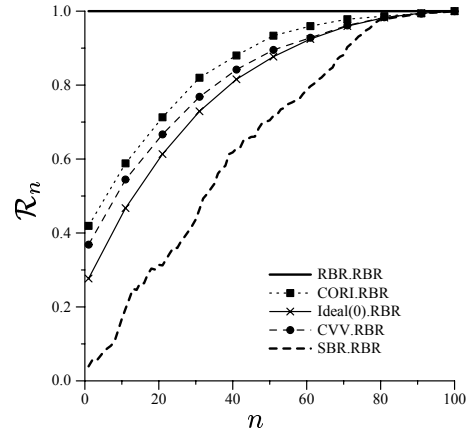
UBC-100, $\widehat{\mathcal{R}}_n$, long queries



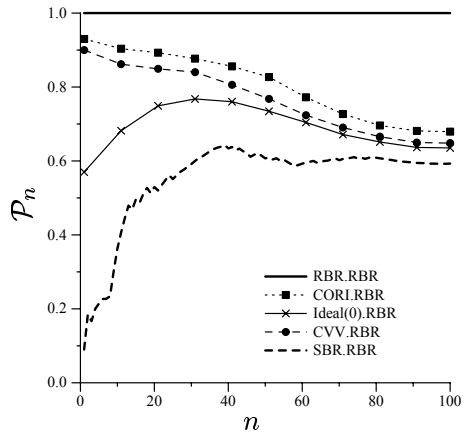
UBC-100, $\widehat{\mathcal{R}}_n$, short queries



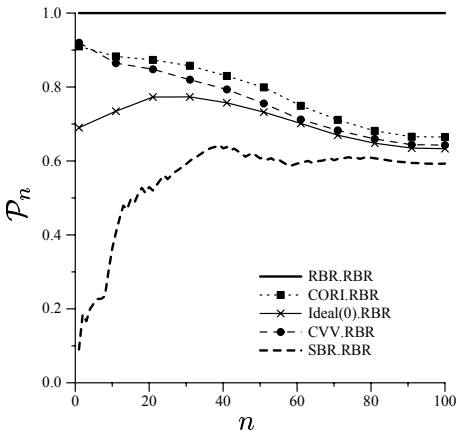
UBC-100, $\mathcal{R}_n$, long queries



UBC-100, $\mathcal{R}_n$, short queries

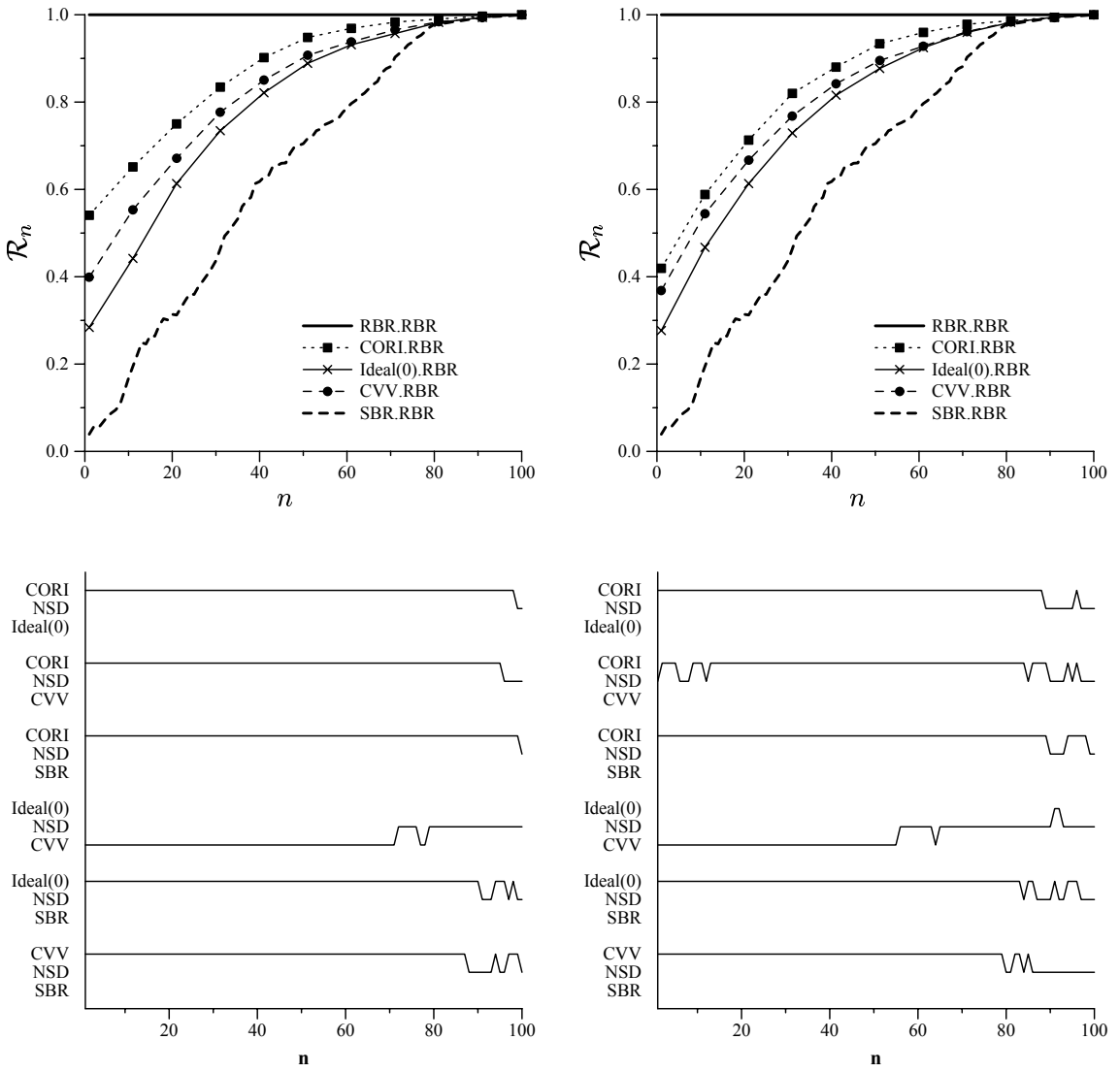


UBC-100, $\mathcal{P}_n$, long queries



UBC-100, $\mathcal{P}_n$, short queries

Figure 7.7: Collection selection results for the *UBC-100* testbed, long and short queries, $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ evaluation measures.



UBC-100, long queries

UBC-100, short queries

Figure 7.8: Comparison of approaches using the $\mathcal{R}_n$ measure, *UBC-100* testbed, long and short queries, plus significance of comparison.

number of queries for which each algorithm performed worse than the expected value given randomly generated rankings. RBR always performed better than the expected value for randomly generated rankings and is therefore not included in the table. Because we evaluated 100 queries at each n, the values in the table may be interpreted as the percentage of queries for which the approach performed worse than the random algorithm. Table 7.1 shows results for all three testbeds. These results tend to reinforce the conclusions that we drew using the Figures 7.3, 7.5 and 7.7. The number of queries for which the collection selection approaches performed worse than the expected performance for random selection is initially startling; however, we note familiar trends. On a whole, *CORI* tends to outperform the other approaches and overall *CORI* performs well relative to expected performance for random selection. Again, we note that for all approaches *UDC-236* is a more challenging testbed. Finally, we note that the SBR and *Ideal(0)* results suffer badly in the *UBC-100* testbed.

Table 7.1 also provides valuable insight into the performance of the different collection selection approaches. When examining Figures 7.3, 7.5 and 7.7 it would be easy to conclude that for low values of n there is not much difference in the performance of the competing algorithms. Table 7.1 tells a different story and implies that *CORI* has fewer failures than the other algorithms tested. This is an additional data point for consideration when evaluating such algorithms.

Table 7.1 was truncated at $n = 40$ for brevity. Most approaches continued to exhibit failures for larger values of n, although the instance of failures continued to decrease as n grew.

7.6 Discussion

In this chapter, we have presented a comparative collection selection evaluation for *CORI*, *CVV* and *gGlOSS* using six different test environments. We have also compared the performance to that of the SBR baseline and the expected performance of random selection.

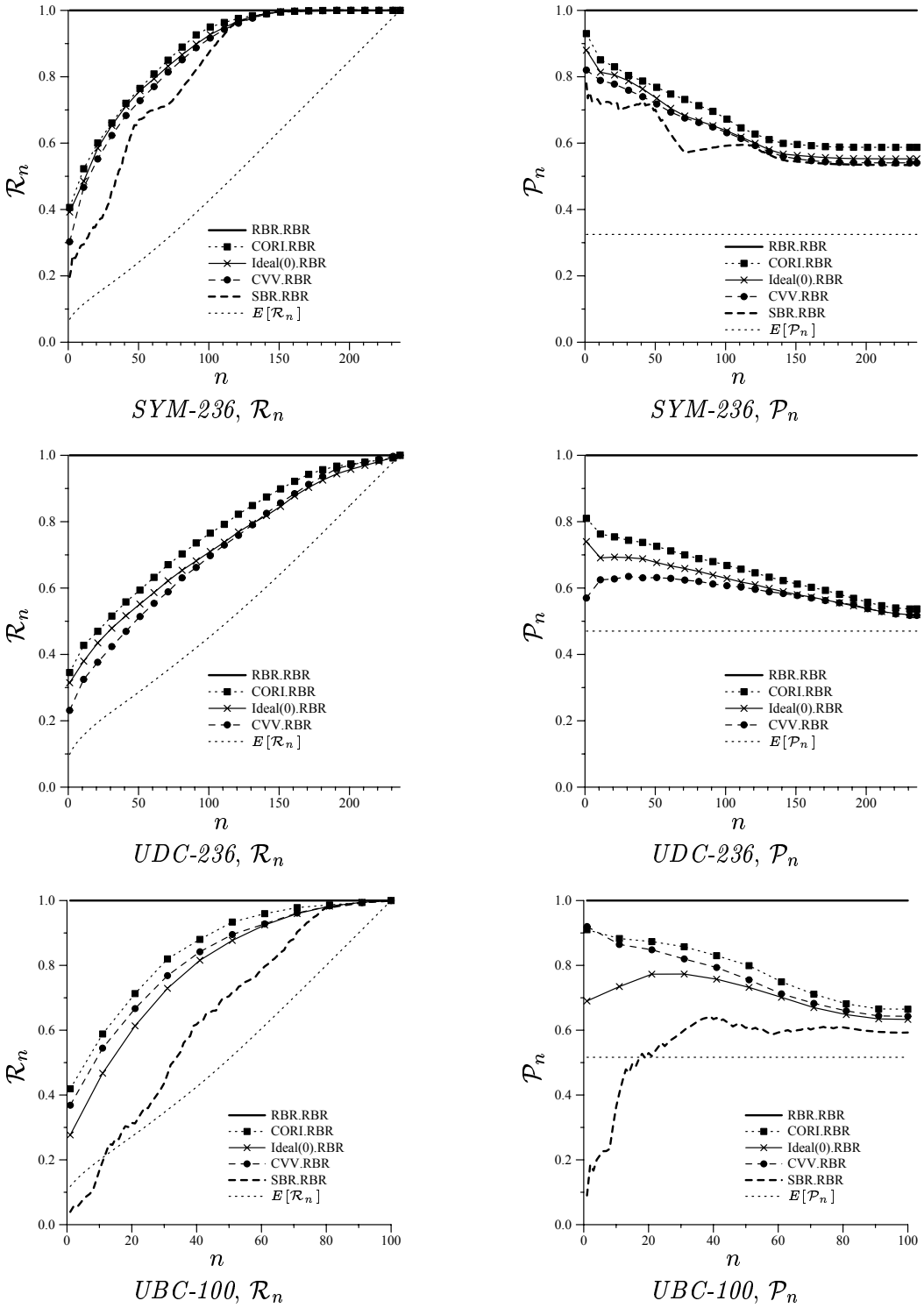


Figure 7.9: Expected performance of $\mathcal{R}_n$ and $\mathcal{P}_n$ evaluation measures.

	SYM-236				*UDC-236*			*UBC-100*			
n	**SBR**	**CVV**	**Ideal(0)**	**CORI**	**CVV**	**Ideal(0)**	**CORI**	**SBR**	**CVV**	**Ideal(0)**	**CORI**
1	36	24	18	13	50	35	29	93	22	44	22
2	32	18	11	5	42	25	19	87	16	42	18
3	30	16	9	6	31	19	13	90	16	41	16
4	22	13	10	5	31	17	14	86	16	40	13
5	20	11	8	5	34	16	12	84	16	41	13
6	19	10	9	4	33	16	12	83	15	39	12
7	19	12	8	3	32	20	14	83	11	29	12
8	15	10	9	3	31	21	13	84	13	29	9
9	13	8	7	2	31	20	13	80	14	26	8
10	14	9	7	2	30	22	12	77	11	22	8
11	13	8	6	2	27	18	12	72	11	23	7
12	11	7	6	2	30	17	12	64	9	20	7
13	14	9	6	2	29	18	11	61	7	17	9
14	11	8	5	2	29	17	11	61	7	16	9
15	8	8	6	2	29	17	12	59	8	15	8
16	10	9	6	2	30	16	12	60	6	12	9
17	11	6	5	2	31	17	11	57	3	11	8
18	10	6	5	1	31	17	11	53	5	9	8
19	10	6	5	1	30	16	13	56	3	10	6
20	9	6	4	1	29	16	10	51	3	8	4
21	10	6	3	1	28	15	9	52	2	6	4
22	9	8	3	1	26	17	9	53	2	8	4
23	10	7	3	1	23	18	10	47	2	7	4
24	4	5	3	1	22	18	9	46	2	7	3
25	4	5	3	1	24	16	10	49	2	6	3
26	6	4	3		22	15	9	45	2	5	3
27	6	4	3		22	14	9	43	2	5	3
28	6	5	3		22	13	9	39	3	5	3
29	7	5	3		24	13	8	38	5	5	3
30	7	3	4		24	12	8	33	4	5	3
31	6	3	4		22	10	7	24	4	5	2
32	7	3	4		22	10	6	20	4	5	2
33	7	3	4		21	10	7	19	4	5	2
34	7	3	3		20	10	6	17	4	5	1
35	7	3	3		21	10	6	17	4	6	2
36	7	3	3	1	21	10	5	14	4	5	2
37	7	3	2		18	10	5	15	4	5	2
38	8	3			19	11	5	14	4	5	1
39	7	2			21	9	4	13	4	5	2
40	7	2			20	9	4	14	4	5	2

Table 7.1: Number of times each algorithm achieved a value of $\mathcal{R}_n$ lower than the expected performance for random selection.

This had produced a potentially baffling array of figures containing results.

Despite the variety of experimental environments, the overall results are fairly consistent. While there is significant difference between *CORI*, *Ideal(0)*, *CVV* and SBR, they tend to perform similarly when compared to the performance of the baseline against itself (RBR.RBR). There is substantial room for improvement for all of the collection selection approaches.

Overall, the *CORI* collection selection approach performs most accurately and most consistently. For all six experimental environments, *CORI* is significantly better than (or not significantly different from) the other approaches. In Chapter 8, we will study *CORI* and the overall *df.icf* approach in more detail. In Chapter 9, we will use *CORI* as the collection selection approach when we investigate the impact of collection selection on document retrieval in a multi-collection environment.

The *gGlOSS* representative *Ideal(0)* performed adequately but its correlation with SBR proved problematic for the *UBC-100* testbed. When comparing *CORI* and *Ideal(0)* in a broader scope, the main advantage of *CORI* is that it requires less statistical information about the collections. The *df* information that *CORI* requires is also easier to compare across collections using different information retrieval systems than the term weight information required by *gGlOSS*. Also, as we will see later, *df* information can be efficiently approximated by sampling techniques.

While CVV is an intuitively appealing approach that also requires limited statistical information, CVV tended to be the worst-performing approach in all of our experiments.

8

Collection Selection — Details and Analysis

In Chapter 7 we performed comparisons of the published *CORI*, *gGlOSS* and *CVV* collection selection approaches using six different experimental environments. With those experiments, we were able to identify differences in collection selection performance and to identify some of the reasons for the differences. We found that a correlation with SBR is partly responsible for the *gGlOSS* representative *Ideal(0)*'s performance. However, the experiments of Chapter 7 did not study the components of the collection selection algorithms in detail. Here, we isolate components of the *CORI* algorithm, which consistently performed well in the experiments of Chapter 7, to provide additional insight into collection selection performance.

We begin by providing context for these experiments. We then abstract the general *df.icf* approach employed by *CORI* and describe our experiments. We present results that examine the influence of the general *df* and *icf* components, discuss normalization approaches and examine the *CORI* algorithm in more detail. We follow this with a comparison of several straightforward $df \cdot icf$ approaches. We wrap up the chapter with a summary of the collection selection experiments reported in this chapter as well as in Chapters 6 and 7.

8.1 Context

Because *gGlOSS*, *CVV* and *CORI* use similar collection statistics as the basis of their ranking algorithms, we saw an opportunity to learn more about the behavior of collection selection algorithms by means of a careful study of some of the components of the *CORI* algorithm. In particular, we hoped to discover why *CORI* did not exhibit the pronounced tendency to select large collections observed for *Ideal(0)*.

The object of our study was to discover what factors in the *CORI* algorithm were most important in terms of its performance. There were two obvious differences in the *CORI* and *gGlOSS* approaches.

1. *CORI* represents collections as virtual documents with suitable term weights and employs a document processing strategy on this representation.

2. *CORI* contains a normalization component that scales document frequency (*df*) information within those virtual documents based on collection size.

In part the *CORI* advantage over *gGlOSS* is due to the fact that it is not highly correlated to SBR. There is clearly some size compensation going on, which may be intrinsic to the representation or may be due to collection size normalization. We undertook a systematic study to examine these and other effects.

One set of experiments examined *CORI* directly, individually disabling portions of the *CORI* computation to gauge the impact on performance. Additional experiments investigated the basic components of the *CORI* approach. For these experiments, we abstracted the collection selection process in a way that let us have reasonably fine control over a family of weighting functions so that we could observe the effects of changes in a controlled environment.

8.2 Abstracting $df \cdot icf$ Approaches

CORI represents collections as virtual documents, that is, each collection is represented by terms and their document frequencies. In fact this is nothing more than the matrix $F = [df_{ij}]$ that *gGlOSS* requires for its operation. The difference between *CORI* and *gGlOSS* is simply in the usage of the document frequency information. Where *gGlOSS* uses the information in F in conjunction with term weight information from an additional W matrix, each row of F is a virtual document in the *CORI* terminology. F contains N rows, one for each collection and a column for each term in the union vocabulary, the set of all unique terms in all the collections. It will be the case that $df_{ij} = 0$ when term t_j does not occur in any document of collection C_i. Depending on the heterogeneity of the collections, F can be a very sparse matrix.

CORI treats the virtual documents as a collection and performs a similarity calculation of a query q against these "documents." The subsequent document ranking is the desired collection ranking. A virtual document representation is not limited to the *CORI* collection selection approach; this interpretation can be used for other collection selection techniques. This is the approach that we take for these experiments. Treating collections as virtual documents is simply a variation of the general document retrieval $tf \cdot idf$ approach discussed in Chapter 2, with a subtle difference. In the representation for the collections we are using df as the tf component and inverse collection frequency (icf) as the idf component. This means that the weighting strategies are of the form $df \cdot icf$, and this is actually the form we investigate. The intuition is that a collection having many documents containing a term t_j that is reasonably rare across all the collections (i.e., occurs in very few other collections) should be ranked highly; the term t_j is a good discriminator.

We note that *CORI* does not have this precise form (see Equation 7.1 in Chapter 7). In particular, the df component of the *CORI* ranking function is $df/(df + 50 + 150 \cdot cw/\overline{cw})$, and the icf component is normalized by $\log(N + 1.0)$. However, while both components are normalized, *CORI* *is* a form of the general $df \cdot icf$ approach.

Considering *df* and *icf* simply as abstract components is useful from an experimental standpoint. Information retrieval research has found that different combinations of tf and *idf* are effective for document retrieval [SB88, SBM96]. Just as we can vary the influence of components for documents using $tf \cdot idf$ approaches, we can vary the influence of the analogous components for collection selection. In French *et al.* [FPC99a], we presented an in-depth study of varying the influence of the *df* and *icf* components. We provide a high-level summary of similar experiments here, but do not cover the full range of detail. Instead, in this chapter, we focus on the overall potential of the $df \cdot icf$ approaches and the details of the *CORI* algorithm. Emmitt [Emm99] performed an extensive evaluation of the components of the *CVV* algorithm, including adding a *CVV* component to the $df \cdot icf$ formulation. He found, as we did, that while *CVV* uses the same information of generic $df \cdot icf$ approaches, the generic $df \cdot icf$ approaches are simpler to compute and often more effective.

8.3 Experimental Environment

For these experiments, we used the same six test environments used in Chapter 7. These test environments utilize the *SYM-236*, *UDC-236* and *UBC-100* testbeds and both the long and short query formulations of TREC topics 51-150. The six test environments can be specified as:

- (*SYM-236*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$), (*SYM-236*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$),
- (*UDC-236*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$), (*UDC-236*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$),
- (*UBC-100*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$) and (*UBC-100*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$).

For each test environment and all experiments, the default configurations of $\mathcal{LM}$ and $R_{\mathcal{LM}\rightarrow\mathcal{C}}$ apply. We used the same preparation of the F matrix that was described in Chapter 7.

8.4 Experiments and Results

In this section we report performance results for two classes of experiments. In the first set of experiments, we consider the basic $df \cdot icf$ approach, then investigate some obvious ways to normalize df values. In the second set of experiments, we examine the *CORI* algorithm by systematically disabling the features that differentiate *CORI* from the bare-bones $df \cdot icf$ approach. Queries are handled in the same way for each approach. As was the case for *CORI*, the collection score is the average of the values due to each query term. We will co-opt the description approach used by Callan *et al.* [CLC95] to define our test cases. T is used to represent the df component and I the icf component, so that these functions will have the form $T \cdot I$. For brevity, we report performance using only the $\mathcal{R}_n$ evaluation measure. We later summarize representative approaches and discuss the significance of the differences in performance.

8.4.1 Influence of Algorithm Components

We start by considering a very simple use of the df information, using unaugmented, un-normalized df information to represent each collection. For this case,

$$\begin{aligned} T &= df \\ I &= 1 \end{aligned}$$

This approach is labeled $df.RBR$ in Figure 8.1.

We then considered a very simple un-normalized $df \cdot icf$ approach:

$$\begin{aligned} T &= df \\ I &= \log\left(\frac{N + 0.5}{cf}\right) \end{aligned}$$

This approach is labeled $df - icf.RBR$ in Figure 8.1.

In Figure 8.1, these two approaches are compared to *CORI* and SBR. It is no surprise

to find that the addition of an icf component improves average performance over that seen with df alone; the icf component decreases the influence of very common terms, allowing terms with more discriminating power to guide selection. However, it is interesting to note that even collection selection based on very simple df information exhibits reasonable performance.

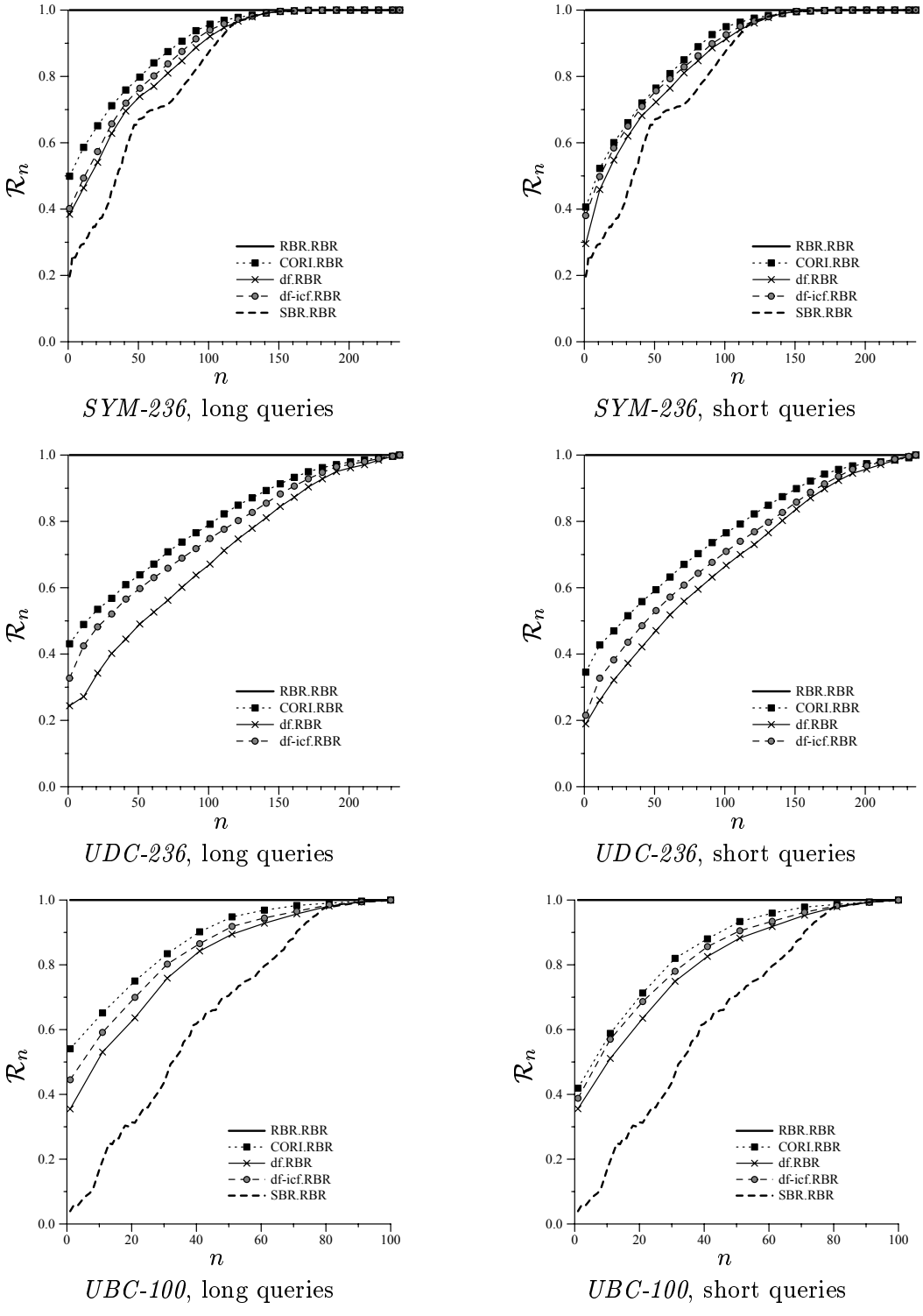
We have noted before that $df \cdot icf$ approaches require less statistical information from collections than *Ideal(0)*. In addition, simple $df \cdot icf$ formulations are simpler to compute than either *Ideal(0)* or *CVV*. In Chapter 7, we established that *CORI* tends to outperform both *Ideal(0)* and *CVV* in our six test environments. We have also compared the performance of *Ideal(0)* and *CVV* to the very simple $df \cdot icf$ formulation defined above. *CVV* did not significantly outperform $df - icf$ for any of our six test environments. *Ideal(0)*, on the other hand, did perform better than $df - icf$ for some values of n for the (*SYM-236*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$) and (*UDC-236*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$) test environments.

Our second step was to consider a few obvious approaches for normalizing the df component. Approaches that employ an un-normalized df component, and do not use any other form of collection size normalization have the potential to exhibit a high correlation with SBR. Put another way, a df value of 50 has a different interpretation in a collection of 100 documents than in a collection of 1,000 documents. We would like for a collection selection approach to reflect this.

Our first approach was to consider the obvious solution of normalizing the df value by the number of documents in the collection:

$$\begin{aligned} T &= \frac{df}{|C|} \\ I &= \log\left(\frac{N+0.5}{cf}\right) \end{aligned}$$

These results are labeled $(df/|C|) - icf.RBR$ in Figure 8.2. Unfortunately, we find that this can give disproportionate influence to small collections. When terms occur in even one document in these collections, the normalized value causes the term to appear very



SYM-236, long queries

SYM-236, short queries

UDC-236, long queries

UDC-236, short queries

UBC-100, long queries

UBC-100, short queries

Figure 8.1: $\mathcal{R}_n$ performance of simple df and $df \cdot icf$ approaches.

representative of that collection. Note that this is not an issue for the *UDC-236* testbed, in which all collections contain roughly the same number of documents. However, for both long and short queries using the *SYM-236* and *UBC-100* testbeds, the performance of $(df/|C|) - icf$ was poor.

Our next approach was a scaling approach rather than a normalization approach. In Callan *et al.* [CLC95], T was originally defined as

$$T = d_t + (1 - d_t) \cdot \frac{df}{df + K}$$

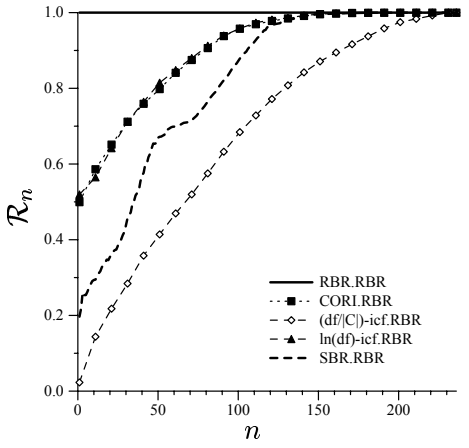
where $K = k \cdot ((1 - b) + b \cdot cw/\overline{cw})$ and k and b are constants. The basic formulation was originally suggested by Robertson and Walker [RW94] as an approximation to the 2-Poisson model in a document retrieval context. This approach has been shown to perform well for document retrieval [RWJ+94]. It was transformed to accommodate collection selection for its use in *CORI*. Figure 8.3(a) illustrates the impact of the *CORI* normalization approach on *df* values when collections are average size, one half average size and twice average size. Note that *df* values have higher impact in smaller collections. However, the normalization overall is very subtle.

Observing the shape of the curves representing normalized values, we considered the possibility of approximating this effect by taking the log of the *df* values. While this does not meet all of the criteria set forth by Robertson and Walker [RW94][1], it may prove to be a reasonable approximation for our purposes. $\log(df)$ is included in Figure8.3(b). While the effect differs for very small values of *df*, the overall shapes of the curves are similar. When considered on the larger scale of un-normalized *df* (Figure8.3(c)), the impact on *df* values is similar.

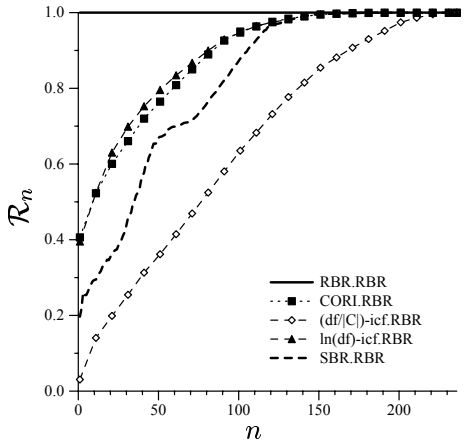
We can describe this approach as:

$$T = \log(df)$$

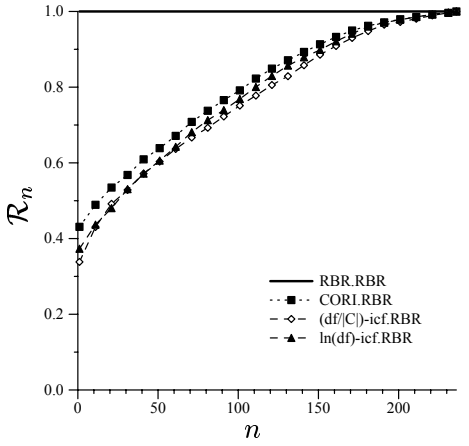
[1]While $\log(df)$ increases monotonically with *df*, and increases asymptotically, $\log(df)$ is not defined for $df = 0$ and is not appropriate for approximating probabilities because it is not in the range $[0, 1]$.



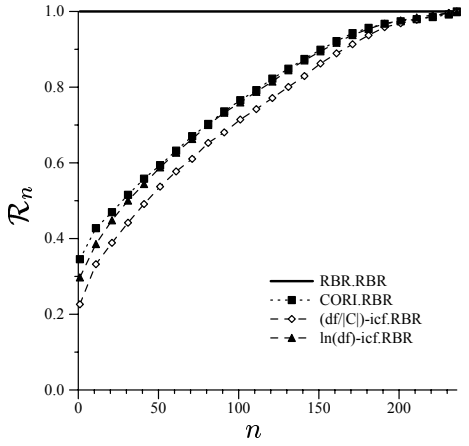
SYM-236, long queries



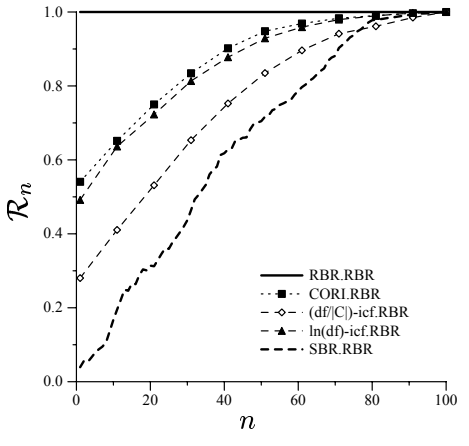
SYM-236, short queries



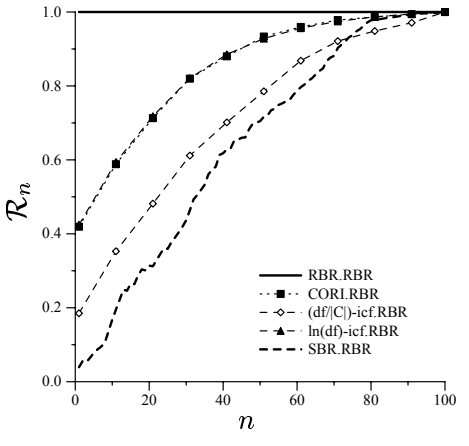
UDC-236, long queries



UDC-236, short queries



UBC-100, long queries



UBC-100, short queries

Figure 8.2: $\mathcal{R}_n$ performance of simple normalization approaches for the *df* component.

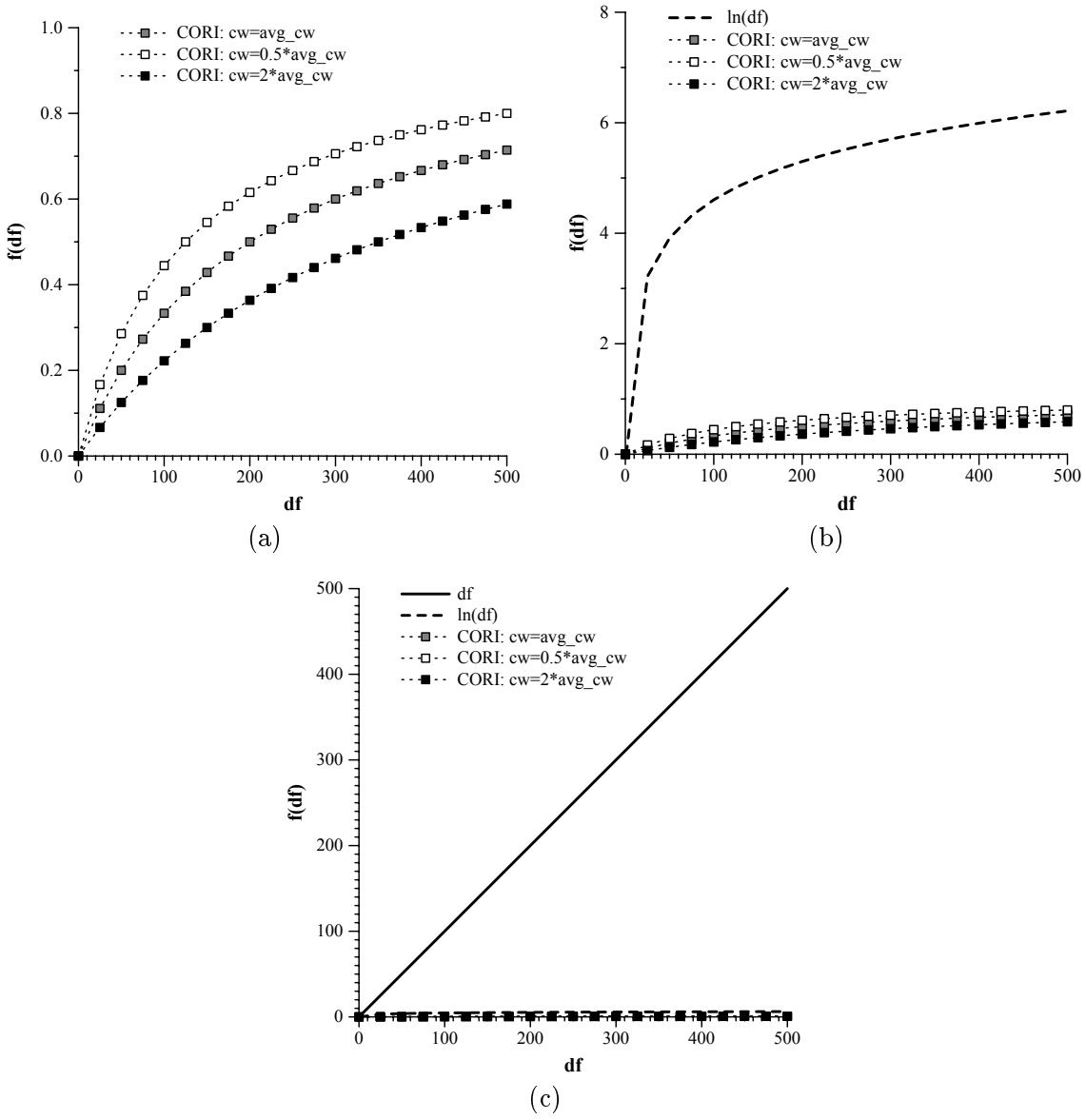


Figure 8.3: The impact of normalization approaches for values of *df*.

$$I = \log\left(\frac{N+0.5}{cf}\right)$$

The results are labeled $ln(df)-icf.RBR$ in Figure 8.2. The results are much more promising than those seen for $(df/|C|) - icf.RBR$. We will consider the $ln(df) - icf.RBR$ approach in more detail later.

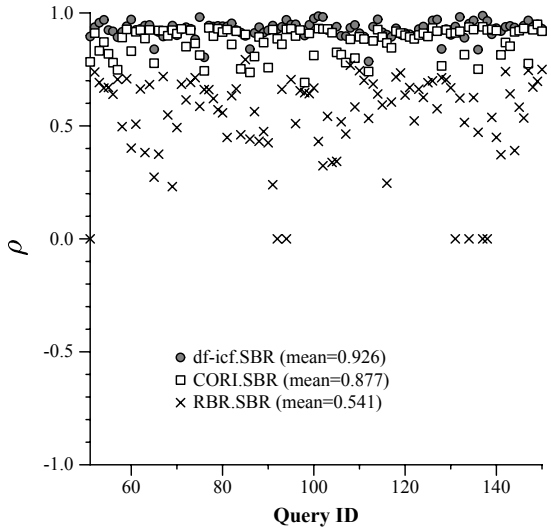
8.4.2 Modifications to *CORI*

When we began this chapter, we speculated that the performance exhibited by *CORI* in Chapter 7 is due to either the $df \cdot icf$-based virtual document representation or to *CORI*'s normalization step. We were particularly interested in the fact that *CORI* does not exhibit a pronounced SBR correlation. Stated operationally, we noted that *CORI* consistently identifies medium-sized collections that contain large numbers of relevant documents.

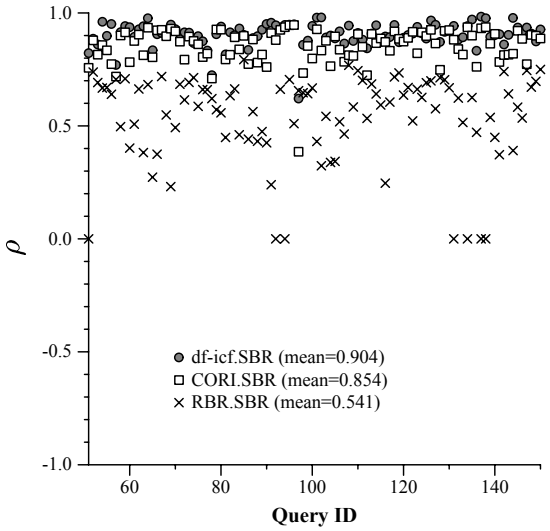
To determine if *CORI*'s $df \cdot icf$-based virtual document representation is responsible for its performance, we used the Spearman correlation coefficient to compare the correlation of a simple $df \cdot icf$ approach to SBR and to compare the correlation of *CORI* to SBR. The results are shown in the graphs of Figure 8.4. Mean values for ρ are included in the plot labels. While the correlation results for the simple $df \cdot icf$ approach and *CORI* are similar, we find that the simple $df \cdot icf$ approach is more highly correlated with SBR on average. In addition, for the *SYM-236* testbed 89 of the long queries and 86 of the short queries were more highly correlated with SBR using $df - icf$ than *CORI*; for the *UBC-100* testbed, 90 of the long queries and 76 of the short queries were more correlated with SBR using $df - icf$ than *CORI*. This suggests that the *CORI* normalization approach bears further investigation.

We proceeded with a direct examination of the *CORI* algorithm, systematically disabling features that differentiated it from a simple $df \cdot icf$ approach. For reference, the standard *CORI* approach can be specified as follows:

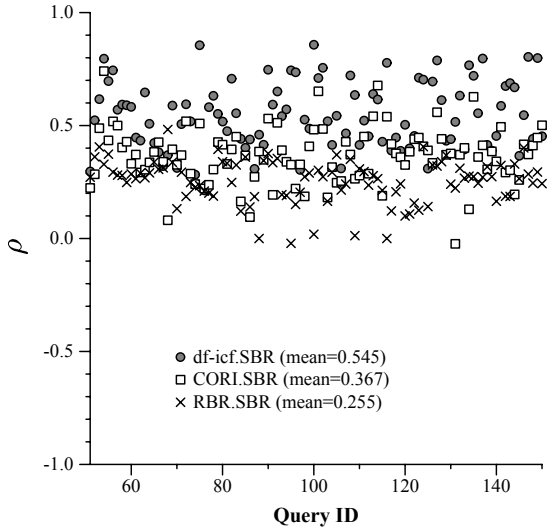
$$T = \frac{df}{df + 50 + 150 \cdot cw/\overline{cw}}$$



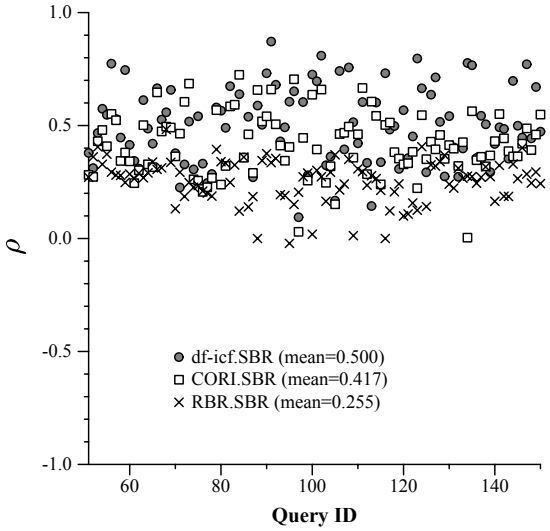
SYM-236 testbed, long queries



SYM-236 testbed, short queries



UBC-100 testbed, long queries



UBC-100 testbed, short queries

Figure 8.4: Spearman correlation of $df \cdot icf$ selection approaches with the SBR baseline.

$$I = \frac{\log\left(\frac{N+0.5}{cf}\right)}{\log\left(N+1.0\right)}$$

Our first step was to consider the normalization of the icf component. Given our treatment of queries, the denominator $\log\left(N+1.0\right)$ is simply a constant that can be factored from the computation. We verified that performance did not change when we removed this normalization component. Given this, the only difference between *CORI* and a basic $df \cdot icf$ approach is in the df component (specified as T here).

The cw normalization component used by *CORI* requires information about the number of words in a collection. In an operational environment, this information may be difficult to obtain. To study the impact of the cw component, we performed two slight modifications to the T component, but found little impact on the average results. We first considered the impact on performance if we assume that all collections are "average sized", i.e. $cw = \overline{cw}$ or $cw/\overline{cw} = 1$ for all collections. This modification had little impact on the average results for our six test environments. For the *UDC-236* and *UBC-100* testbeds, assuming that collections are "average sized" is not unreasonable, given their parameters for construction. For the *SYM-236* testbed, a subtle difference was visible. Because this heuristic defeats collection size normalization, we do not recommend it in the general case; however, if collection size information is unavailable, or if most collections are average sized, this appears to be a reasonable heuristic. We also considered replacing the cw term with the number of documents per collection, statistical information that may be easier to obtain. Unfortunately, these results are less promising than the standard *CORI* results. For our six test environments, simply assuming that collections are average sized, or taking the log of the df values is a better heuristic.

8.4.3 Comparison of $df \cdot icf$ Results

The variations of $df \cdot icf$ collection selection approaches that we have discussed so far have helped to isolate the impact of components on performance. However, we are also interested in the comparative performance of approaches that are fairly obvious choices for inclusion

in a system. In Figures 8.5–8.7 we present comparisons of the $df.RBR$, $df - icf.RBR$, $ln(df) - icf.RBR$ and $CORI.RBR$ results from Figures 8.1 and 8.2. We also include the paired Wilcoxon results for the significance of the comparisons. Overall, we find that $CORI$ and $ln(df) - icf$ tend to outperform df and $df - icf$. The differences between $CORI$ and $ln(df) - icf$ vary by testbed and query set. If cw information is available, we prefer $CORI$ because it incorporates collection size information. For environments with even greater variety in collection size than that seen in *SYM-236*, the collection size normalization may prove to be crucial. We also know that $CORI$ performs well using sampled language models and we have no information for $ln(df) - icf$ in this case. However, lacking cw information, $ln(df) - icf$ appears to be a reasonable heuristic.

8.4.4 Summary

These rather fine-grained experiments were designed to study the components of the $CORI$ collection selection approach, which was shown in Chapter 7 to perform well in our test environments. We find that, overall, $df \cdot icf$ collection selection approaches perform well. They require limited information from the underlying collections and the computations are generally simple. We find that the $CORI$ df normalization approach is the key to its improved performance over more bare-bones $df \cdot icf$ approaches.

In addition to answering this broader question, we also uncovered some more detailed findings.

- We confirmed that adding an icf component to a df virtual document representation improves collection selection performance by allowing terms with higher discriminatory power to influence collection selection.

- We note that a simple $df \cdot icf$ approach consistently meets or exceeds the performance of CVV, while having a much simpler computation. This very simple $df \cdot icf$ approach can perform better or worse than *Ideal(0)*, depending on the test environment.

- When normalizing df values, an obvious approach is to divide them by the number of

documents in the collection. Unfortunately, we find that this can give undue influence to very small collections.

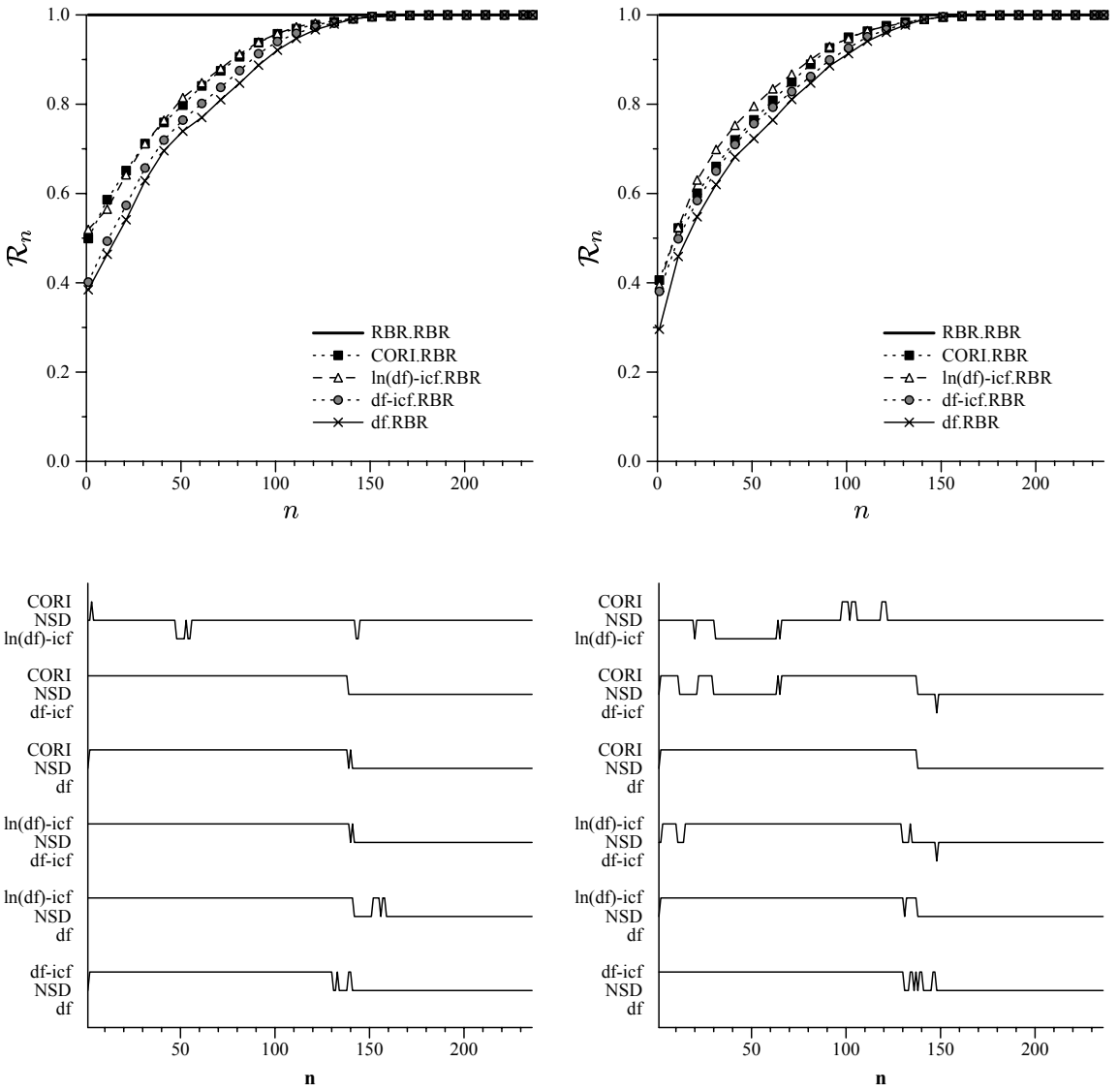
- The more subtle *CORI df* normalization approach is more effective than dividing *df* by the number of documents in the collection. However, simply taking the log of the *df* values is a reasonable heuristic.

8.5 Summary of Collection Selection Experiments

With this chapter, we bring to a close the experiments focused directly on collection selection. With Chapter 9 we will move on to study the implications of collection selection for information retrieval in a multi-collection environment. For those experiments, we will use *CORI* and RBR collection selection. However, before we move to the next phase of our experiments, we would like to recap our collection selection experiments.

Our earliest experiments, covered in Chapter 6, focused on the *gGlOSS* collection selection approach, studied within a single experimental environment. With those experiments, we showed that the *gGlOSS Max(l)* and *Sum(l)* estimators approximate the *Ideal(0)* baseline well, but that *Ideal(0)* is not a strong predictor of the presence of relevant documents in a collection. During the course of these experiments, we chose *Ideal(0)* as a representative for *gGlOSS* for further experiments.

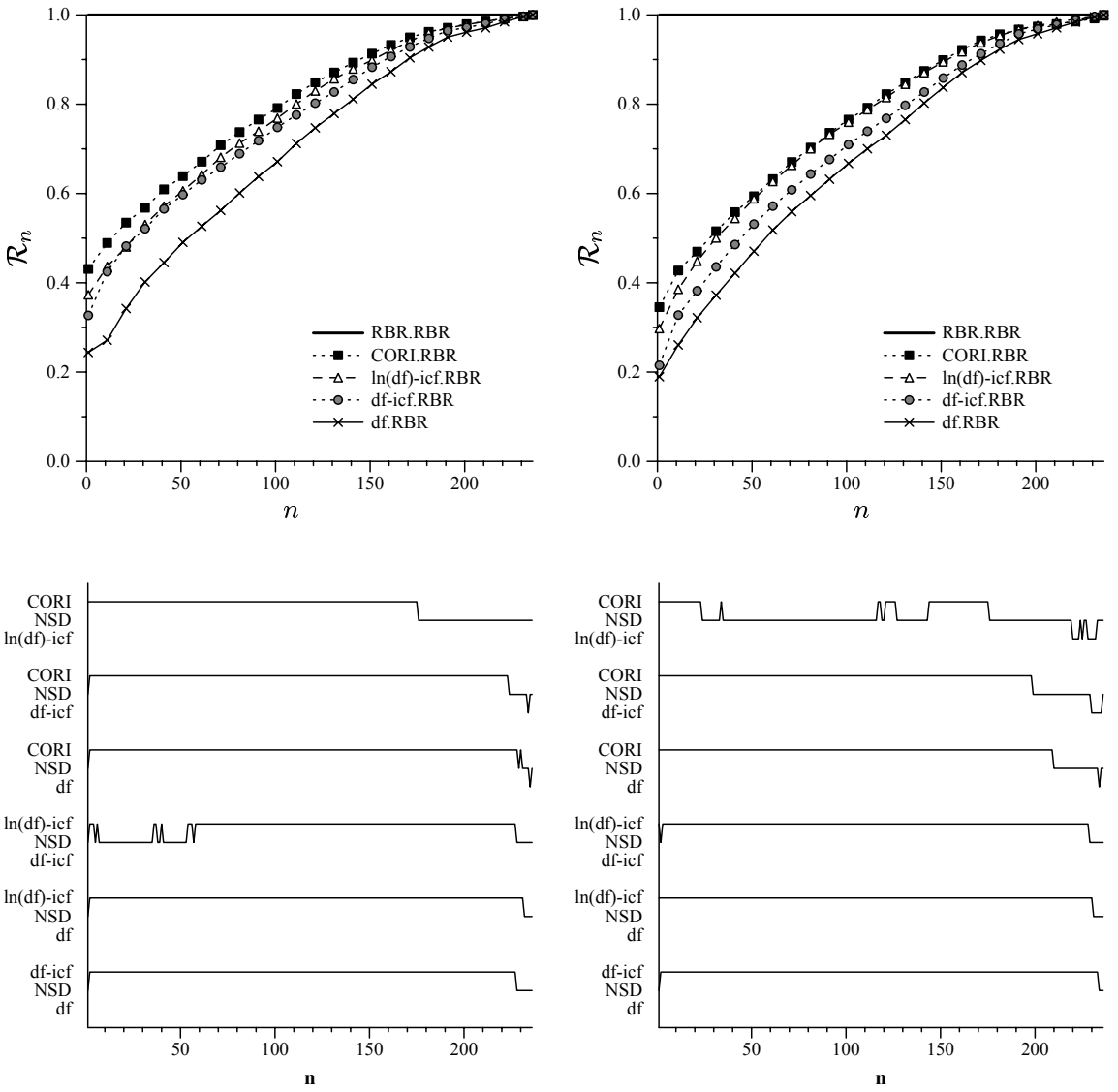
Following those early experiments, we expanded the scope of our efforts substantially. We undertook a comparative study of the *CORI*, *CVV* and *gGlOSS* approaches using six different test environments. These experiments represented the first large-scale comparisons of these approaches in common environments. With these experiments, we found that *CORI* consistently performs well in all six of our test environments. This led us to the investigation of the general $df \cdot icf$ approach to collection selection, of which *CORI* is an example. In this chapter, we showed that $df \cdot icf$ approaches are generally effective and that the *CORI df* normalization approach gives it the advantage over very simple instances of the approach.



SYM-236, long queries

SYM-236, short queries

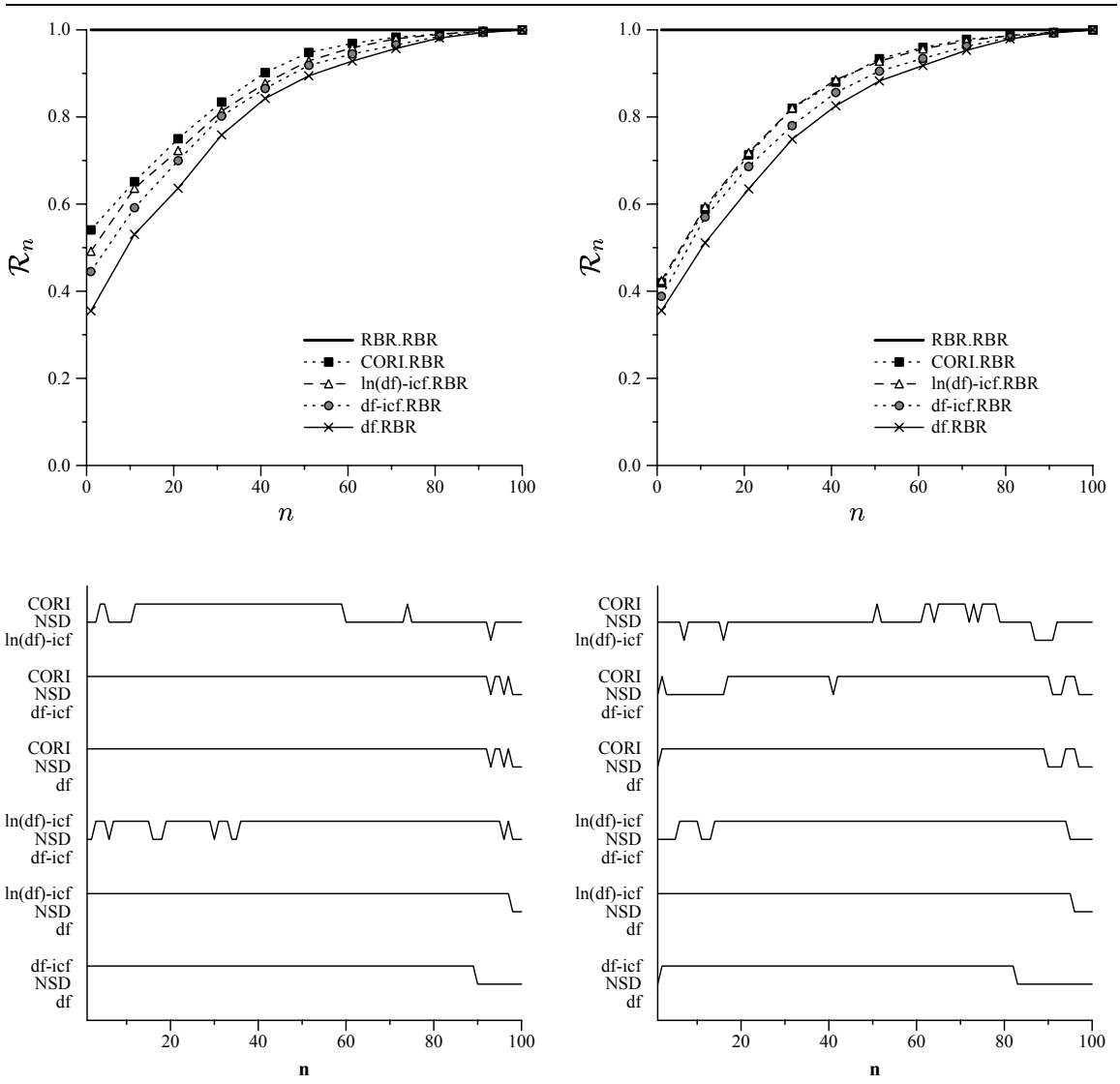
Figure 8.5: Comparison of approaches using the $\mathcal{R}_n$ measure, *SYM-236* testbed, long and short queries, plus significance of comparison.



UDC-236, long queries

UDC-236, short queries

Figure 8.6: Comparison of approaches using the $\mathcal{R}_n$ measure, *UDC-236* testbed, long and short queries, plus significance of comparison.



UBC-100, long queries

UBC-100, short queries

Figure 8.7: Comparison of approaches using the $\mathcal{R}_n$ measure, *UBC-100* testbed, long and short queries, plus significance of comparison.

9

Multi-collection Retrieval Experiments

In Chapters 6–8 we studied a variety of collection selection approaches in detail. However, collection selection is only one facet of multi-collection retrieval. We are also concerned with the effectiveness of document retrieval within a multi-collection environment in which collection selection is employed. In this chapter, we explore multi-collection document retrieval using the same six test environments employed in Chapters 7 and 8. We study document retrieval effectiveness when *CORI* and RBR are used for collection selection. These experiments are an expansion of the work reported in Powell *et al.* [PFC+00].

To date, document retrieval in a multi-collection environment has been compared to, and performed less effectively than, retrieval in an equivalent single-collection environment. However, in recent work, Xu and Croft [XC99] discussed the possibility that retrieval performance in a multi-collection environment may exceed performance in an equivalent single-collection environment. In that work, Xu and Croft were pessimistic about the potential to achieve both retrieval efficiency and effectiveness in heterogeneous multi-collection environments. Instead they focused on document collections created by clustering documents. They achieved good results with this clustering approach; however, clustering requires the cooperation of the owning organizations that administer the collections being searched. Unfortunately, this level of cooperation may not be feasible in most operational environments.

We believe that the potential exists to exceed single-collection document retrieval performance while maintaining search efficiency even in multi-collection environments where clustering is not possible or where the composition of document collections is imposed by the owning organization and is outside our control. Document collections may be created according to many different criteria, for example, according to publication source, publication date, or to equalize collection size. We will need to engineer retrieval systems that are robust to such decompositions. Accordingly, we need to know how various algorithms behave in such environments. In this chapter we report on experiments conducted using six different test environments.

Our goal for this chapter is to investigate the following general questions. For different multi-collection organizations of documents, how does document retrieval performance compare to equivalent single-collection performance? What is the overall impact of selecting more or fewer collections to search? What is the impact on document retrieval of disseminating collection-wide information among the collections?

We restate these questions as three hypotheses to focus the problem.

Hypothesis 1: When very good collection selection is employed, multi-collection retrieval can outperform single-collection retrieval in a variety of environments.

Hypothesis 2: It is possible to achieve good document retrieval performance when few collections are selected; however, increasing the number of collections selected will improve performance.

Hypothesis 3: In a multi-collection environment, the use of collection wide information (CWI) will improve document retrieval performance.

The first hypothesis requires some clarification. We are interested in determining if the use of very good collection selection can enable multi-collection retrieval to outperform single-collection retrieval in multi-collection environments where documents may not have been organized to enhance retrieval. For example, the documents may be organized chronologically or to equalize the size of collections.

9.1 Test Environments and Experimental Setup

In the experiments reported here, we examined retrieval performance in both single-collection and multi-collection test environments. We used the six multi-collection test environments specified in Chapter 7:

- (*SYM-236*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$), (*SYM-236*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$),
- (*UDC-236*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$), (*UDC-236*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$),
- (*UBC-100*, $\mathcal{Q}_l$, $\mathcal{J}_{TREC4}$) and (*UBC-100*, $\mathcal{Q}_s$, $\mathcal{J}_{TREC4}$),

as well as their corresponding equivalent single-collection environments. For each test environment and for all experiments, the default configurations of $\mathcal{LM}$ and $R_{\mathcal{LM}\rightarrow\mathcal{C}}$ apply. Some of our experimental scenarios varied the documents used to create I_i for collection C_i. We will elaborate on this in a later section. For our experiments, we varied the collection selection approach, the number of collections searched and the results-merging approach.

We evaluated the impact that these variations had on the final document retrieval results. We searched the highest-ranked collections, merged the returned results, then evaluated the quality of the merged list of documents. Descriptions of the testbeds can be found in Chapter 4, while details of the selection approaches are provided in Chapter 7. A description of the merging approaches, and a more detailed description of the evaluation approach are given below.

This work differs from previous research in multi-collection retrieval in several ways. First, we utilized multiple testbeds with different distributions of relevant documents for our experiments. Second, whereas other efforts have fixed the number of collections selected [XC98, XC99, CPFC00], we study the impact of selecting more or fewer collections. We also consider the combination of both collection selection and the dissemination of collection-wide information. Viles and French [VF95b, FV96] studied the use of CWI in a multi-collection environment in which collection selection was not used, while Xu and Croft used CWI for all experiments reported in [XC99].

We employed two different collection selection approaches in our experiments—we chose one achievable approach as a realistic case and one very good, but as yet unachievable approach as a best-case scenario. A comparison of different existing collection selection algorithms reported in Chapter 7, plus the comparisons reported by Callan *et al.* [CPFC00] showed that the *CORI* [CLC95] approach outperforms both *CVV* [YL97] and *gGlOSS* [GGM95, GGMT99]. As a result, we chose *CORI* as our achievable collection selection approach[1]. As our best-case approach, we chose the *RBR* baseline that was used to evaluate the different collection selection approaches in Chapters 6–8.

In these experiments, we used *RBR* as an oracle for selection; *RBR* provides the best collection ordering that is possible given *only* knowledge of where the relevant documents for a query are located. It has no knowledge of document ranking or merging. As a result, there may be situations for which a different ordering of collections produces a better overall document retrieval performance than *RBR*. For example, a collection containing a large number of relevant documents may be poorly maintained and may be inoperative when queries are issued. Given this additional information, not available to RBR, selecting a different collection might be considered more appropriate.

Our six test environments are composed of the *SYM-236*, *UDC-236* and *UBC-100* testbeds and the $\mathcal{Q}_s$ and $\mathcal{Q}_l$ query formulations of TREC topics 51-150. The three testbeds represent three convenient ways to organize documents into collections or to partition a large collection into several smaller ones. Xu and Croft [XC99, p. 256] expressed concern that the distribution of relevant documents in collection decompositions such as these may adversely affect the efficiency or effectiveness of multi-collection retrieval. We discuss this issue in Section 9.3.2 and summarize the distribution of relevant documents in the *UBC-100*, *SYM-236*, and *UDC-236* testbeds in Table 9.8.

[1]Recall that for these experiments, we are using the official UMass implementation of *CORI* instead of the UVA implementation that was used for the comparative collection selection experiments. For a comparison of the UMass and UVA implementations of *CORI*, see Appendix A.

9.1.1 Query Processing

In all scenarios and all experiments, query processing at the collections was performed using Inquery [CCH92]. We used unstructured queries and retrieved 100 documents from each collection.

9.1.2 Results Merging

We used two different merging approaches in these experiments. The first approach was a simple raw-score merge. When collection-wide information is used in a multi-collection environment, the document scores from different collections are comparable and a raw-score merge is feasible. Our second merging approach was to use the default Inquery multi-collection merging algorithm. This approach uses a combination of the score for the collection and the score for the document to estimate a normalized score.

The collection score was computed differently for the two collection selection approaches. When *CORI* was used for collection selection, the normalized collection score was computed as follows:

$$C' = (C - C_{min})/(C_{max} - C_{min}) \tag{9.1}$$

where C is the raw collection belief score for the collection (see Chapter 7 for the definition of the raw belief score), and C_{max} and C_{min} are the maximum and minimum scores a collection could obtain for a particular query. When *RBR* was used for collection selection, the normalized collection score was computed as:

$$C' = (101 - R)/100 \tag{9.2}$$

where R is the collection rank.

The normalized document score D'' for a document with initial score D was computed as:

$$D' = ((D - D_{min})/(D_{max} - D_{min}) \tag{9.3}$$

$$D'' = (1.0 \cdot D' + 0.4 \cdot C' \cdot D')/1.4 \quad (9.4)$$

where D_{max} was the highest score an *ideal* document could get for that query in that collection, and D_{min} was the lowest score a document could get for that query in that collection. The normalization of D'' by 1.4 is done to restrict document scores to the range $[0, 1]$.

9.1.3 The Three Scenarios

To enable a comparison of multi-collection and single-collection performance, and to judge the impact of the use of CWI in multi-collection retrieval, we used three different document organization scenarios.

`single` – For each of the test environments, all documents from the collections in the testbed, $\mathcal{D} = \cup_{i=1}^{N} C_i$, are considered as a single collection. No merging step is necessary.

`multi-LI` **(local information)** – For each of the multi-collection test environments, the index I_i for each collection C_i is constructed using only statistical information from documents contained in C_i. For this scenario, CWI is *not* available, so document scores from the different collections are not directly comparable. The default Inquery merge is used.

`multi-CWI` **(collection-wide information)** – For each of the multi-collection test environments, CWI is available for collection indexing. That is, the index I_i for each collection C_i is constructed using statistical information gathered from all documents $d \in \mathcal{D}$. For example, global *idf* values are available for determining term selectivity and all document length values are available for performing document length normalization. Operationally, for some document d, query q, and similarity measure $sim(q, d)$, the document-query similarity is the same if document d is located in collection C_i or C_{i+1} or if d is located in an equivalent single-collection environment.

Note however that I_i only contains representations for the documents in C_i. Document scores from different collections are comparable, so a raw-score merge is used.

9.1.4 Execution and Evaluation

Given a testbed, a document organization scenario and a selection approach, we used the selection approach to rank all of the collections in the testbed. Then, the top-ranked 2, 5, 10 and 20 collections were considered selected for search, i.e. elements of $\mathcal{C}_{sel}$. The query used to rank the collections was executed at each selected collection and 100 documents were returned from each collection. The individual result lists were merged using the merge algorithm specified in the scenario description. The top 100 documents from the merged list were evaluated.

We used the `trec_eval`[2] program to measure precision at document ranks 5, 10, 15, 20, 50 and 100. We used both the paired t-test and paired Wilcoxon test discussed by Hull [Hul93] for significance testing. Due to the presence of ties in our data, we used an alternate formulation of the Wilcoxon test [Ott93]. There was a high degree of agreement between the two tests—the Wilcoxon results are reported here.

9.2 Results

9.2.1 Single-Collection Scenario

Table 9.1 contains the results for all six environments under the `single` scenario, where all documents from the testbed are located in a single collection. Note that because they contain exactly the same documents, the results for the *SYM-236* and *UDC-236* test environments are identical. The documents contained in *SYM-236* and *UDC-236* are a subset of those contained in *UBC-100*—the overall *SYM-236*/*UDC-236* performance results are very close to the *UBC-100* results.

[2] `trec_eval` is the official evaluation program for the TREC experiments. It is available from NIST via `http://trec.nist.gov` and as part of the SMART information retrieval system.

Precision at Rank	Long Queries		Short Queries	
	UBC-100	*SYM-236* *UDC-236*	*UBC-100*	*SYM-236* *UDC-236*
5 docs	0.642	0.640	0.470	0.482
10 docs	0.609	0.600	0.468	0.486
15 docs	0.596	0.593	0.469	0.477
20 docs	0.582	0.574	0.461	0.461
50 docs	0.538	0.534	0.418	0.424
100 docs	0.495	0.484	0.374	0.379

Table 9.1: Average precision values for the `single` collection scenario for the *UBC-100*, *SYM-236* and *UDC-236* testbeds using both short and long queries.

9.2.2 Multiple Collections vs. Single Collection

Tables 9.2 and 9.3 present a summary of results for both of the multi-collection scenarios over all six test environments. The results using the long query formulations are presented in Table 9.2 while the results for the short query formulations are found in Table 9.3. Within each table, the first sub-table shows the results for the *UBC-100* testbed, the second sub-table the results for the *SYM-236* testbed and the third sub-table the results for the *UDC-236* testbed. Within each sub-table, results for the the two multi-collection scenarios are shown using both *RBR* and *CORI* selection at 2, 5, 10 and 20 collections selected. Overall, the results from the long queries and the short queries show similar performance trends. However, the numeric average precision scores were lower for the short queries.

The typography of Tables 9.2 and 9.3 is used to show the results of a comparison with the corresponding single-collection document retrieval performance. Using a paired Wilcoxon test at $p = 0.05$, items shown in boldface are significantly better than the corresponding `single` performance from Table 9.1, while italicized items are significantly worse. The default typeface denotes no significant difference[3]. Referring back to Hypothesis 1, we find that when very good (*RBR*) selection is employed, it is possible to exceed the corresponding

[3]We used the paired Wilcoxon test as our primary significance test because it does not require an underlying normal distribution. We verified those significance results using a paired t-test at $p = 0.05$. There were differences in determination of significance for only 3% of the comparisons for long queries and 4% for short queries.

UBC 100-collection testbed									
	Precision	RBR selection				CORI selection			
	at Rank	2 sel	5 sel	10 sel	20 sel	2 sel	5 sel	10 sel	20 sel
multi-CWI	5 docs	0.670	0.652	0.670	0.670	*0.509*	*0.528*	*0.570*	*0.602*
	10 docs	0.633	**0.651**	**0.647**	**0.661**	*0.477*	*0.512*	*0.551*	0.588
	15 docs	0.608	**0.637**	**0.633**	**0.643**	*0.451*	*0.498*	*0.531*	*0.567*
	20 docs	0.586	**0.623**	**0.623**	**0.628**	*0.430*	*0.481*	*0.520*	*0.553*
	50 docs	*0.481*	0.551	**0.571**	**0.582**	*0.340*	*0.416*	*0.462*	*0.508*
	100 docs	*0.375*	*0.468*	0.508	**0.523**	*0.259*	*0.348*	*0.398*	*0.452*
multi-LI	5 docs	**0.686**	**0.694**	0.680	0.682	*0.540*	*0.571*	*0.586*	0.610
	10 docs	**0.656**	**0.683**	**0.685**	**0.691**	*0.501*	0.561	*0.567*	0.595
	15 docs	**0.629**	**0.665**	**0.669**	**0.682**	*0.475*	*0.535*	*0.558*	0.593
	20 docs	0.605	**0.653**	**0.659**	**0.673**	*0.454*	*0.508*	*0.546*	0.579
	50 docs	*0.490*	**0.570**	**0.606**	**0.616**	*0.361*	*0.432*	*0.485*	0.529
	100 docs	*0.378*	0.481	**0.531**	**0.556**	*0.265*	*0.358*	*0.412*	*0.473*

SYM 236-collection testbed									
	Precision	RBR selection				CORI selection			
	at Rank	2 sel	5 sel	10 sel	20 sel	2 sel	5 sel	10 sel	20 sel
multi-CWI	5 docs	0.646	0.680	0.674	**0.690**	*0.483*	*0.546*	*0.554*	*0.592*
	10 docs	0.613	**0.643**	**0.653**	**0.673**	*0.464*	*0.499*	*0.527*	*0.555*
	15 docs	0.569	0.620	**0.635**	**0.653**	*0.423*	*0.469*	*0.510*	*0.540*
	20 docs	0.545	0.598	**0.621**	**0.636**	*0.397*	*0.448*	*0.493*	*0.535*
	50 docs	*0.431*	0.513	0.542	**0.580**	*0.287*	*0.368*	*0.416*	*0.472*
	100 docs	*0.309*	*0.414*	0.469	**0.506**	*0.195*	*0.287*	*0.343*	*0.404*
multi-LI	5 docs	**0.700**	**0.718**	**0.704**	**0.726**	*0.538*	*0.568*	*0.554*	0.624
	10 docs	**0.647**	**0.682**	**0.696**	**0.705**	*0.480*	*0.537*	*0.553*	0.586
	15 docs	0.608	**0.656**	**0.663**	**0.698**	*0.435*	*0.506*	*0.536*	0.570
	20 docs	0.572	**0.626**	**0.652**	**0.677**	*0.403*	*0.483*	*0.520*	0.556
	50 docs	*0.440*	0.532	**0.569**	**0.603**	*0.296*	*0.379*	*0.435*	*0.495*
	100 docs	*0.316*	*0.427*	0.484	**0.528**	*0.200*	*0.297*	*0.356*	*0.420*

UDC 236-collection testbed									
	Precision	RBR selection				CORI selection			
	at Rank	2 sel	5 sel	10 sel	20 sel	2 sel	5 sel	10 sel	20 sel
multi-CWI	5 docs	**0.726**	**0.700**	**0.708**	**0.708**	*0.480*	*0.506*	*0.546*	*0.557*
	10 docs	**0.658**	**0.684**	**0.680**	**0.693**	*0.427*	*0.479*	*0.499*	*0.531*
	15 docs	0.604	**0.653**	**0.669**	**0.679**	*0.384*	*0.444*	*0.491*	*0.514*
	20 docs	0.574	**0.623**	**0.650**	**0.666**	*0.346*	*0.419*	*0.468*	*0.493*
	50 docs	*0.386*	0.512	**0.565**	**0.598**	*0.228*	*0.315*	*0.384*	*0.430*
	100 docs	*0.246*	*0.380*	0.462	**0.515**	*0.143*	*0.221*	*0.293*	*0.359*
multi-LI	5 docs	**0.718**	**0.722**	**0.732**	**0.732**	*0.501*	*0.508*	*0.549*	*0.561*
	10 docs	**0.662**	**0.692**	**0.707**	**0.699**	*0.443*	*0.499*	*0.541*	0.547
	15 docs	0.620	**0.676**	**0.681**	**0.693**	*0.400*	*0.460*	*0.513*	*0.530*
	20 docs	0.574	**0.638**	**0.665**	**0.668**	*0.362*	*0.431*	*0.493*	*0.518*
	50 docs	*0.396*	0.520	**0.575**	**0.614**	*0.236*	*0.325*	*0.391*	*0.455*
	100 docs	*0.249*	*0.388*	0.471	**0.526**	*0.147*	*0.228*	*0.303*	*0.370*

Table 9.2: Average precision over 100 long queries achieved in the `multi-CWI` and `multi-LI` scenarios for the *UBC-100*, *SYM-236* and *UDC-236* testbeds. Typeface changes reflect a comparison with `single` performance (bold = significantly better, italics = significantly worse).

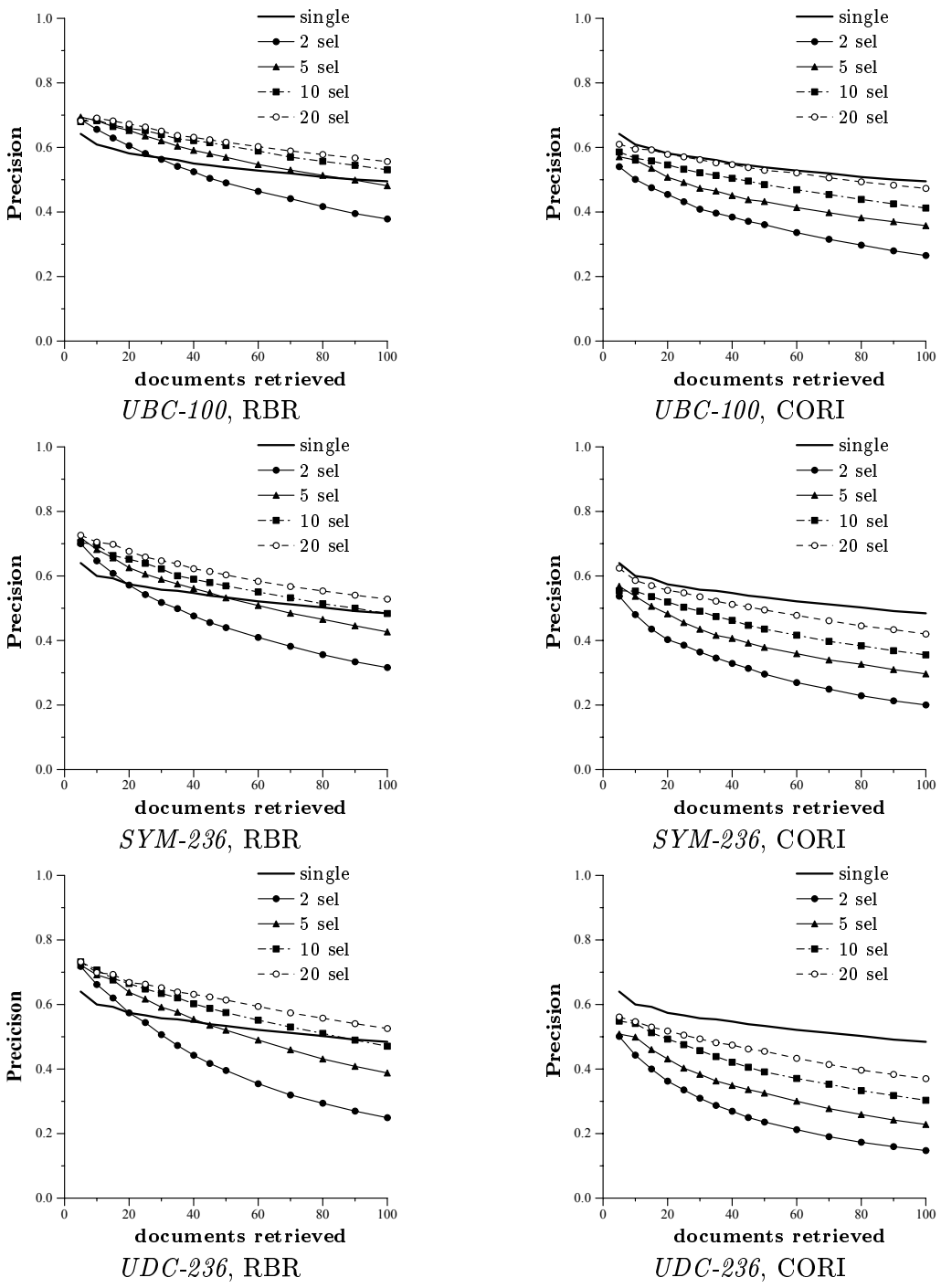


Figure 9.1: Precision values for the `multi-LI` scenario using long queries, both RBR and *CORI* selection approaches and the three testbeds. The values plotted here are those presented in Table 9.2.

UBC 100-collection testbed									
	Precision	RBR selection				CORI selection			
	at Rank	2 sel	5 sel	10 sel	20 sel	2 sel	5 sel	10 sel	20 sel
multi-CWI	5 docs	**0.520**	**0.540**	**0.560**	**0.544**	*0.373*	*0.412*	0.448	0.460
	10 docs	0.486	**0.523**	**0.521**	**0.515**	*0.337*	*0.401*	*0.430*	0.447
	15 docs	0.475	**0.505**	**0.519**	**0.504**	*0.312*	*0.373*	*0.406*	*0.433*
	20 docs	0.452	**0.487**	**0.516**	**0.496**	*0.297*	*0.353*	*0.397*	*0.425*
	50 docs	*0.362*	0.421	**0.450**	**0.452**	*0.232*	*0.293*	*0.338*	*0.383*
	100 docs	*0.278*	0.352	**0.394**	**0.401**	*0.168*	*0.240*	*0.288*	*0.333*
multi-LI	5 docs	**0.528**	**0.532**	**0.568**	**0.560**	*0.362*	0.420	0.462	0.494
	10 docs	0.493	**0.538**	**0.538**	**0.542**	*0.333*	*0.389*	*0.425*	0.458
	15 docs	0.466	**0.515**	**0.539**	**0.527**	*0.310*	*0.362*	*0.411*	0.444
	20 docs	0.449	**0.499**	**0.519**	**0.516**	*0.293*	*0.347*	*0.391*	0.426
	50 docs	*0.361*	0.431	**0.459**	**0.461**	*0.224*	*0.289*	*0.340*	*0.387*
	100 docs	*0.274*	0.357	**0.405**	**0.412**	*0.164*	*0.235*	*0.289*	*0.333*

SYM 236-collection testbed									
	Precision	RBR selection				CORI selection			
	at Rank	2 sel	5 sel	10 sel	20 sel	2 sel	5 sel	10 sel	20 sel
multi-CWI	5 docs	**0.530**	**0.572**	**0.584**	**0.560**	*0.394*	*0.404*	*0.434*	0.484
	10 docs	0.485	**0.517**	**0.543**	**0.539**	*0.324*	*0.377*	*0.403*	0.453
	15 docs	0.458	0.494	**0.523**	**0.525**	*0.298*	*0.347*	*0.391*	*0.436*
	20 docs	0.436	**0.477**	**0.508**	**0.520**	*0.279*	*0.322*	*0.378*	*0.416*
	50 docs	*0.332*	0.404	0.433	**0.461**	*0.203*	*0.259*	*0.311*	*0.359*
	100 docs	*0.237*	*0.318*	0.371	**0.396**	*0.136*	*0.196*	*0.254*	*0.302*
multi-LI	5 docs	**0.530**	**0.574**	**0.574**	**0.570**	*0.390*	*0.402*	*0.422*	0.470
	10 docs	0.488	**0.532**	**0.547**	**0.540**	*0.325*	*0.363*	*0.406*	*0.432*
	15 docs	0.460	**0.503**	**0.519**	**0.526**	*0.291*	*0.339*	*0.377*	*0.411*
	20 docs	0.430	**0.481**	**0.504**	**0.506**	*0.269*	*0.318*	*0.360*	*0.401*
	50 docs	*0.324*	0.399	0.437	**0.453**	*0.197*	*0.252*	*0.304*	*0.348*
	100 docs	*0.235*	*0.314*	0.364	**0.396**	*0.134*	*0.192*	*0.246*	*0.295*

UDC 236-collection testbed									
	Precision	RBR selection				CORI selection			
	at Rank	2 sel	5 sel	10 sel	20 sel	2 sel	5 sel	10 sel	20 sel
multi-CWI	5 docs	**0.550**	**0.568**	**0.592**	**0.580**	*0.366*	*0.419*	0.436	0.478
	10 docs	0.504	**0.528**	**0.555**	**0.585**	*0.312*	*0.369*	*0.402*	0.454
	15 docs	0.467	**0.517**	**0.534**	**0.564**	*0.281*	*0.341*	*0.383*	*0.431*
	20 docs	0.430	**0.498**	**0.517**	**0.547**	*0.254*	*0.318*	*0.364*	*0.409*
	50 docs	*0.299*	0.402	**0.454**	**0.479**	*0.166*	*0.233*	*0.289*	*0.339*
	100 docs	*0.198*	*0.294*	0.359	**0.412**	*0.106*	*0.163*	*0.221*	*0.277*
multi-LI	5 docs	**0.536**	**0.572**	**0.604**	**0.600**	*0.361*	*0.400*	0.434	0.478
	10 docs	0.499	**0.531**	**0.550**	**0.581**	*0.318*	*0.361*	*0.390*	*0.433*
	15 docs	0.463	**0.510**	**0.535**	**0.549**	*0.280*	*0.335*	*0.377*	*0.404*
	20 docs	0.430	0.491	**0.520**	**0.533**	*0.255*	*0.314*	*0.358*	*0.396*
	50 docs	*0.300*	0.396	**0.445**	**0.472**	*0.164*	*0.229*	*0.281*	*0.329*
	100 docs	*0.196*	*0.293*	0.356	**0.404**	*0.106*	*0.160*	*0.216*	*0.267*

Table 9.3: Average precision over 100 short queries achieved in the `multi-CWI` and `multi-LI` scenarios for the *UBC-100*, *SYM-236* and *UDC-236* testbeds. Typeface changes reflect a comparison with `single` performance (bold = significantly better, italics = significantly worse).

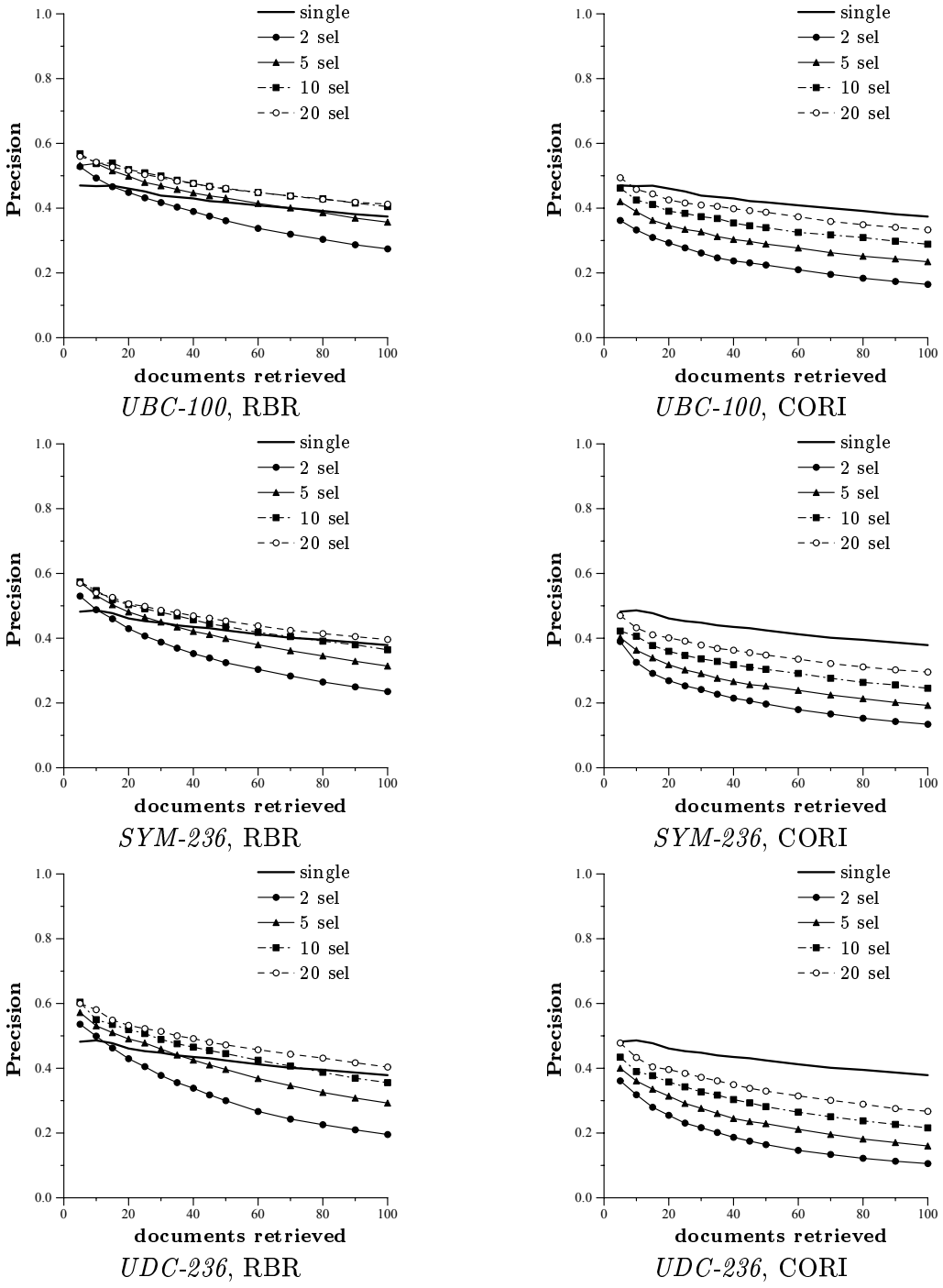


Figure 9.2: Precision values for the `multi-LI` scenario using short queries, both the RBR and *CORI* selection approaches and the three testbeds. The values plotted here are those presented in Table 9.3.

single-collection performance in all six test environments. Referring to the first portion of Hypothesis 2, we see that it is possible to meet or exceed single-collection performance even when a small number of collections are selected using *RBR* selection. We do, however, see decreased effectiveness at high document ranks when a very small number (2 or 5) of collections are selected. This decreased effectiveness is due to a combination of effects. The first effect is a phenomenon that concerned Xu and Croft [XC99]—for some queries, there are very few relevant documents to be found in the top-ranked 2 or 5 collections. The second effect is an aspect of the evaluation approach. When very few relevant documents are available in the top 2 or 5 collections, there exist queries for which all available relevant documents may be retrieved in the top 10 or 20 documents. However, because all 100 retrieved documents are evaluated, precision at the 50 or 100 document cutoff for these queries will be very low. These queries can depress the average precision values for high document cutoff values.

The results of the `multi-LI` portions of Tables 9.2 and 9.3 are also shown in Figures 9.1 and 9.2. Figures 9.1 and 9.2 provide a visual illustration of the potential to exceed `single` performance when RBR selection is used. The average performance curve when 20 collections are selected using RBR is consistently above the single-collection performance curve for all six test environments.

When currently achievable (*CORI*) selection is employed, the results tend to be significantly worse than the corresponding `single` performance (see the right-hand columns of Tables 9.2 and 9.3 and the CORI results from Figures 9.1 and 9.2. However, the results do approach `single` performance when 20 collections are selected and for some cases there is no significant difference between the approaches.

9.2.3 The Effect of Selecting More Collections

In the previous section, we noted that it is possible for both `multi-CWI` and `multi-LI` to exceed `single` performance, even when a small number of collections are selected for search, confirming the first portion of Hypothesis 2. We now turn our attention to the

second portion of that hypothesis. We are interested in the impact of selecting more or fewer collections to search. Examining Tables 9.2 and 9.3, we noted that at each document rank level, increasing the number of collections selected for search tended to (but did not always) improve document retrieval performance. Tables 9.4 and 9.5 address the question of whether the observed improvement was significant.

In Tables 9.4 and 9.5, items in boldface type are significantly better (using a paired Wilcoxon test at $p = 0.05$) than the corresponding item in the column immediately to the left. Bold-italic denotes cases for which an item is significantly better than the corresponding item in *some* column to the left. The default typeface denotes no significant difference. Note that this typography convention is different from the convention used in Tables 9.2, 9.3, 9.6 and 9.7.

There are a number of interesting things to observe in Tables 9.4 and 9.5. First, when *CORI* is used for selection, selecting a larger number of collections for search tends to be advantageous. This is understandable given that, while its collection selection performance is good, *CORI* is not guaranteed to select collections with the most (or even any) relevant documents. In this case, selecting more collections increases the chances of selecting a relevant-rich collection. The beneficial effect of selecting additional collections when *CORI* selection is employed is also illustrated in Figures 9.1 and 9.2.

When *RBR* is used for selection, the greatest improvement can be seen when 5 or 10 collections are selected (instead of 2). This can be seen in both Tables 9.4 and 9.5 and in Figures 9.1 and 9.2. This may be due to the effect discussed in Section 9.2.2—there are queries for which there are few relevant documents to be found in the top 2 collections. Searching a larger number of collections increases the number of available relevant documents. The lesser improvement when 20 collections are selected can be explained by a similar phenomenon—there also exist queries for which many relevant documents can be found in the top 10 selected collections. For these queries, searching a larger number of collections does not provide a large benefit.

	Precision at Rank	RBR selection 2 sel	RBR selection 5 sel	RBR selection 10 sel	RBR selection 20 sel	CORI selection 2 sel	CORI selection 5 sel	CORI selection 10 sel	CORI selection 20 sel
UBC-100	5 docs	0.686	0.694	0.680	0.682	0.540	0.571	***0.586***	***0.610***
	10 docs	0.656	0.683	0.685	0.691	0.501	**0.561**	***0.567***	**0.595**
	15 docs	0.629	**0.665**	***0.669***	***0.682***	0.475	**0.535**	***0.558***	**0.593**
	20 docs	0.605	**0.653**	***0.659***	***0.673***	0.454	**0.508**	***0.546***	**0.579**
	50 docs	0.490	**0.570**	**0.606**	***0.616***	0.361	**0.432**	**0.485**	**0.529**
	100 docs	0.378	**0.481**	**0.531**	**0.556**	0.265	**0.358**	**0.412**	**0.473**
SYM-236	5 docs	0.700	0.718	0.704	0.726	0.538	0.568	0.554	***0.624***
	10 docs	0.647	**0.682**	***0.696***	***0.705***	0.480	**0.537**	***0.553***	***0.586***
	15 docs	0.608	**0.656**	***0.663***	**0.698**	0.435	**0.506**	**0.536**	**0.570**
	20 docs	0.572	**0.626**	**0.652**	**0.677**	0.403	**0.483**	**0.520**	**0.556**
	50 docs	0.440	**0.532**	**0.569**	**0.603**	0.296	**0.379**	**0.435**	**0.495**
	100 docs	0.316	**0.427**	**0.484**	**0.528**	0.200	**0.297**	**0.356**	**0.420**
UDC-236	5 docs	0.718	0.722	0.732	0.732	0.501	0.508	***0.549***	***0.561***
	10 docs	0.662	**0.692**	***0.707***	***0.699***	0.443	**0.499**	**0.541**	***0.547***
	15 docs	0.620	**0.676**	***0.681***	***0.693***	0.400	**0.460**	**0.513**	***0.530***
	20 docs	0.574	**0.638**	**0.665**	***0.668***	0.362	**0.431**	**0.493**	**0.518**
	50 docs	0.396	**0.520**	**0.575**	**0.614**	0.236	**0.325**	**0.391**	**0.455**
	100 docs	0.249	**0.388**	**0.471**	**0.526**	0.147	**0.228**	**0.303**	**0.370**

Table 9.4: The impact of selecting more or fewer collections for search using scenario `multi-LI`. Long queries. Bold = significantly better than item directly to left, Bold italic = better than *some* item to left.

9.2.3.1 More isn't always better

Finally, we should point out that while searching additional collections tends to improve retrieval performance in Tables 9.4 and 9.5, there are limits to that trend. When all collections that contain relevant documents have been selected, no additional improvement will be seen.

In fact, beyond a certain point, searching additional collections may degrade performance. For example, consider the `multi-CWI` portions of Tables 9.2 and 9.3 when *RBR* is used for collection selection. For all six test environments, there are numerous cases for which `multi-CWI` outperforms `single`. However, given the construction of `multi-CWI`, when all collections are selected the performance (when up to 100 documents are retrieved) will be exactly that of `single`. At some point between selecting 20 collections and selecting all of them, performance began to degrade.

	Precision	RBR selection				CORI selection			
	at Rank	2 sel	5 sel	10 sel	20 sel	2 sel	5 sel	10 sel	20 sel
UBC-100	5 docs	0.528	0.532	0.568	0.560	0.362	**0.420**	***0.462***	***0.494***
	10 docs	0.493	**0.538**	***0.538***	***0.542***	0.333	**0.389**	**0.425**	**0.458**
	15 docs	0.466	**0.515**	***0.539***	***0.527***	0.310	**0.362**	**0.411**	**0.444**
	20 docs	0.449	**0.499**	***0.519***	***0.516***	0.293	**0.347**	**0.391**	**0.426**
	50 docs	0.361	**0.431**	**0.459**	***0.461***	0.224	**0.289**	**0.340**	**0.387**
	100 docs	0.274	**0.357**	**0.405**	***0.412***	0.164	**0.235**	**0.289**	**0.333**
SYM-236	5 docs	0.530	0.574	0.574	0.570	0.390	0.402	0.422	***0.470***
	10 docs	0.488	**0.532**	***0.547***	***0.540***	0.325	0.363	**0.406**	***0.432***
	15 docs	0.460	**0.503**	**0.519**	***0.526***	0.291	0.339	**0.377**	**0.411**
	20 docs	0.430	**0.481**	**0.504**	***0.506***	0.269	**0.318**	**0.360**	**0.401**
	50 docs	0.324	**0.399**	**0.437**	**0.453**	0.197	**0.252**	**0.304**	**0.348**
	100 docs	0.235	**0.314**	**0.364**	**0.396**	0.134	**0.192**	**0.246**	**0.295**
UDC-236	5 docs	0.536	0.572	***0.604***	***0.600***	0.361	0.400	***0.434***	***0.478***
	10 docs	0.499	**0.531**	***0.550***	**0.581**	0.318	0.361	**0.390**	***0.433***
	15 docs	0.463	**0.510**	**0.535**	***0.549***	0.280	**0.335**	**0.377**	***0.404***
	20 docs	0.430	**0.491**	**0.520**	***0.533***	0.255	**0.314**	**0.358**	***0.396***
	50 docs	0.300	**0.396**	**0.445**	**0.472**	0.164	**0.229**	**0.281**	**0.329**
	100 docs	0.196	**0.293**	**0.356**	**0.404**	0.106	**0.160**	**0.216**	**0.267**

Table 9.5: The impact of selecting more or fewer collections for search using scenario `multi-LI`. Short queries. Bold = significantly better than item directly to left, Bold italic = better than *some* item to left.

9.2.4 The Effect of Collection-Wide Information

In addition to comparisons of the two multi-collection approaches to the single-collection approach, we were also interested in the relative performance of `multi-CWI` and `multi-LI`. Examining Table 9.2, we noted that under a strict numeric comparison, `multi-LI` often outperformed `multi-CWI`. The question of whether the difference between `multi-CWI` and `multi-LI` was significant remained. Table 9.6 repeats the `multi-LI` sections of Table 9.2 for all three testbeds. Here, however, the numbers in boldface denote cases where the `multi-LI` performance is significantly greater than the corresponding `multi-CWI` performance. In this table, there is no case for which `multi-LI` performance is significantly worse than the corresponding `multi-CWI` performance. The results using short queries were less clear-cut. Examining Table 9.3, we find that the differences in `multi-CWI` and `multi-LI` performance tended to be mixed, with neither scenario holding a clear advantage. Table 9.7 repeats the `multi-LI` sections of Table 9.3 for all three testbeds with significance denoted by changes in typeface. For the short queries, it is generally the case that there is no significant difference between `multi-CWI` and `multi-LI`. Overall, for all six test environments, we find that Hypothesis 3 is false.

9.3 Discussion

9.3.1 CWI and Merging Analysis

An issue that deserves immediate attention is the apparent contradiction of this work and the work of Viles and French [VF95b, FV96]. Based on the work of Viles and French, we expected that Hypothesis 3 would be true (i.e., the use of CWI would improve multi-collection retrieval performance); however, the `multi-LI` results were significantly better than the `multi-CWI` results for long queries and generally not significantly different for short queries.

Our initial reaction was that the difference in the `multi-CWI` and `multi-LI` results was due to the difference in the merging step for the two scenarios. `multi-CWI` used a

	Precision	RBR selection				CORI selection			
	at Rank	2 sel	5 sel	10 sel	20 sel	2 sel	5 sel	10 sel	20 sel
UBC-100	5 docs	0.686	**0.694**	0.680	0.682	**0.540**	**0.571**	0.586	0.610
	10 docs	0.656	**0.683**	**0.685**	**0.691**	**0.501**	**0.561**	0.567	0.595
	15 docs	**0.629**	**0.665**	**0.669**	**0.682**	**0.475**	**0.535**	**0.558**	**0.593**
	20 docs	**0.605**	**0.653**	**0.659**	**0.673**	**0.454**	**0.508**	**0.546**	**0.579**
	50 docs	0.490	**0.570**	**0.606**	**0.616**	**0.361**	**0.432**	**0.485**	**0.529**
	100 docs	0.378	**0.481**	**0.531**	**0.556**	**0.265**	**0.358**	**0.412**	**0.473**
SYM-236	5 docs	**0.700**	**0.718**	**0.704**	**0.726**	**0.538**	0.568	0.554	**0.624**
	10 docs	**0.647**	**0.682**	**0.696**	**0.705**	0.480	**0.537**	**0.553**	**0.586**
	15 docs	**0.608**	**0.656**	**0.663**	**0.698**	0.435	**0.506**	**0.536**	**0.570**
	20 docs	**0.572**	**0.626**	**0.652**	**0.677**	0.403	**0.483**	**0.520**	0.556
	50 docs	0.440	**0.532**	**0.569**	**0.603**	**0.296**	0.379	**0.435**	0.495
	100 docs	**0.316**	**0.427**	**0.484**	**0.528**	**0.200**	**0.297**	**0.356**	**0.420**
UDC-236	5 docs	0.718	0.722	0.732	0.732	0.501	0.508	0.549	0.561
	10 docs	0.662	0.692	**0.707**	0.699	0.443	0.499	**0.541**	0.547
	15 docs	0.620	**0.676**	0.681	0.693	**0.400**	0.460	**0.513**	0.530
	20 docs	0.574	**0.638**	0.665	0.668	**0.362**	0.431	**0.493**	**0.518**
	50 docs	**0.396**	0.520	0.575	**0.614**	**0.236**	0.325	0.391	**0.455**
	100 docs	0.249	**0.388**	**0.471**	**0.526**	0.147	**0.228**	**0.303**	**0.370**

Table 9.6: Is `multi-LI` significantly better than `multi-CWI`? Long queries. Typeface changes — bold indicates `multi-LI` significantly better.

	Precision	RBR selection				CORI selection			
	at Rank	2 sel	5 sel	10 sel	20 sel	2 sel	5 sel	10 sel	20 sel
UBC-100	5 docs	0.528	0.532	0.568	0.560	0.362	0.420	0.462	**0.494**
	10 docs	0.493	0.538	**0.538**	**0.542**	0.333	0.389	0.425	0.458
	15 docs	0.466	0.515	**0.539**	**0.527**	0.310	0.362	0.411	0.444
	20 docs	0.449	**0.499**	0.519	**0.516**	0.293	0.347	0.391	0.426
	50 docs	0.361	**0.431**	**0.459**	**0.461**	*0.224*	0.289	0.340	0.387
	100 docs	0.274	0.357	**0.405**	**0.412**	0.164	0.235	0.289	0.333
SYM-236	5 docs	0.530	0.574	0.574	0.570	0.390	0.402	0.422	0.470
	10 docs	0.488	0.532	0.547	0.540	0.325	0.363	0.406	0.432
	15 docs	0.460	0.503	0.519	0.526	0.291	0.339	0.377	*0.411*
	20 docs	0.430	0.481	0.504	0.506	*0.269*	0.318	*0.360*	0.401
	50 docs	*0.324*	0.399	0.437	0.453	*0.197*	*0.252*	0.304	0.348
	100 docs	0.235	0.314	*0.364*	0.396	0.134	*0.192*	*0.246*	*0.295*
UDC-236	5 docs	0.536	0.572	0.604	0.600	0.361	0.400	0.434	0.478
	10 docs	0.499	0.531	0.550	0.581	0.318	0.361	0.390	*0.433*
	15 docs	0.463	0.510	0.535	0.549	0.280	0.335	0.377	*0.404*
	20 docs	0.430	0.491	0.520	0.533	0.255	0.314	0.358	0.396
	50 docs	0.300	0.396	0.445	0.472	0.164	*0.229*	*0.281*	*0.329*
	100 docs	*0.196*	0.293	0.356	0.404	0.106	*0.160*	*0.216*	*0.267*

Table 9.7: Is `multi-LI` significantly better than `multi-CWI`? Short queries. Typeface changes — bold indicates `multi-LI` significantly better.

raw-score merge while `multi-LI` used the default *CORI* merge. We speculated that the incorporation of the collection score into the *CORI* merge contributed to the performance difference. Therefore, we replaced the raw score merge used in the `multi-CWI` case with the *CORI* merge, creating `multi-CWI-CM`.

When we compared `multi-CWI-CM` and `multi-CWI`, we found that the performance of `multi-CWI` was either the same as or better than the performance of `multi-CWI-CM`, eliminating the merge explanation. However, there are additional differences between the `multi-CWI` and `multi-LI` experiments and between our experiments and those of Viles and French that help explain the results. First, Viles and French were investigating a different problem. They showed that when a query was broadcast to *all* collections, a raw score merge using CWI is better than raw score merge using only local collection information. Second, Viles and French showed that the usefulness of CWI is related to characteristics of the collections. Third, and related to the first point, `multi-CWI` and `multi-LI` represent different ways to make the document scores from different collections comparable. In `multi-CWI`, the use of CWI makes the document scores directly comparable. The intent of the D'' normalization step in `multi-LI` is also to make the document scores from different collections comparable. In `multi-LI`, the general interpretation is that documents that scored well within their collection and that also came from highly-ranked collections should be ranked highly.

However, these differences should not be allowed to distract from the take-home message. Given a `multi-CWI` or `multi-LI` scenario, very good collection performance enables very good document retrieval performance. Currently-achievable collection selection performance enables document retrieval performance on par with single-collection; better selection can enable multi-collection information retrieval performance to exceed performance in an equivalent single-collection environment.

Singhal *et al.* [SAM+99] found similar results in their experiments dividing the 100GB TREC-8 VLC2 corpus into 20 5GB collections. They compared results using CWI to a raw score merge and found very little difference. They concluded that when a very large

collection was subdivided into smaller collections CWI was not necessary. We found that this is the case for these experiments but will show in Chapter 10 that CWI can be useful in some environments.

9.3.2 Distribution of Relevant Documents

Xu and Croft [XC99] expressed concern that the distribution of relevant documents in multiple collections organized by publication source or collection size might hinder multi-collection retrieval performance. As a result, we examine that issue as it applies to our test environments.

Table 9.8 summarizes the distributions of relevant documents in the *UBC-100*, *SYM-236* and *UDC-236* testbeds. The number of collections containing relevant documents, and the distribution of those relevant documents is tied to the relevance judgements $\mathcal{J}_{TREC4}$ for the TREC topics. Because the short and long queries were generated from the TREC topics, these distributions apply to both the short and long queries. The values shown here are computed over TREC topics 51-150. The first data column, labelled *Average* n^* is simply the average (over all 100 queries) of the number of collections that contain at least one relevant document. The remaining three data columns summarize the distribution of relevant documents. For each query, we divided the total number of relevant documents by the n^* value for that query. We report the minimum, maximum and average values for that ratio over all 100 queries. Table 9.8 can be considered in conjunction with Figures 4.3–4.5 as characterization of the number of relevant documents per collection within the testbeds.

Note that in Table 9.8, the *UBC-100* testbed tends to have both more collections with relevant documents and more relevant documents per collection. However, recall that the *UBC-100* testbed contains more documents per collection. Also note that relevant documents are more evenly distributed in the *UDC-236* testbed than in *SYM-236* and *UBC-100*.

While Xu and Croft did report better performance when using clustered collections, we have not observed difficulties of the type Xu and Croft [XC99] predicted. For our

Testbed	Average n*	Relevant Documents per Collection		
		Min.	Avg.	Max.
UBC-100	51.6	1.9	8.7	26.8
SYM-236	76.7	1.3	5.1	15.4
UDC-236	111.0	1.1	3.4	8.8

Table 9.8: Summary statistics for the testbeds.

three testbeds, the distribution of relevant documents does not appear to have had a large impact on the overall retrieval performance. Each testbed has different relevant document distribution characteristics; however, the overall performance for the three testbeds was similar (see Tables 9.2 and 9.3).

9.3.3 Conceptual Subdivisions of Collections

In these experiments, the `multi-CWI` scenario was considered in the context of an existing multi-collection environment. However, given the potential of multi-collection retrieval to outperform single-collection retrieval, we consider an alternate interpretation.

Given an existing large single collection, the documents in that collection could be conceptually organized into a "multi-collection" arrangement. The physical storage, organization and indexing of the documents need not change, but each document would be assigned to a conceptual "pseudo-collection". Queries could be handled at the collection as usual, producing a single result list. The impact of the conceptual "multi-collection" organization could be realized as a post-processing step. Given a query, a collection selection step could be added, performed on the languages models representing the pseudo-collections. Documents from the selected pseudo-collections would be declared eligible for retrieval. Only eligible documents would be presented to a user.

The effects of this conceptual organization are clarified by an example. Assume that we have a single large indexed collection for which documents have been conceptually organized into a set of pseudo-collections $\{A, B, C, D, E, F, G, H, I, J\}$. Consider the lefthand side of

	Ranked List	Score	Relevant?		Ranked List	Score	Relevant?
•	A57	0.97	1		A57	0.97	1
	B17	0.95	0		C05	0.82	1
	F22	0.90	0	$\longrightarrow$	G33	0.75	0
•	C05	0.82	1		A93	0.72	1
	B80	0.81	1		C68	0.67	1
	D15	0.77	0				
	H45	0.76	0				
•	G33	0.75	0				
•	A93	0.72	1				
•	C68	0.67	1				

Figure 9.3: An example use of a conceptual "multi-collection" arrangement. In this example, documents from pseudo-collections A, C and G are declared eligible for retrieval.

Figure 9.3 which shows a ranked list of documents[4] for some query, the document score and the relevance of the retrieved documents. Assume that we select pseudo-collections A, C and G, and declare documents assigned to those collections eligible for retrieval. The documents from the other collections would simply be elided from the results list, producing the modified results list on the righthand side of Figure 9.3. Note that the similarity scores and order of the remaining documents remains the same. In our example, precision at five documents retrieved improves from 0.6 to 0.8, despite losing relevant document B80 from pseudo-collection B which was not selected.

Our results from the `multi-CWI` experiments suggest that this post-processing step could improve the quality of the result-list. In order to achieve this, however, improvements in collection selection performance are necessary. With existing approaches, we have seen that it is possible to equal single-collection performance. It was only with the best-case selection scenario that we saw improvements over single-collection performance.

[4]For clarity, we denote the pseudo-collection assignment in the document ID. That need not be the case in an operational environment.

9.4 Summary

In this chapter we have reported experiments that support the following conclusions.

- When very good selection is employed, multi-collection retrieval can outperform retrieval in an equivalent single-collection environment.
- It is possible to achieve good document retrieval performance by selecting a small number of heterogeneous collections. Selecting more collections does improve performance (up to a point).
- Given very good selection, *conceptually* decomposing a single collection into multiple collections and interposing a selection step has the potential to improve performance.
- The use of collection-wide information is a complex issue. Given the scenario in the bullet above, the straightforward approach of using already-available CWI plus a raw score merge produces good results. However, given a pre-existing multi-collection environment, using local information works well if collection selection is employed and the document scores are suitably normalized before merging is performed.

By examining six test environments utilizing three different document testbeds we have also shown that these conclusions have wide applicability. There are several implications to these conclusions. First, single-collection performance is not necessarily the gold standard that we should be aiming for. It is possible for multi-collection searches to achieve better retrieval performance. Second, we can get good retrieval performance when only a few collections are selected. This implies that multi-collection searching with good collection selection should scale well. Third, we can conceptually decompose a single collection into subcollections and by introducing a selection step it is possible to achieve better performance than by searching (ranking) the entire collection. So we find that selection plus ranking has the potential to improve the effectiveness of ranking alone. Moreover, we can use a simple raw score merge in this case and nothing more elaborate. Fourth, local information is adequate for good retrieval performance when good collection selection is

employed. This means that it is unnecessary to disseminate collection-wide information when selection is a part of the search strategy.

We set out to examine the effect of collection selection on end-user retrieval performance. Previous work focused on explicit evaluation of the collection selection technique. Our work sought to determine the degree to which collection selection would have an impact on retrieval performance. We believe that we met our goal and have provided concrete conclusions that can usefully guide the engineering of large-scale multi-collection information retrieval systems.

10

Applicability to the WWW

Our multi-collection information retrieval experiments detailed in Chapter 9 suggest that information retrieval performance in a multi-collection environment can meet or exceed performance in an equivalent single-collection environment. In this chapter, we address the question of the broader applicability and effectiveness of the approach. Here we show a straightforward application of the multi-collection information retrieval approaches described in Chapters 7 and 9 to an operational WWW-based environment. We also show how to extend the approach to metasearching both when the search engines will supply collections statistics to the metasearcher and when they will not. In the latter case we show how to develop language models [CCD99, XC99] for the WWW-based collections using query-based sampling.

The question of the effectiveness of information retrieval systems on the WWW has been addressed to some extent by other researchers. Hawking *et al.* [HCTH99] found that for TREC queries internet search engines operating on a "live" version of the WWW exhibited poorer performance than official TREC results of experimental information retrieval systems operating on an older snapshot of a subset of the WWW. Hawking *et al.* note that the different document sets make direct comparisons difficult, but that the internet search engines had the advantage of more and more current data. Also of interest, results of the TREC-8 Web Track showed a strong correlation between information retrieval system

performance on a standard TREC task and for the Web Track [HVCB99].

Previous experiments have focused on the ability of information retrieval systems to handle the volume and type of data found in the WWW. The experiments presented here study the impact of adding a collection selection step to a WWW environment with a pre-existing organization of documents into collections.

Obviously, the experiments we conduct here cannot make use of the six experimental environments used in Chapters 7–9. As a result, we first describe a new experimental environment and discuss some of the issues that are particular to that environment. We found it necessary to address issues related to language-model building, query set construction, duplicate page removal and the acquisition of relevance judgements. We then describe our experiments as they relate to traditional metasearching and what we term *fine-grained metasearching*. We present some preliminary results, then conclude with lessons learned and future work.

10.1 The Conceptual Search Model

Information retrieval as conducted by the current search engines on the WWW is achieved by treating all of the pages indexed by the search engine as a single monolithic collection. However, recent research outlined in Chapter 9 suggests that in a multi-collection environment it is possible to achieve performance comparable to, and in some cases better than, the performance obtained by treating all of the indexed pages as a single collection. The crucial observation here is that the gain in effectiveness is not due to the fact that the data is distributed across different servers, rather it is due entirely to the fact that the data is *decomposed into subcollections* [PFC+00]. The decomposition lets us discard much useless data while confining the search to more relevant-rich data. This potential to improve retrieval effectiveness has also been observed by Craswell *et al.* [CBH00].

In the general case, we can take advantage of this phenomenon if we are able to find a suitable decomposition of a single collection into multiple collections and then employ

a collection selection step prior to document ranking. In a WWW-based environment, there are a number of scenarios that may allow us to take advantage of a multi-collection organization of documents. First, the overlap between the existing internet search engines tends to be small [LG99]. As a result, the potential exists to apply a collection selection approach to a traditional metasearch environment, in which each collection $C \in \mathcal{C}$ is an existing internet search engine[1]. A second option is to find a suitable organization of the documents of a search engine or other searchable WWW source and to decompose the source into multiple subcollections.

Given a suitable organization of documents into multiple collections, the overall operation of a WWW-based multi-collection system is conceptually very similar to the general approach described in Chapter 2. Given a set of collections $\mathcal{C}$ to which we might send a query, we require a set of language models $\mathcal{LM}$ to be used by a collection selection approach. We use the language models to rank the collections, then send the queries to the selected collections. The individual results lists from each collection are then merged into a single list of results for presentation to a user. In an operational environment, $\mathcal{C}$ is unlikely to be a partition, raising the issues that we touched on in Chapter 2. We will revisit the problem of duplicate data items in a moment.

So far we have not discussed the manner in which we arrive at a decomposition of a collection C into subcollections. For arbitrary collections, this is an open research question. However, certain operational constraints suggest an appropriate partition for extant search engines. We discuss this further in Section 10.3.1.

10.2 Fine-Grained Metasearching

We suggest that a combination of the two collection-division strategies described above may be advantageous. Given a set of search engines in a traditional metasearch environment, we suggest that the content of the search engines be subdivided so that a finer-grained

[1]Note that $\mathcal{C}$ is highly unlikely to be a partition and the union of all documents contained in $\mathcal{C}$ represents only a fraction of the documents available via the WWW.

selection may be made. Rather than targeting a query to a search engine, we select specific subcollections of the indexed content. The intuition here is that we are trying to eliminate as much irrelevant material as possible from consideration. In this way we hope to increase the overall retrieval effectiveness for a query.

Many of the large search engines operating today offer a category structure as part of their service. While organized as a browsing aid, the search engines often provide a means to search for documents within the category organization. In our work, we used this organization as a subdivision of the search engines.

To achieve this described level of selectivity among search engine subcollections, a metasearcher must have available a language model for each candidate component. In some environments the required metadata might be available directly from search engines, for example the Networked Computer Science Technical Reference Library (NCSTRL) provides an open protocol for requesting metadata [DL00]. It is more likely, however, that the required information is not readily available. In this latter case we will build language models by *query-based sampling* [CPFC00].

10.3 Constructing a WWW-Based Test Environment

A WWW-based TREC environment has been under development for some time now. The environment is based upon a roughly 300GB spider crawl of the Internet. From that, a 100GB subset was extracted to form the TREC-7 VLC2 collection [HCT98]. For TREC-7, much of the focus was on efficiency and the ability of systems to handle collections of that size. TREC-8 offered optional "Large Web" and "Small Web" tracks using the VLC2 collection and a smaller 2GB subset respectively [HCTH99, HVCB99]. These tracks focused more on the effectiveness of the participating systems for WWW data. Further refinements of the collections have been made for TREC-9. We are considering experiments using the TREC-9 data, but it is not yet widely available. While the TREC-9 collections are interesting because of their scope, the lack of current availability was problematic. We performed

the initial experiments reported in this chapter using a different test environment.

In this section, we describe the test environment that we created for these experiments. Because we essentially constructed this test environment from scratch for these experiments, we will need to cover a number of points that were not issues for our previous test environment. First, we will describe the collections and their language models and how each was constructed. We will discuss our choice of queries as well as how we obtained relevance judgements.

Unlike our previous test environments, the complete set of documents $\mathcal{D}$ and the exact contents of the collections $C \in \mathcal{C}$ are unknowable. Having chosen some set of collections $\mathcal{C}$, our only access to the collections and the documents within is via their search mechanism.

10.3.1 Using Categories to Subdivide a Search Engine's Content

For this study we adopted the browsable top-level category structure provided by four search engines as the subdivision of the search engines' content into a set of collections. This is only one possible subdivision of the data; however, for our purposes it serves as a convenient assumption, giving us access to topically subdivided data. This organization has a feature that was not found in the three testbeds that we used for previous experiments. In this test environment, web pages have been organized by topic. As a result, we might expect relevant documents to occur in only a few collections, and might expect the documents within a given collection to be self-similar. We can also qualitatively judge the quality of collection selection output.

In deciding upon the set of collections to be used, we originally examined the category structure of six search engines (Altavista, Infoseek, Lycos, NorthernLight, Snap, and Yahoo!) chosen because each had query syntax for restricting searches to specific categories, a necessary requirement of our methodology. However, because we are also interested in traditional metasearching, we also require that the search engine provide a means to restrict a query to documents contained in the category tree as a whole. We will discuss this further in a moment. Snap and Lycos did not offer the latter capability and were later excluded

Altavista (15)	**Infoseek (19)**	**NorthernLight (16)**	**Yahoo (14)**
arts	arts & humanities	arts	arts
autos	automotive		
business & finance	business	business & investing	business & economy
	careers		
computers	computing	computing & internet	computers & internet
		contemporary life	
	education	education	education
	entertainment	entertainment	entertainment
	family		
games			
	gov't & politics	gov't, law & politics	government
health & fitness	health	health & medicine	health
home & family			
		humanities	
internet			
	living		
	marketplace		
	money		
news & media	news		news & media
	people		
		products & services	
	real estate		
recreation & travel			recreation & sports
reference		reference	reference
regional			regional
science	science	science & mathematics	science
		social sciences	social science
society & culture			society & culture
sports	sports	sports & recreation	
		technology	
	travel	travel	

Table 10.1: Category structure of the four search engines used in this chapter.

from our experiments. The categories from the remaining four search engines, available as of June 2000 are shown in Table 10.1. Each search engine was treated as having been subdivided into its categories, giving rise to 64 collections available for selection by our metasearcher. This is in contrast to conventional metasearching strategy that would select from among the four search engines used in our study. Note that this subdivision strategy results in the search engines being decomposed into different numbers of subcollections: Altavista (15), Infoseek (19), NorthernLight (16) and Yahoo! (14). This is not a problem nor is it a requirement of our approach.

It is tempting to attempt to infer category content from the category labels. Table 10.1 reinforces that temptation by attempting to place "like" categories on the same row. However, this is simply a value judgement without any appeal to the content. In this work we simply assume that similar category names imply similar content. That is a reasonable assumption for our purposes.

It is important to note one feature of our strategy for acquiring subdivided data. If we compared our category-based fine-grained metasearching to traditional metasearching using the four search engines, we would be implicitly assuming that the union of the pages represented within the category structure is equal to the full content indexed by the search engine. At the present time this is not true for any of the search engines in our study. This is why we require the search engines to support a search of the documents in the entire category tree. We allow this search to stand in for the search engine as a whole. This allows us to make a fair comparison between the approaches, but means that our "traditional metasearching" approach has access to far fewer documents than are indexed by the search engines.

10.3.2 Processing Queries in the Multi-Collection Environment

In the conventional metasearching environment, query processing has three steps:

1. identify the search engine(s) to which the query should be sent;

2. send the query to the search engine(s) identified and get a result list from each; and

3. merge the result lists according to some strategy for presentation to the user.

The modification to query processing in our fine-grained metasearching environment is straightforward. The first two steps become:

1′. identify the subcollections of the search engines over which the query should be processed; and

2′. process the query at each of the selected subcollections.

The third step remains the same.

Note that there are two ways to interpret step 2′. First, we can process the query by sending it to a search engine while requiring the search engine to restrict the search to a specific category. This strategy might require repeated searches at the same search engine with different category restrictions. Second, we can process the query by sending it to a search engine and requiring the search engine to restrict the search to a list of categories, i.e., those identified in step 1′. The former approach is presently supportable with extant search engines, the latter approach is not.

The first approach could, in principle, result in 64 separate searches in our test environment. The second approach would never require more than four. The effectiveness of the results could be directly affected by the choice of approaches. The first approach offers the best granularity for merging; any score normalization performed during the merge step would be under the control of the metasearcher. The second approach potentially has the edge in efficiency at the possible expense of some retrieval effectiveness; the way in which document scores for the documents from different collections were computed and/or possibly already merged may be unknown. These issues need further study.

As a concrete example, if we are given the query "*How do you prevent and treat Lyme disease?*" we would expect our metasearcher to focus on the categories *health & fitness* (Altavista), *health* (Infoseek, Yahoo), and *health & medicine* (NorthernLight) from Table 10.1 in preference to many of the other categories.

To summarize, our approach centers around identifying the collections (search engine categories) thought to be of most use in answering a query. This requires generating a language model for each collection and using these models together with a collection selection algorithm. Here we are only interested in collection selection algorithms requiring no user intervention, i.e., that are based solely on examination of the language models. Previous work reported in Chapters 7–9 has shown that the *CORI* [CLC95] collection selection approach performs well. Further, our approach involves generating language models by query-based sampling [CCD99]. Our studies have shown that $df \cdot icf$ algorithms, of which *CORI* is an example, are robust when using sampled language models [CPFC00], so we restrict our choice of selection algorithms to one in this class.

10.3.3 Probing to Build Language Models

In our previous experiments, we had access to all of the documents in each collection $C_i \in \mathcal{C}$ from which to construct each language model $LM_i \in \mathcal{LM}$. That is not the case here. We have no access to summary statistical information from the collections, nor is it feasible to download all documents in each collection C to use in constructing a full language model. As a result, we employ a query-based sampling [CCD99] approach to acquire the sampled set of documents from each collection $C_i \in \mathcal{C}$ from which to build language model $LM_i \in \mathcal{LM}$. The query-based sampling approach uses a set of single-term probe queries to acquire the sampled set of documents.

The assumption behind this approach is that by randomly sampling a collection's pages, we will eventually build a language model that is a representative view of the vocabulary content contained in the collection. If we can effectively create language models of collections by sampling in this way, we can use a small subset of each collection's pages to build an accurate language model of the collection to be used by a collection selection approach.

We sample each of the subcollections at each search engine using a very general language model building tool, LAMB [OF00]. Given a collection that we wish to sample, we start with an initial probe query, submit the query to the search engine and retrieve an initial

set of results. In our sampling runs to date, up to ten results pages are viewed for each query. Dead links and obviously non-HTML documents are dropped from the list of pages seen for each query; usually fewer than all ten of the sampled pages retrieved by a probe query are actually useful. A link may be dead, too short or not HTML content. Long HTML documents are truncated to a fixed byte length to prevent very long documents (for example novels from Project Gutenberg[2] or inventory lists from online catalogs) from unduly influencing the language model. Each remaining page is fetched using the URL that the search engine returned, the HTML tags and any javascript or other scripting code are removed, and the page is filtered by a stoplist. The remaining content is stemmed, and the resulting terms and their statistics are then added to an accumulating language model of the contents of the search engine category being sampled. Our current collection selection approach requires only document frequency information.

Once we have processed the pages retrieved by the initial probe query, another query term is selected. All probe queries are single term queries, and currently the new probe query is taken from a growing list of the total unstemmed vocabulary seen so far. Each term in the vocabulary has an equal chance of being chosen as the next probe query, but the terms are chosen without replacement (i.e., no term is used more than once as probe query). The chosen term is used as the next query, and the process described above is repeated until a stopping criterion has been met. For the work reported here, we sample until 500 pages have been included in the language model. Work by Callan *et al.* [CCD99] and Monroe *et al.* [MMF00, Mon00] suggests that a large portion of the most frequently-occurring vocabulary can be identified using a sample of 500 documents.

Clearly it is improbable that a fixed sample size is appropriate for all collections, especially given the potential for wide variability of the underlying population sizes. A flexible stopping criterion is more appropriate. For example, we would like to sample until the language model is "good enough". Of course, defining "good enough" is a difficult task even when a full language model based on complete statistical information about a collection is

[2] `http://promo.net/pg/`

available for comparison. We only have the option to collect information about changes in the language model as we sample more documents from a collection. Initial investigations geared towards identifying a flexible stopping criterion are underway [MMF00].

The collection selection algorithms of interest to us all require information about the document frequency of each term in the language model. Recall that the document frequency df_{ij} of a term t_j in collection C_i is the number of documents in collection C_i containing at least one instance of term t_j. However, our sampled language model only contains true *df* information for the *sample*, not for the entire collection. We can only estimate the document frequency of a term from a sample of size s if we know the size of the underlying collection, $|C_i|$. If we know the collection size, we can estimate the document frequency as

$$\frac{df_{ij}}{s} \cdot |C_i|.$$

Unfortunately, we generally do not know $|C|$ for any collection C that we sample. The lack of knowledge of collection size, combined with the potential for stopping criteria that produce samples of different size, led us to investigate collection selection methods based exclusively on the *df* proportions $\left(\frac{df_{ij}}{s}\right)$. A preliminary study showing the effectiveness of this approach can be found in [SPM+00]. However, we do not consider this approach further in this chapter. For these experiments, we use fixed samples of 500 documents. We assume that language models constructed from these samples are representative of the underlying collections and use them directly.

10.3.4 Queries, Pages and Relevance Judgements

Having built language models for each of our 64 category-based collections, we needed queries, retrieved pages and relevance judgements in order to evaluate our fine-grained metasearching approach.

The TREC-8 conference included a question and answer track with 200 short fact-based topics (queries) phrased as simple questions. While we did not use the TREC documents or

relevance judgements, we did want to use a set of independently-generated queries for our experiments, so we began with this set of questions. Before retrieving any documents, we eliminated some questions for which relevance rules might be ambiguous. For example, one question asked *Who was the Democratic nominee in the American presidential election?*, but did not specify which election year. We also removed questions for which the answer might have changed recently, complicating the task of judging the documents. For example: *Who is the prime minister of Japan?* We also eliminated questions for which we felt there was a reasonable chance of finding the answer in an image. While we did download page images, our original plan had been to download only the text of the pages.

When we had finished eliminating questions, 58 remained. An additional two questions were later eliminated when we found that they were ambiguous, leaving 56 questions. For each question (TREC topic), we formulated a short query statement. For each query, we began by selecting 20 categories using the *CORI* collection selection approach over our previously obtained sampled language models. We issued the query to each of the four search engines at large, searching only the categorized pages, then to the 20 categories individually. In each case, we stored the top 20 retrieved documents, keeping track of which search engine or category returned each document. The documents were retrieved and stored due to the changing nature of the WWW and search engines. For the four search engines, we had the potential to retrieve 80 documents; for the 20 categories, 400, yielding a total of 480 potential documents to judge.

Obtaining relevance judgements is time-consuming and tedious, prompting us to retrieve documents from only the top 20 selected categories for each collection. While this choice does not affect our final retrieval results, it does have unfortunate implications for our ability to judge collection selection performance. Because we do not know how many relevant documents are found in the unselected collections, we cannot use the $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ measures defined in Chapter 5. Our ability to use the $\mathcal{R}_n$, $\widehat{\mathcal{R}}_n$ and $\mathcal{P}_n$ measures is also hampered because we do not know how many relevant documents we did *not* retrieve from each selected collection. We will discuss implications of this later.

Our next step was to completely anonymize the queries for the purpose of acquiring relevance judgements. We renamed each downloaded page then sorted pages by the original URL—it was impossible to tell from which engine or category a page had been retrieved. It was also impossible to tell which general approach had yielded the page. In the case of duplicate pages (exact same URL), we only judged one copy.

We then judged the pages based on whether or not they answered the *whole* question, not just some portion of the question [3]. We were also strict in our judgements. For example, for question 77 shown in Table 10.2, it was not sufficient for a page to state that Marlon Brando played Don Vito Corleone in "The Godfather". It was also necessary for it to be clear that Vito Corleone was the title character. Despite the fact that these queries all have simple answers, we assume that each page that answers the question is relevant. We were able to expedite the acquisition of relevance judgements by requiring that terms from the answer appear on a page in order to be judged. For example, for question 77, all pages that did not include the term "brando" were automatically judged non-relevant. Only pages that included the term "brando" were examined to see if they met our stricter criteria for relevance.

At present, we have relevance judgements for all 56 queries. Table 10.2 contains a summary of information about 10 representative queries from the set of queries that we used. The ID is the original TREC topic number; we have also included the text of the question asked. We show the total and unique number of pages downloaded by both approaches, plus the number of relevant pages found by each approach. Note that there may be overlap in the pages and relevant pages found by the two approaches. There are three interesting things to note about Table 10.2. First, we have the potential to retrieve up to 80 pages for each query for the search engines. However, in Table 10.2, we find that we retrieve all 80 pages for only one of the 10 queries shown; in many cases we only retrieve 60 pages. For these 10 queries, this is due to a difficulty with the Yahoo search engine—for some queries Yahoo may retrieve no pages, for others it may not retrieve all 20 allowed. We

[3] Gordon and Pathak [GP99] note that when fact-based questions are used relevance judgements by experimenters are acceptable.

Query (TREC Q&A Topic)		URLs engines		URLs 20 categories		Relevant pages	
ID	Query text	tot.	uniq.	tot.	uniq.	eng.	cat.
33	What is the largest city in Germany?	60	60	360	306	4	6
77	Who played the part of the Godfather in the movie, "The Godfather"?	60	60	318	280	2	2
80	What is the name of the promising anticancer compound derived from the pacific yew tree?	61	61	274	243	19	40
82	How many consecutive baseball games did Lou Gehrig play?	60	57	350	295	26	66
87	Who followed Willy Brandt as chancellor of the Federal Republic of Germany?	60	60	254	225	17	15
125	In what city is the US Declaration of Independence located?	80	74	350	230	6	4
173	How many moons does Jupiter have?	75	69	291	236	19	26
183	What was the name of the computer in "2001: A Space Odyssey"?	72	68	348	287	20	37
193	Who was the 16th President of the United States?	60	59	260	204	30	29
199	How tall is the Matterhorn?	60	60	295	259	5	6

Table 10.2: Characteristics of 10 of the 56 queries, showing total and unique pages retrieved using both approaches, plus the number of relevant pages.

will revisit this difficulty with Yahoo shortly. Another interesting feature is the source of relevant documents. Given that the four search engines retrieve up to 80 pages while the 20 categories can retrieve up to 400 pages, we would expect to find more relevant documents retrieved by the categories. However, for many of the queries shown here, the numbers of relevant documents per approach is similar. Finally, we find that some queries have very few relevant documents; the implications of this will be discussed later.

10.4 Experimental Setup

The general approach of using search engine categories as a subdivision of the total content, then applying a fine-grained metasearching approach to searching is applicable to a broad array of applications. However, the specific test environment that we described above, including the stored pages, the queries and the relevance judgements, was created for a

specific set of metasearching experiments, and as a result is slightly more limited. However, it does allow us to perform a number of interesting comparisons. For our set of queries, we can compare the performance of the individual search engines, the performance of traditional metasearching and the performance of fine-grained metasearching. We also have the capability to study fine-grained metasearching when smaller numbers of collections are selected (i.e. we can use fewer than the twenty collections originally selected). We can examine different results-merging approaches and have the option of examining the results when different subsets of our queries are employed.

In this section, we provide some further details of the implementation of our experimental environment, including a recap of our collection selection approach, how we issued queries to the individual collections and our results merging approach.

10.4.1 Collection Selection

As we mentioned earlier, our language models are built from 500-document samples of each collection. These 500-document samples are treated as representatives of the collections—selecting the representative implies that the query will be sent to the original collection. On a conceptual level, there is some question as to whether this is a viable approach. In our current environment, we do not have access to the full language models, so we cannot determine how representative the sampled language model is. However, earlier work has shown that *CORI* collection selection holds up well for sample-based language models [CPFC00, CBH00], so we begin with this approach.

Table 10.3 demonstrates the behavior of the collection selection approach in our experimental environment. The score shown in the first column of the tables is a normalized collection ranking. The top 10 highest-ranked categories are listed, along with the number of relevant documents returned when the query is issued to the category. There are a number of things to note here. At this point, we have not accounted for duplicates—the same pages may have been retrieved via multiple categories. At this point, each category will get credit for retrieving a relevant page, even if others retrieve it as well.

TREC Q&A topic 80			
Description: What is the name of the promising anticancer compound derived from the pacific yew tree? **Actual query:** anticancer pacific yew tree			
Score	Search Engine	Category	Rel. Docs
1.00	Yahoo	business & economy	
0.81	Yahoo	social science	
0.70	Altavista	home	
0.64	Altavista	health	2
0.63	Northern Light	health & medicine	20
0.61	Infoseek	health	1
0.59	Yahoo	government	
0.59	Infoseek	travel	
0.52	Altavista	regional	
0.50	Altavista	science	

TREC Q&A topic 154			
Description: How many Grand Slam titles did Bjorn Borg win? **Actual query:** grand slam titles bjorn borg			
Score	Search Engine	Category	Rel. Docs
1.00	Yahoo	entertainment	
0.97	Infoseek	sports	2
0.66	Altavista	sports	1
0.61	Northern Light	technology	1
0.58	Altavista	arts	7
0.48	Altavista	games	
0.47	Altavista	society	2
0.44	Infoseek	arts & humanities	
0.43	Yahoo	arts	
0.42	Altavista	autos	

TREC Q&A topic 173			
Description: How many moons does Jupiter have? **Actual query:** moons jupiter			
Score	Search Engine	Category	Rel. Docs
1.00	Altavista	news	13
0.49	Northern Light	science & mathematics	5
0.43	Altavista	science	14
0.31	Infoseek	science	5
0.29	Northern Light	technology	2
0.25	Infoseek	entertainment	
0.24	Altavista	arts	
0.22	Yahoo	entertainment	
0.20	Infoseek	education	5
0.20	Infoseek	computing	

Table 10.3: Three queries demonstrating collection selection behavior.

We first consider the results on a qualitative level. Note that 3 of the 4 "health" categories are suggested in the first 6 ranks of query 80. The "science" categories are similarly represented in query 173. Query 154 shows more variability in the suggested categories, choosing only two of the "sports" categories and a slightly wider variety of other categories. Overall, these categories seem eminently reasonable. When we consider the number of relevant pages available from each category, we note that there is variability in the number of relevant pages available. On this point, we are at the mercy of both the selection step and the ability of the search engine to locate and rank highly relevant documents within the category. We find that there are different numbers of relevant documents available to each query but until we actually retrieve and merge the pages, the impact of this is unknown.

10.4.2 Issuing the Queries to the Collections

Each of the search engines that we employed was chosen because it provides the capability to restrict a search to a single category and to the entire category hierarchy. While this is generally accomplished by selecting check-boxes or pull-down menus on the search engine's search page, the selections are embedded in the search URL, allowing us to achieve the same results in a more automated fashion.

We used the default "simple search" protocol of each of the search engines to issue each query to each selected category. This seemed a reasonable choice, given that this is the default behavior of the search engines and this is the choice of most users [JSBS98, SHMM99].

Of course, for our experiments, we do not re-fetch the pages. We simply use the stored versions of the pages that were used to acquire relevance judgements.

10.4.3 Results Merging

As we discussed in Chapter 2, there are a number of different approaches to results merging that can be employed. For these experiments, we did not always have access to the document scores used by the search engines to rank the documents. At the same time, we were

aware that the results from some collections may be more relevant-rich than the results from others. We chose to re-rank the retrieved documents using Excite for Web Servers (EWS)[4].

EWS has a number of appealing properties. Because it is intended for searching WWW sites, it can automatically handle HTML documents. Because Excite was not one of the search engines that we used for our test environment, EWS can serve as an independent merge step. For example, if we had used Altavista to merge, there may be the potential that documents originally retrieved from an Altavista category would be preferred in some way during the merge.

EWS was used to index and rank all of the unique pages retrieved from the selected collections. For our experiments, only pages with identical URLs were counted as duplicates. The general problem of duplicate detection is beyond the scope of this work, but has been considered by other researchers. For example, Yan and García-Molina [YGM95] consider duplicate detection in the context of routing incoming documents to standing queries, Gauch *et al.* [GWG96] consider duplicate removal for metasearching and Bharat *et al.* [BBDH00] examine the problem of identifying mirrored WWW sites (replicated collections in our parlance). For our experiments, given the original query, the EWS results were the merged results for the multi-collection search.

Unfortunately, because the algorithm is not described and the source code is not distributed, EWS is a black box results merging step. While the document rankings that are returned appear to be quite good overall, we cannot comment upon features of the ranking algorithm that we use for merging.

10.5 Preliminary Results

In this section, we present preliminary results from our metasearching and fine-grained metasearching experiments. Overall, our results are inconclusive as to retrieval effective-

[4] `http://www.excite.com/navigate`

ness, due to some issues with our underlying data set[5], nonetheless the results are instructive. We will comment upon this as we discuss the results and present suggestions for improvement. However, the main thing to take away from this chapter is that multi-collection information retrieval approaches can be applied successfully in an operational environment.

All results are presented in two versions. For each table, subtable (a) represents results for all 56 queries. However, query-by-query analysis of the results revealed that there are many queries for which there are very few relevant documents. All approaches appear to be similarly affected by these queries and the overall affect is to depress the average precision values. We elided all queries for which there are less than 20 unique relevant documents across all search engine categories and category trees (i.e. less than 20 unique relevant documents out of the potential 480 judged pages). This alternate view of the data is shown in sub-table (b) of each table.

10.5.1 Effectiveness of Traditional Metasearching

Our first step was to verify the usefulness of traditional metasearching for our test environment. Given our relevance judgements, we examined precision at 5, 10, 15 and 20 documents for each of the four engines individually and for the merged traditional metasearch results. Tables 10.4(a) and (b) show the results. The most striking feature of both sub-tables is the apparently very poor performance of Infoseek and Yahoo (a difficulty with Yahoo was first noted in conjunction with Table 10.2). Both Infoseek and Yahoo appear to have fallen prey to our multi-term queries. While we cannot tell exactly how Infoseek and Yahoo process queries, it appears that pages are required to contain all query terms. In addition, Yahoo appears to index very little information about pages in its category structure. As a result, for many queries Infoseek and Yahoo return very few documents (the number of queries for which an engine returned at least one page is noted in the table heading). This factor is

[5] As we will note in a moment, there were issues with Infoseek and Yahoo regarding our multi-term queries. In addition, we use at most 56 queries for these experiments. Buckley and Voorhees [BV00] recommended that at least 100 queries be used when evaluating using precision at 20 documents.

Precision at Rank	Metasearch (54 queries)	Altavista (54 queries)	Infoseek (51 queries)	Northern Light (54 queries)	Yahoo (29 queries)
5 docs	0.404	0.230	0.098	0.422	0.097
10 docs	0.432	0.194	0.071	0.422	0.066
15 docs	0.430	0.175	0.052	0.396	0.060
20 docs	0.414	0.151	0.045	0.399	0.057

(a)

Precision at Rank	Metasearch (26 queries)	Altavista (26 queries)	Infoseek (26 queries)	Northern Light (26 queries)	Yahoo (17 queries)
5 docs	0.592	0.377	0.162	0.608	0.165
10 docs	0.669	0.319	0.123	0.619	0.119
15 docs	0.654	0.290	0.092	0.600	0.098
20 docs	0.648	0.252	0.079	0.619	0.094

(b)

Table 10.4: Comparison of engine and metasearch performance for (a) all 56 queries and (b) the 26 queries with at least 20 relevant documents.

unfortunate on a number of fronts. First, the performance results for these search engines is depressed. Second, this in turn affects the metasearch results differently for each query. Third, when categories from these search engines are selected for fine-grained metasearching, no documents may be returned. This can affect the fine-grained metasearching results for a given query in unpredictable ways, depending on the number of categories from these search engines that are selected.

This issue aside, the main result to take away is that, despite problems at the constituent search engines, metasearching generally outperforms even the two best constituents. Also note that the same general performance trends are seen for all 56 queries and the 26 queries with at least 20 relevant documents. The primary difference between the two sets of queries is in the numeric values of the scores.

10.5.2 Metasearching vs. Fine-Grained Metasearching

Our next step was to compare the more highly focused fine-grained metasearching to the traditional metasearching approach. Tables 10.5(a) and (b) show the results.

Our initial approach was to issue the queries to all 20 collections selected using *CORI*,

Precision at Rank	Metasearch	20 cat (CORI)	10 cat (CORI)	4 cat (CORI)	4 cat (RBR) (local)	4 cat (RBR) (CWI)
5 docs	0.404	0.426	0.382	0.293	0.441	0.467
10 docs	0.432	0.432	0.352	0.276	0.419	0.459
15 docs	0.430	0.417	0.341	0.249	0.419	0.441
20 docs	0.414	0.385	0.323	0.232	0.411	0.421

(a)

Precision at Rank	Metasearch	20 cat (CORI)	10 cat (CORI)	4 cat (CORI)	4 cat (RBR) (local)	4 cat (RBR) (CWI)
5 docs	0.592	0.639	0.554	0.431	0.615	0.654
10 docs	0.669	0.662	0.570	0.442	0.612	0.654
15 docs	0.654	0.649	0.562	0.410	0.623	0.651
20 docs	0.648	0.604	0.542	0.389	0.621	0.633

(b)

Table 10.5: Comparison metasearch and fine-grained metasearch performance for (a) all 56 queries and (b) the 26 queries with at least 20 relevant documents.

merge the results, then compare to the traditional metasearch results. The average numeric performance was sometimes better and sometimes worse than traditional metasearch.

Unfortunately, a direct comparison of traditional metasearching using four search engines and category metasearching using 20 selected categories is less than satisfying. While the fine-grained category metasearching searches a smaller set of documents (each category is a subset of the documents in the whole category tree), the effort expended in terms of network traffic and searches executed is not comparable, making it difficult to comment upon any efficiency gain fine-grained metasearching might eventually provide. In addition, 20 collections selected represents approximately one third of the available collections. For scalability reasons, we would prefer to be able to search a smaller subset of the collections.

As a result, we also considered performance when both ten and four categories are selected. We chose to examine selecting four categories because that allows network traffic and number of queries issued to be held constant. While the full category tree searches

have a much larger set of documents to work with during initial retrieval, the hope is that the set of documents available to the selected categories was more relevant-rich.

We note a steady decline in retrieval effectiveness performance when ten, then four categories are selected. The results when only four categories are selected are much worse than the traditional metasearching results. We considered the possibility that our selection step performed poorly in selecting just the top four collections. To test for this, we performed selection using the RBR[6] selection baseline. For this approach, we selected the four categories with the most relevant pages (i.e. used an oracle for category selection). Note that we did not account for duplicates—there may exist a case for which the second selected category was not in fact the best choice because it contained pages already found by the first selected category.

These results are reported in Tables 10.5 (a) and (b), column "4 cat (RBR) (local)". There was substantial improvement over the original "4 cat (CORI)" results, and performance approaches that seen for traditional metasearch. This implies that poor selection was a major cause of the poor performance of "4 cat (CORI)".

An additional observation prompted another experiment. We noticed a disturbing trend when examining the raw results of "4 cat (RBR)". The ranks of some relevant documents had in fact degraded from the ranks seen when twenty categories were selected (i.e. non-relevant documents were now being ranked ahead of relevant ones). This effect would not be seen if we simply removed documents from un-selected collections from the ranking. This effect is due to a change in merging. While we do not know the exact algorithm of EWS, we do know that when only four categories are selected, the merge algorithm has a much smaller pool of pages from which to draw statistical information. Given a homogeneous set of pages (like we would get if, as we intended, we selected pages from the same general topic or category area), the collection statistics might not be accurate [VF95b]. While our recent research, reported in Chapter 9, has suggested that CWI might not be necessary given good selection, given our current very small sample of pages, it might be necessary. To

[6]Note that for these experiments, RBR means the best four collections of the original 20 selected.

test this possibility, we selected the best four categories but we merged using the statistical information for the pages from all 20 categories. These results are shown in Table 10.5 as column "4 cat (RBR) (CWI)". The use of CWI did lead to a small increase in average performance.

10.6 Discussion and Future Work

In this chapter, we have shown that the multi-collection retrieval techniques covered in Chapter 9 are applicable to an operational WWW-based environment. We have described a novel application of data decomposition to web searching and metasearching and have demonstrated the feasibility of acquiring the necessary summary statistical information by query-based sampling. The technique is readily adaptable to the search engine technology deployed today.

Our evaluation of the approach was hampered by data issues which we are currently working to resolve. Throughout the chapter, we noted specific experimental points that need to be addressed. The primary point to be addressed is the way in which queries are issued to the underlying search engines. For some search engines, the default operation is not well suited to multi-term queries such as those that we use.

11

Conclusions

This dissertation reports the results of a large empirical study that has produced a number of interesting findings that can be usefully applied to the engineering of single- and multi-collection information retrieval systems. We have presented detailed results on a number of fronts; unfortunately, this may have obscured the broader picture. The overall problem that we have addressed has been information retrieval in a multi-collection environment. This problem presents challenges beyond those of traditional single-collection information retrieval because, in addition to being concerned about effective retrieval at each of the multiple collections, we must also be concerned with effectively selecting the collections to which queries should be sent and then merging the individual results lists. Our experiments have focused primarily upon comparing collection selection techniques and determining their impact upon multi-collection retrieval.

We begin by briefly summarizing our findings, then outline the major contributions of this work. We conclude with remaining work to be considered and final comments.

11.1 Summary of Findings

Here we summarize the key points of our findings, focusing on collection selection and its impact on multi-collection information retrieval. We also consider the applicability of these approaches to an operational WWW-based environment.

11.1.1 Collection Selection

A large portion of this dissertation (Chapters 6–8) has been concerned with the collection selection step of multi-collection information retrieval.

We conducted comparisons of three published approaches, *CORI* [CLC95], *gGlOSS* [GGM95] and *CVV* [YL97] using six test environments. Among those published approaches, *CORI* consistently performed the best for our test environments.

On a more detailed front, we found that all approaches that we considered exhibit some degree of positive correlation with SBR (the number of documents per collection). A positive correlation with SBR is not necessarily a detriment; the number of *relevant* documents per collection also tends to be correlated with SBR. However, *gGlOSS* exhibits a very strong positive correlation with SBR, which led to performance downfalls for some of our test environments. With a few exceptions, we also found that, on average, *CORI*, *gGlOSS* and *CVV* all tend to perform better than the expected performance of selecting collections at random. While not surprising, this finding is reassuring, suggesting that the effort being expended on collection selection is beneficial.

CORI is an instance of a $df \cdot icf$ approach. We also investigated variations of the general $df \cdot icf$ approach. We found that while the df component normalization of *CORI* is the key to its slight performance advantage, overall, the basic $df \cdot icf$ appears to be very sound. This is very interesting because $df \cdot icf$ approaches tend to require less statistical information from collections than *gGlOSS* and to have simpler computations than *gGlOSS* and *CVV*. The very simple document frequency information required by $df \cdot icf$ approaches tends to make them more easily applicable to collections using different information retrieval systems, so long as resolution of any vocabulary differences can be performed. Finally, *CORI* has been shown to hold up well using incomplete language models. This suggests that *CORI* is particularly useful for environments where the full cooperation of the underlying collections cannot be gained.

We hope that with these experiments we have helped to resolve, or at least clarify the ongoing debate about whether detailed statistical information about collections is necessary

or useful. There is still substantial room for collection selection improvement, so an as-yet-unforeseen use of detailed information may eventually prove to be useful. However, our experiments showed simpler approaches to be more effective.

11.1.2 Multi-collection Information Retrieval

In Chapter 9 we considered the question of the impact of *CORI* and *RBR* collection selection on multi-collection information retrieval. Once again, we performed our experiments using six test environments utilizing three different document testbeds. We gathered a varied set of findings from these experiments.

First, we found that when very good selection is employed, multi-collection retrieval can outperform retrieval in an equivalent single-collection environment. This verified for a broad array of test environments an effect mentioned by Xu and Croft [XC99] and seen by Voorhees *et al.* [VGJL95] and Craswell *et al.* [CBH00]. This result suggests that single-collection performance is not necessarily the gold standard that we should be aiming for. It is possible for multi-collection searches to achieve better retrieval performance.

Second, we found that it is possible to achieve good document retrieval performance by selecting a small number of heterogeneous collections. This bodes well for the scalability of multi-collection information retrieval systems. At the present time, the use of a multi-collection information retrieval system may be an engineering choice or may be imposed by external circumstances. This finding, and the previous one, suggest that environments such as these need not be a performance liability and in fact may become a performance advantage.

Third, we have observed that we can *conceptually* decompose a single collection into subcollections and by introducing a selection step it is possible to achieve better performance than by searching (ranking) the entire collection. This observation is hinged upon improvements in collection selection; however, we find that selection plus ranking has the potential to improve the effectiveness of ranking alone. Moreover, we can use a simple raw score merge in this case and nothing more elaborate.

Fourth, we have found that the use of collection-wide information is a complex issue. While this is not a surprise, it is important to re-iterate that this is something to consider carefully during system design. Given the scenario in the third observation above (labeled `multi-CWI` in Chapter 9), the straightforward approach of using already-available CWI plus a raw score merge produced good results. However, given a pre-existing multi-collection environment, using local information works well if collection selection is employed and the document scores are suitably normalized before merging is performed. Yet, we found in Chapter 10 that if we are dealing with small collections or small samples of collections, CWI can be useful.

With these experiments, we sought to determine the degree to which collection selection would have an impact on multi-collection retrieval performance. We believe that we met this goal and have provided concrete conclusions that can usefully guide the engineering of large-scale multi-collection information retrieval systems.

11.1.3 Metasearching

In Chapter 10, we addressed the question of whether laboratory-based multi-collection information retrieval techniques could be applied to the World Wide Web. We described a novel application of data decomposition to web searching and metasearching and demonstrated the feasibility of acquiring the necessary summary statistical information by query-based sampling. We encountered some data-based issues in our experiments; however, these experiments represent a useful feasibility study and showed that the multi-collection retrieval techniques covered in Chapter 9 are applicable to an operational WWW-based environment.

11.2 Contributions

This work makes two primary contributions, listed here in experimental order. First, our comparison of collection selection approaches showed that effective collection selection can

be performed using very limited collection information (document frequency). We hope that this clarifies the debate about whether more detailed statistical information is required. Second, our investigation of the impact of collection selection on multi-collection retrieval showed the impact of selecting different numbers of collections and showed on a broader scale than previously seen the potential to exceed equivalent single-collection performance.

The environments required to conduct these experiments, and the process of describing the experiments have led to a number of related contributions. First, we conducted experiments using six test environments utilizing three testbeds. We constructed two of those testbeds and have made the details of construction available so that they can be used by other researchers. We have also characterized features of all three testbeds and used those features to shed light on collection selection performance. In the process of describing our experiments, we have introduced notation for describing multi-collection experiments such as those conducted here, and for describing multi-collection experimental environments. Finally, the evaluation methodology for collection selection experiments is not as standardized as the methodology for document retrieval. We collected a variety of evaluation measures for use in these experiments, showed relationships among them and discussed expected random performance under the measures.

The collection selection experiments of the first major contribution above also yielded additional contributions. We performed a direct comparison of three competing collection selection approaches and analyzed reasons for observed performance differences, including a consideration of features of the experimental test environments. We abstracted the approach that performed best in our comparison of collection selection approaches and examined the impact of its constituent components, isolating the difference that is key to its success.

In the process of conducting the experiments for the second major contribution, we found that for all six test environments, good collection selection of a subset of collections can yield data item retrieval results superior to those when all data items are located in an equivalent single collection. We presented an interpretation of multi-collection environ-

ments that allows these techniques to be layered on top of existing single collection systems. We also revisited the issue of collection-wide information in multi-collection retrieval.

Finally, we have studied the multi-collection retrieval approaches discussed here in a WWW metasearch environment. We reported preliminary effectiveness results, plus an experience report of issues faced when applying these techniques to a heterogeneous operational environment.

11.3 Future Work and Summary

Two avenues of remaining work are apparent. For the WWW-based experiments reported in Chapter 10, we pointed out that while our results are instructive and serve as a valuable experience report, they are inconclusive regarding retrieval effectiveness due to data issues. Throughout that chapter, we pointed out issues that need to be addressed. We plan to address those issues and to conduct further experiments to gauge the effectiveness of these techniques in a WWW environment. In addition, throughout this disseration we have noted instances where query-specific issues have impacted average performance. For example, some queries have very few relevant documents or have relevant documents in few collections. A detailed examination of results reveals that collection selection and document retrieval performance can vary significantly on a query-by-query basis. We believe that an in-depth study of query-specific performance would be instructive.

To summarize, we set out to increase understanding of collection selection and its impact on multi-collection information retrieval. We have studied differences in collection selection algorithms and shown how improvements in these algorithms have the potential to impact multi-collection information retrieval. With the experiments reported here and the analysis of collection-specific issues, we have improved understanding of the problem, clarified a number of issues and shown the importance of continued research into improving collection selection. We have given specific guidance that can be applied to the engineering of large-scale multi-collection and single-collection information retrieval systems.

A

Making Fair Comparisons

When performing our early comparisons of *gGlOSS* and *CORI* discussed in Chapter 6, we wanted to be certain that any differences in performance were due solely to differences in the algorithms. One concern was that subtle underlying differences in parsing, tokenizing, stopping, stemming, or indexing could be affecting the two approaches differently.

These subtle differences existed because we were

- attempting to replicate the published version of *gGlOSS* [GGM95], and
- using an official distribution of *CORI*.

Figure A.1 illustrates the differences found and our steps to control for those differences. Published *gGlOSS* results were obtained using SMART indexes of the underlying collections, while the authoritative version of *CORI* uses statistical information gleaned from Inquery indexes. These implementations are denoted by the heavy arrows in Figure A.1. SMART and Inquery differ slightly in the tokenizer, stoplist and stemmer used[1]. There was also the potential for slight differences in parsing the original TREC SGML document format provided by NIST. Finally, Inquery and SMART also differed in their overall indexing approach. The different indexing approach is a factor only for *gGlOSS*, which uses term weight information and document-query similarities. Overall, the potential for variability

[1] A tokenizer breaks a text document into words or tokens for indexing, a stoplist removes extremely common words such as articles, and a stemmer compresses the indexing vocabulary by removing word suffixes.

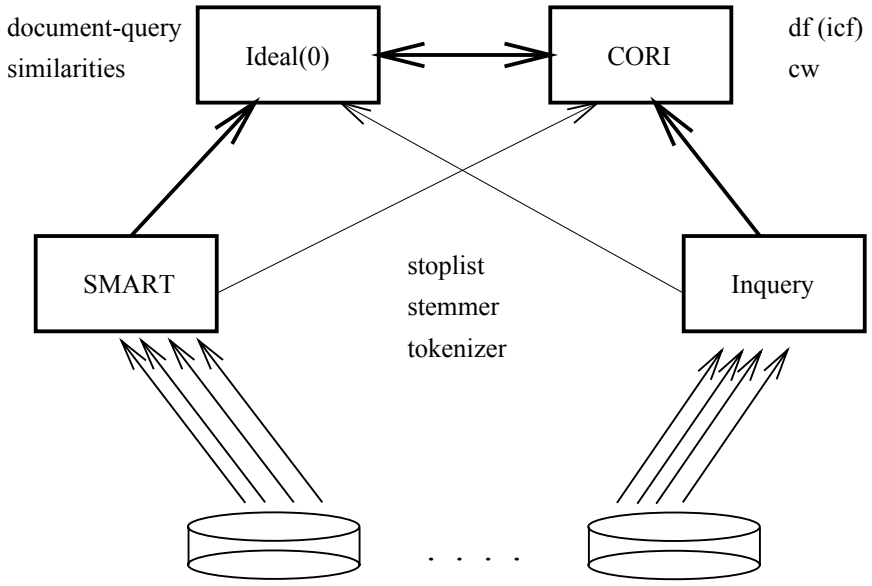


Figure A.1: Ensuring a fair comparison.

was greater for *gGlOSS* than for *CORI*. *CORI* relies primarily on document frequency information and as a result is affected by vocabulary differences caused by differences in the parser, tokenizer, stoplist and stemmer. *gGlOSS* is sensitive to these differences plus the differences in indexing.

Please note that these comparisons were performed during early stages of the experiments. As a result, the "very long" queries were used. The experiments reported here use the same test environment as was used in Chapter 6, $(SYM\text{-}236, \mathcal{Q}_{vl}, \mathcal{J}_{TREC4})$. The graphs shown here are directly comparable to those shown in Chapter 6 but due to query differences, vary from those shown in Chapters 7 and 8.

A.1 Two Versions of *CORI*

For the purpose of the algorithm comparisons reported in French *et al.* [FPC+99b] and summarized in Chapter 6, it was necessary to guarantee that the same indexing vocabulary was used by *gGlOSS* and *CORI*. To control for the aforementioned differences in parsing,

tokenizing, stopping and stemming, we synthesized pseudo-documents for each collection. We used the stemmed and stopped vocabulary from the SMART indexes used for earlier *gGlOSS* experiments [FPV+98], emitted each term tf times, and padded the document back to it original length with a fake word. Inquery was then run with stemming off and a single word stoplist (the fake word). It was also necessary to handle special characters differently. For example, the SMART default would be to treat smith@virginia.edu as a single indexable unit (token), while the Inquery default would be to create three tokens: smith, virginia and edu. To ensure that the same vocabulary was used, it was necessary to override the Inquery default. The end result was that Inquery indexed exactly the same vocabulary that SMART did in the earlier experiments.

CORI was constrained to use the same vocabulary as *gGlOSS* to enable a straightforward comparison between the two approaches. However, the potential still existed that by doing this we had hampered the overall performance. For example, the Inquery tokenizing, parsing, etc. might have lead to better *CORI* or *gGlOSS* performance. As an initial check, we also ranked the collections using *CORI* with Inquery defaults then evaluated the ranking. We compared the average *CORI* performance results using the two underlying indexing approaches. These results are shown in Figure A.2. The plot of *CORI* using Inquery indexing is labeled *CORI-UMass* and the vocabulary-controlled version is labelled *CORI-UMass-V* The average performance results using Inquery default indexing were nearly identical to the *CORI* results using the controlled vocabulary.

There was some some significant difference between the two *CORI* results. In Figure A.2 significance is shown at the very bottom of the graph. Points for which the Inquery version of *CORI* significantly outperforms the SMART version are marked. Both versions of *CORI* outperformed *Ideal(0)*; the impact of the *CORI* differences was negligible[2].

[2]The only difference in significance was when $n = 141$ collections had been selected. The Inquery version of *CORI* outperformed *Ideal(0)* significantly while the vocabulary-controlled version did not.

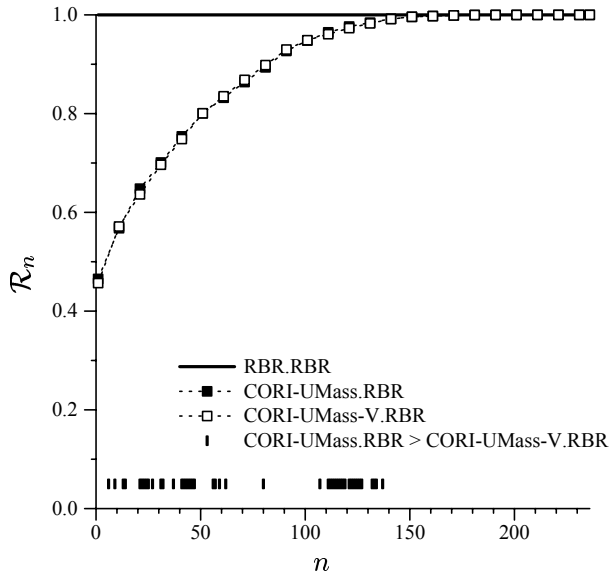


Figure A.2: *CORI* using Inquery stats vs. *CORI* using SMART stats

A.2 Two Versions of *Ideal(0)*

The second half of our vocabulary resolution experiments were prompted when examining the comparison results covered in Chapter 6.

The immediate question was why *CORI* approximated the RBR baseline more closely than *gGlOSS Ideal(0)*. Our initial observation was that the performance differences could be due to indexing differences. Our initial *gGlOSS* experiments were intended to reproduce the experimental setup of Gravano and García-Molina [GGM95]. As a result, we used the exact underlying retrieval engine and parameter settings reported—SMART using *ntc* weights for document terms and *nnn* weights for query terms.

It has been shown that using different weights and normalization approaches in SMART improves document retrieval performance for the large TREC collections [BAS93, BSA92, SBM96], so we considered that Inquery, shown to perform well at TREC, might provide better-tuned indexing information. Therefore, we used the underlying Inquery collection indexes—used to generate the *CORI* inference net—to generate an additional *Ideal(0)*

baseline based upon the Inquery indexing information. We will refer to the new *Ideal(0)* baseline as *Ideal(0)-inquery*. If *Ideal(0)-inquery* approximated RBR more accurately than *Ideal(0)*, it would imply that the indexing weights used by Inquery provided a better input to the *gGlOSS* algorithm. In this case, it would be more representative to compare the *CORI* performance to that of *Ideal(0)-inquery* than to that of *Ideal(0)* (labeled *Ideal(0)-smart* in Figure A.3 for clarity). The comparisons of *Ideal(0)* and *Ideal(0)-inquery* to baseline RBR for evaluation measure $\mathcal{R}_n$ are shown in Figure A.3. Points for which the SMART version of *Ideal(0)* significantly outperforms the Inquery version are marked on Figure A.3.

Note that, in fact, the performance of *Ideal(0)* declined when the Inquery weighting information was used. While there may exist weighting schemes that yield better *Ideal(0)* performance, it was not our goal to uncover them. We merely wished to verify that the already-available Inquery indexes did not yield superior *Ideal(0)* performance.

Closer examination revealed that document-query similarity scores produces by Inquery tend to fall within a more compressed range than the scores produced by SMART using the *ntc.nnn* parameters. An examination of the *Ideal(0)* computation reveals that these similarity scores are summed during the computation. The compressed range of values for Inquery scores exacerbates the tendency of *Ideal(0)* to prefer large collections (see the discussion of the correlation between *Ideal(0)* and SBR in Chapter 7). We used the SMART versions of both *Ideal(0)* and *CORI* for the comparisons reported in French *et al.* [FPC+99b].

At the time of these comparisons, we were not certain what aspect of the *CORI* database selection algorithm was responsible for its performance difference. We conjectured that to *gGlOSS* a database with many documents of low similarity may appear more useful than a database with a few documents of high similarity [FPV+98]. We also considered the possibility that *CORI* was utilizing a better length normalization strategy that allowed it to avoid this difficulty. These issues are discussed in more detail in Chapter 8.

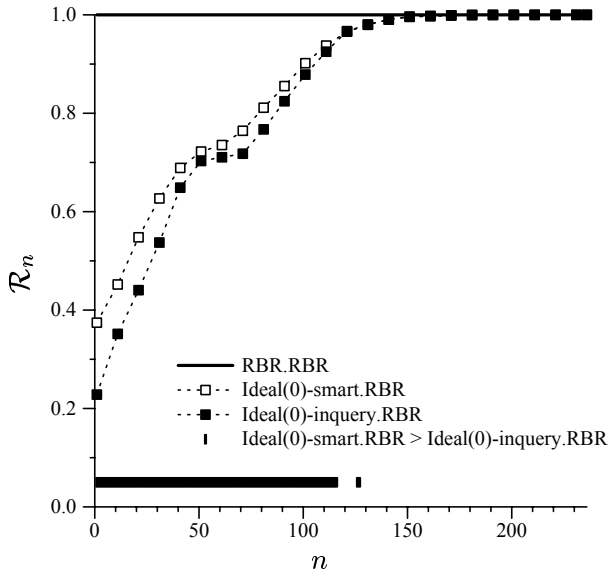


Figure A.3: *Ideal(0)* using Inquery stats vs. *Ideal(0)* using SMART stats.

A.3 UVA Implementation of *CORI* vs. Official (UMass) *CORI*

Because the distributed version of *CORI* is intended for use with Inquery, it is not straightforward to build a *CORI* selection index over statistical information produced by an alternate indexer. The approach that we described above of fabricating pseudo-documents to be indexed by Inquery was cumbersome. As a result, we implemented a version of the published *CORI* algorithm that used directly the *df* information provided by an alternate indexer.

The UVA implementation of *CORI* was used for the collection selection comparisons reported in Chapters 6, 7 and 8 so that the algorithms could be compared using identical vocabularies. However, having verified that *CORI* is a reasonable choice for selection, when we moved to the document retrieval experiments in Chapter 9 we used the authoritative University of Massachusetts (UMass) version of *CORI*.

Here, we summarize experiments performed to verify the effectiveness of the UVA im-

plementation of *CORI*. First, we compare three different versions of *CORI*: the two versions presented in Figure A.2 and the UVA implementation. This comparison is performed using the "very long" query formulation and the *SYM-236* testbed. We later present the authoritative UMass and UVA versions of *CORI* results for short and long queries using the *SYM-236*, *UDC-236* and *UBC-100* testbeds. We use the UVA version of *CORI* when performing the comparisons presented in Chapters 7 and 8 to enable a more straightforward comparison. However, the authoritative UMass version of *CORI* often performs slightly better, so we present those performance curves here for reference.

A.3.1 The UVA Implementation of *CORI*

The first two versions of *CORI* are shown in Figure A.2; the third is our implementation of the published algorithm. To quickly summarize, the *CORI* belief values are computed as follows. A more detailed discussion of *CORI* can be found in Chapter 7.

The belief $p(r_k|C_i)$ in collection C_i due to observing query term r_k is determined by:

$$\begin{aligned} T &= \frac{df}{df + 50 + 150 \cdot cw/\overline{cw}} \\ I &= \frac{\log\left(\frac{N+0.5}{cf}\right)}{\log\left(N + 1.0\right)} \\ p(r_k|C_i) &= 0.4 + 0.6 \cdot T \cdot I \end{aligned} \tag{A.1}$$

where:

- df is the number of documents in C_i containing r_k,
- cf is the number of collections containing r_k,
- N is the number of collections being ranked,
- cw is the number of words in C_i, and
- $\overline{cw}$ is the mean cw of the collections being ranked.

The df values are available from the F matrix that is maintained for use by other selection algorithms. The icf values can be computed using the F matrix. That leaves

only the *cw* values, the number of words in a collection prior to stopping, stemming, etc. Given access to the collections, determining that value is straightforward.

Given this required information, we computed a belief value for each collection using the published algorithm. The belief in a collection is the average of the $p(r_k|C_i)$ values for each query term. Ranking collections using these belief values produces our new *CORI-UVA* rankings.

The three versions of *CORI* are shown in Figure A.4. The average performance curves appear nearly identical; however, there is some significant difference between the implementations (paired Wilcoxon, $p = 0.05$). Pairwise significance is shown in the lower portion of Figure A.4. For each comparison, the baseline is no significant difference (NSD); the plot shifts to one of the approaches when that approach is significantly better than the approach with which it is paired.

The *CORI-UVA* rankings differ from the *CORI-UMass* ranking in vocabulary. There is also the possibility of subtle differences in the calculation of *cw* between *CORI-UVA* and the other two rankings.

A.3.2 Full UVA and Official *CORI* results

Figure A.5 shows both the UMass and UVA *CORI* results for the *SYM-236*, *UDC-236* and *UBC-100* testbeds using short and long queries. The UVA results are used for the selection algorithm comparisons presented in Chapters 7 and 8. The official results are used for the retrieval experiments in Chapter 9.

Overall, the average performance of the two implementations is very similar. The greatest difference can be seen when the short queries are used. Significance values are also provided for Figure A.5. Significance was computed using the paired Wilcoxon test at $p = 0.05$ and is denoted using the black and gray bars at the bottom of the figures. Black marks denote values of n for which *CORI-UMass* significantly outperformed *CORI-UVA*; gray marks denote values of n for which *CORI-UVA* significantly outperformed *CORI-UMass*; no mark denotes no significant difference.

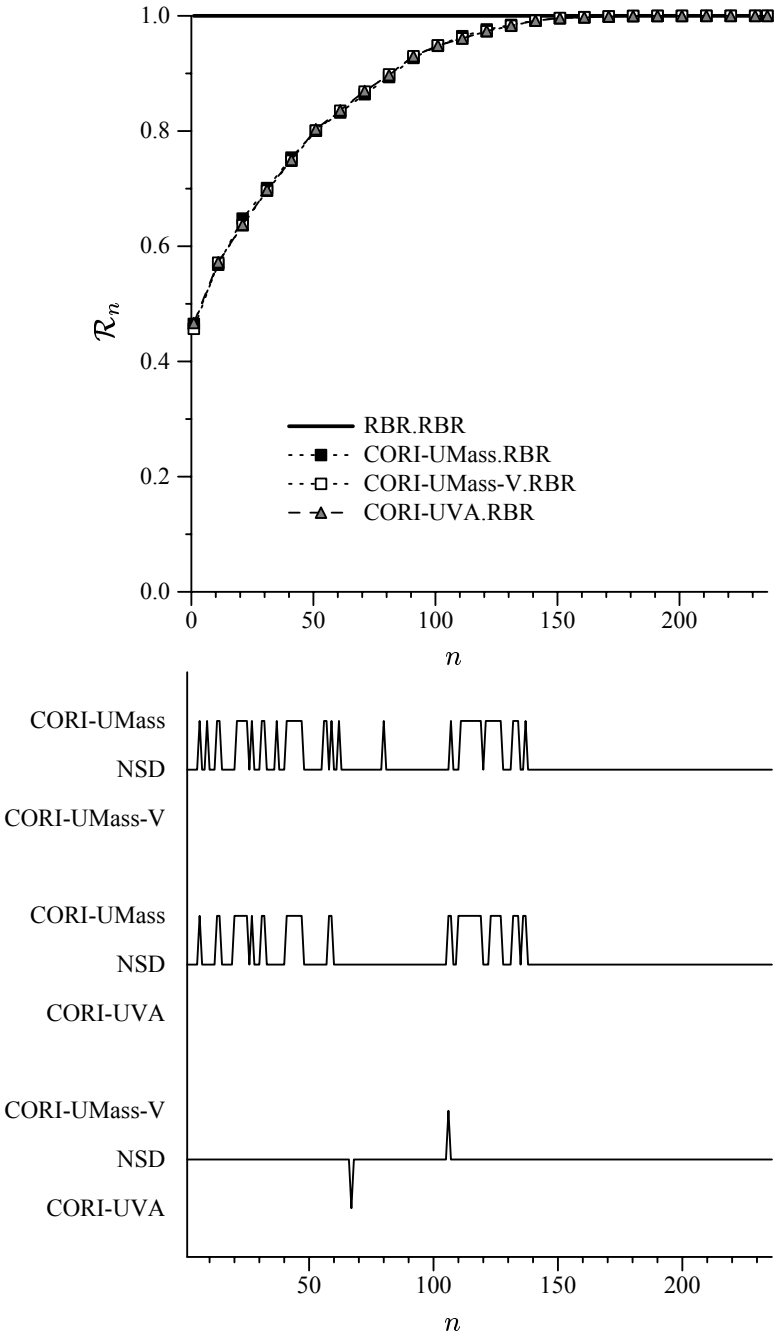
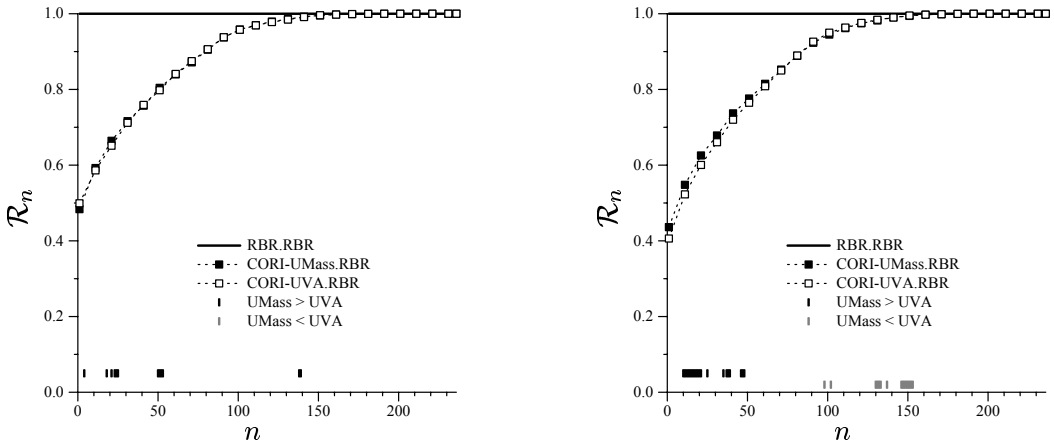
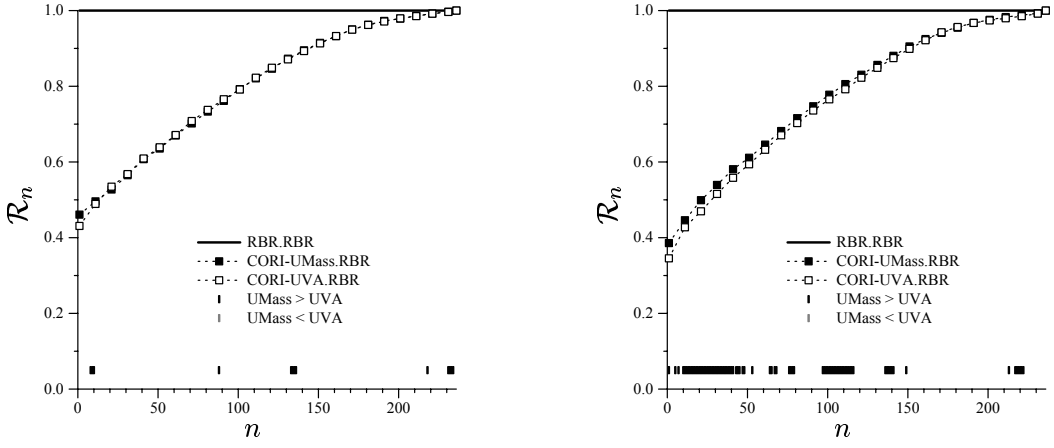


Figure A.4: Three versions of *CORI*, plus significance of comparison. *SYM-236* testbed, very long queries.



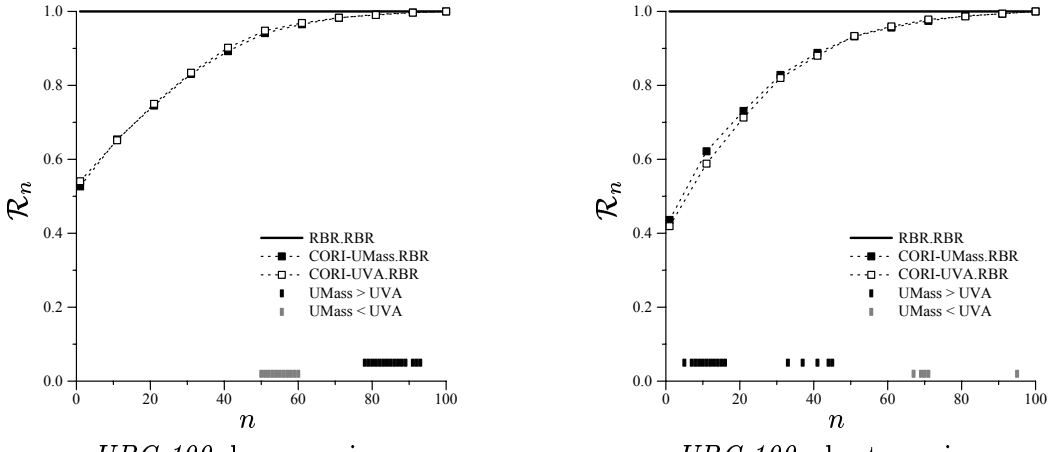
SYM-236, long queries

SYM-236, short queries



UDC-236, long queries

UDC-236, short queries



UBC-100, long queries

UBC-100, short queries

Figure A.5: A comparison of the UVA and UMass implementations of CORI.

B

Significance Testing

B.1 Paired Data

In the experiments presented here, the comparisons between competing approaches were performed using the same set of queries. Therefore, the performance results for a query q_i in one approach are *paired* with the results for q_i in the second approach. Significance tests that assume independent random samples are not appropriate in this case because the variability of the performance of an approach on different queries obscures the difference between the two approaches. Instead, paired significance tests were used.

Descriptions of paired significances tests can be found in many statistics textbooks. The use of these approaches in information retrieval was summarized by Hull [Hul93].

When performing a paired significance test, we assume that two retrieval techniques A and B are being compared and that at a given evaluation point, both A and B have some evaluation score for each query, using the same evaluation measure. A_i is the score of approach A and B_i is the score of B for query q_i.

B.2 Paired t-test

This description is a summary of the definitions given in Hull [Hul93], Ott [Ott93] and McClave and Dietrich [MF94] but can be found in many intermediate statistics textbooks.

Define $D_i = A_i - B_i$ for queries $i = 1 \ldots n$. The null hypothesis is that there is no difference between the two approaches.

Here we summarize the one-tailed test for determining of approach A is significantly better than approach B. Let $D_i = A_i - B_i$ for queries q_i, $i = 1 \ldots n$. The null hypothesis is that there is no difference between the two approaches.

$$H_0 : \quad \mu_A = \mu_B$$

$$H_a : \quad \mu_A > \mu_B$$

$$\text{Test Statistic} : \quad t = \frac{\overline{D}}{s(D_i)/\sqrt{n}}, \quad \text{where}$$

$$\overline{D} = \frac{1}{n}\sum_{i=1}^{n} D_i \quad \text{is the sample mean of the paired differences and}$$

$$s(D_i) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i - \overline{D})^2} \quad \text{is the sample standard deviation.}$$

For a specified value of α and $n-1$ degrees of freedom, reject H_0 if $t > t_\alpha$.

The paired t-test assumes that the differences D_i are normally distributed and becomes less reliable if that assumption does not hold. This distribution under H_0 is Student's t distribution with $n-1$ degrees of freedom.

B.3 Paired Wilcoxon Signed-Rank Test

The paired Wilcoxon signed-rank test is a non-parametric test and does not have the normality assumption of the paired t-test. This test is based upon ranking the differences between competing approaches.

Assume that we wish to determine if some retrieval or selection approach B tends to outperform an alternate approach A over a set of n queries. This comparison calls for a one-tailed comparison.

First, we define $D_i = A_i - B_i$ for queries q_i, $i = 1 \ldots n$. A_i and B_i are respectively effectiveness scores for approaches A and B applied to query q_i. We discard any queries for which $D_i = 0$, leaving k queries. Next sort the absolute values of the differences, keeping track of the sign for later use. Rank the sorted differences, using the midrank in case of ties. T_+ and T_-, defined below, are calculated by summing the ranks of positive and negative differences, respectively.

$$\begin{aligned} k &= \text{the number of queries for which } D_i \neq 0. \\ T_+ &= \text{the sum of the ranks of positive differences.} \\ T_- &= \text{the sum of the ranks of negative differences.} \end{aligned}$$

If approach B tends to outperform approach A, we would expect that for paired values $B_i > A_i$, yielding a larger number of negative D_i values. As a result, T_- will be large and T_+ will be small. If T_+ is below a specified threshold, we will accept the hypothesis that approach B tends to perform approach A.

Given the defined values, the Wilcoxon test is performed as follows. The test is different for $k \leq 25$ and $k > 25$.

H_0 : There is no difference between approach A and approach B.

H_a : The effectiveness of approach B tends to be greater than that of approach A.

Test Statistic ($k \leq 25$) : T_+

For a specified value of α, and number of non-zero differences k, reject H_0 if the value of T_+

is less than or equal to the appropriate entry in a table of critical values for the Wilcoxon signed-rank test.

$$\text{Test Statistic } (k > 25): \qquad z = \frac{T_+ - \frac{k(k+1)}{4}}{\sqrt{\frac{k(k+1)(2k+1)}{24}}}$$

For a specified value of α, reject H_0 if $z < -z_\alpha$.

C

Hypergeometric Distribution

The following is taken from [Lar74] but can be found in any introductory probability textbook.

Definition 2 *An urn contains M balls of which W are white. Let X denote the number of white balls that occur in a sample of n balls drawn at random from the urn without replacement. X is called the hypergeometric random variable.*

Theorem 6 *If X is the hypergeometric random variable, then*

$$p_X(k) = \frac{\begin{pmatrix} W \\ k \end{pmatrix} \begin{pmatrix} M-W \\ n-k \end{pmatrix}}{\begin{pmatrix} M \\ n \end{pmatrix}}, k = 0, 1, ..., n.$$

The mean, μ_X, and variance, σ_X^2, of the hypergeometric random variable X are given by

$$\mu_X = \frac{nW}{M}$$

and

$$\sigma_X^2 = \frac{nW}{M} \cdot \frac{M-W}{M} \cdot \frac{M-n}{M-1}$$

respectively.

Bibliography

[ALSF97] Ghaleb Abdulla, Binzhang Liu, Rani Saad, and Edward A. Fox. Characterizing World Wide Web Queries. Technical Report TR-97-04, Virginia Polytechnic Institute and State University, Department of Computer Science, 1997.

[BAS93] Chris Buckley, James Allan, and Gerard Salton. Automatic Routing and Adhoc Retrieval Using SMART : TREC 2. In *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 45–56, September 1993.

[Bau97] Christoph Baumgarten. A Probabilistic Model for Distributed Information Retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 258–266, July 1997.

[Bau99] Christoph Baumgarten. A Probabilistic Solution to the Selection and Fusion Problem in Distributed Information Retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246–253, August 1999.

[BBDH00] Krishna Bharat, Andrei Broder, Jefferey Dean, and Monika R. Henzinger. A Comparison of Techniques to find Mirrored Hosts on the WWW. *Journal of the American Society for Information Science*, 51(12):1114–1122, 2000.

[BDH+94] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. Harvest: A Scalable, Customizable Discovery and Ac-

cess System. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado–Boulder, August 1994.

[BDH+95] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. The Harvest Information Discovery and Access System. *Computer Networks and ISDN Systems*, 28(1-2):119–125, 1995.

[BDMS94] C. Mic Bowman, Peter B. Danzig, Udi Manber, and Michael F. Schwartz. Scalable Internet Resource Discovery: Research Problems and Approaches. *Communications of the ACM*, 37(8):98–107, 114, 1994.

[BKCQ93] N. J. Belkin, P. Kantor, C. Cool, and R. Quatrain. Combining Evidence for Information Retrieval. In *TREC2, Proceedings of the Second Text REtrieval Conference*, pages 35–44, 1993.

[BKFS95] Nicholas J. Belkin, Paul Kantor, Edward A. Fox, and Joseph A. Shaw. Combining the Evidence of Multiple Query Representations for Information Retrieval. *Information Processing & Management*, 31(3):431–448, 1995.

[BP97] Michael Buckland and Christian Plaunt. Selecting Libraries, Selecting Documents, Selecting Data. In *Proceedings of the International Symposium on Research, Development & Practice in Digital Libraries, ISDL'97*, November 1997.

[BSA92] Chris Buckley, Gerard Salton, and James Allan. Automatic Retrieval With Locality Information Using SMART. In *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 59–72, November 1992.

[Buc92] Chris Buckley. SMART version 11.0, 1992. ftp://ftp.cs.cornell.edu/pub/-smart/.

[Buc95] Michael Buckland. Searching Multiple Digital Libraries: A Design Analysis.

http://www.sims.berkeley.edu/research/basis/oasis/multisrch.html, November 1995.

[BV00] Chris Buckley and Ellen M. Voorhees. Evaluating Evaluation Measure Stability. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval*, pages 33–40, July 2000.

[CBH00] Nick Craswell, Peter Bailey, and David Hawking. Server Selection on the World Wide Web. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 37–46, June 2000.

[CCD99] Jamie Callan, Margaret Connell, and Aiqun Du. Automatic Discovery of Language Models for Text Databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 479–490, June 1999.

[CCH92] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY Retrieval System. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[CH95] Anil S. Chakravarthy and Kenneth B. Haase. NetSerf: Using Semantic Knowledge to Find Internet Information Archives. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, July 1995.

[CHT99] Nick Craswell, David Hawking, and Paul Thistlewaite. Merging Results from Isolated Search Engines. In *Proceedings of the Tenth Australasian Database Conference*, pages 189–200, January 1999.

[CLC95] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching Distributed Collections with Inference Networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, July 1995.

[CPFC00] Jamie Callan, Allison L. Powell, James C. French, and Margaret Connell. The Effects of Query-Based Sampling on Automatic Database Selection Algorithms. Technical Report CMU-LTI-00-162, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2000.

[DAA99] R. Dolin, D. Agrawal, and E. El Abbadi. Scalable Collection Summarization and Selection. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, pages 49–58, August 1999.

[DAAD97] R. Dolin, D. Agrawal, A. El Abbadi, and L. Dillon. Pharos: a Scalable Distributed Architecture for Locating Heterogeneous Information Sources. In *Proceedings of the Sixth International Conference on Information and Knowledge Management*, pages 348–355, November 1997.

[DAAP98] R. Dolin, D. Agrawal, E. El Abbadi, and J. Pearlman. Using Automated Classification for Summarizing and Selecting Heterogeneous Information Sources. *D-Lib Magazine*, January 1998. http://www.dlib.org/.

[DH97] Daniel Dreilinger and Adele E. Howe. Experiences with Selecting Search Engines Using MetaSearch. *ACM Transactions on Information Systems*, 15(3):195–222, 1997.

[DL00] J. R. Davis and C. Lagoze. NCSTRL: Design and Deployment of a Globally Distributed Digital Library. *JASIS*, 51(3):273–280, 2000.

[dMSZ98] Owen de Kretser, Alistair Moffat, Tim Shimmin, and Justin Zobel. Methodologies for Distributed Information Retrieval. In *Proceedings of the 18th International Conference on Distributed Computing Systems*, pages 66–73, 1998.

[DT99] Daryl J. D'Souza and James A. Thom. Collection Selection Using n-Term Indexing. In *Proceedings of the 2nd International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'99)*, March 1999.

[Emm99] Travis Emmitt. Cue-Validity Variance Database Selection Algorithm Evaluation and Enhancement. Master's project report. Department of Computer Science, University of Virginia., August 1999.

[FKS+92] Edward A. Fox, M. Prabhakar Koushik, Joseph Shaw, Russell Modlin, and Durgesh Rao. Combining Evidence from Multiple Searches. In *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 319–328, November 1992.

[FP99] James C. French and Allison L. Powell. Metrics for Evaluating Database Selection Techniques. In *Proceedings of the International Workshop on Internet Data Management (IDM'99) at the 10th International Conference and Workshop on Database and Expert Systems Applications (DEXA'99)*, September 1999.

[FP00] James C. French and Allison L. Powell. Metrics for Evaluating Database Selection Techniques. *World Wide Web*, 3(3), 2000.

[FPC99a] James C. French, Allison L. Powell, and Jamie Callan. Effective and Efficient Automatic Database Selection. Technical Report CS-99-08, Department of Computer Science, University of Virginia, February 1999.

[FPC+99b] James C. French, Allison L. Powell, Jamie Callan, Charles L. Viles, Travis Emmitt, Kevin J. Prey, and Yun Mou. Comparing the Performance of Database Selection Algorithms. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245, August 1999.

[FPV+98] James C. French, Allison L. Powell, Charles L. Viles, Travis Emmitt, and Kevin J. Prey. Evaluating Database Selection Techniques: A Testbed and Experiment. In *Proceedings of the 21st Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval, pages 121–129, August 1998.

[FS93] Edward A. Fox and Joseph A. Shaw. Combination of Multiple Searches. In *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 243–252, September 1993.

[Fuh99] Norbert Fuhr. A Decision-Theoretic Approach to Database Selection in Networked IR. *ACM Transactions on Information Systems*, 17(3):229–249, 1999.

[FV96] James C. French and Charles L. Viles. Ensuring Retrieval Effectiveness in Distributed Digital Libraries. *Journal of Visual Communication and Image Representation*, 7(1):61–73, 1996.

[GCGMP97] Luis Gravano, Chen-Chuan K. Chang, Hector Garcia-Molina, and Andreas Paepcke. Starts: Stanford proposal for internet meta-searching. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 207–218, May 1997.

[GGM95] Luis Gravano and Héctor García-Molina. Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. In *Proceedings of the 21st VLDB Conference*, pages 78–89, 1995.

[GGM97] Luis Gravano and Héctor García-Molina. Merging Ranks from Heterogeneous Internet Sources. In *Proceedings of the 23rd VLDB Conference*, pages 196–205, 1997.

[GGMT94a] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. Precision and Recall of GlOSS Estimators for Database Discovery. In *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems*, pages 103–106, September 1994.

[GGMT94b] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. The Effectiveness of GlOSS for the Text Database Discovery Problem. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 126–137, June 1994.

[GGMT99] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. GlOSS: Text-Source Discovery over the Internet. *ACM Transactions on Database Systems*, 24(2):229–264, 1999.

[Gib76] J. D. Gibbons. *Nonparametric Methods for Quantative Analysis*. Holt, Rinehart and Winston, 1976.

[Gol96] Jeffrey L. Goldberg. CDM: An Approach to Learning in Text Categorization. *International Journal on Artificial Intelligence Tools*, 5(1–2):229–253, 1996.

[GP99] Michael Gordon and Praveen Pathak. Finding Information on the World Wide Web: the Retrieval Effectiveness of Search Engines. *Information Processing & Management*, 35(2):141–180, 1999.

[GWG96] Susan Gauch, Guijun Wang, and Mario Gomez. ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines. *Journal of Universal Computing*, 2(9):637–649, 1996.

[Har95] Donna K. Harman, editor. *Proceedings of the Fourth Text Retrieval Conference (TREC-4)*, Gaithersburg, MD, November 1995. Department of Commerce, National Institute of Standards and Technology. NIST Special Publication 500-236.

[Har96] Donna Harman. Overview of the Fourth Text REtrieval Conference (TREC-4). In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, 1996.

[HCT98] David Hawking, Nick Craswell, and Paul Thistlewaite. Overview of TREC-7

Very Large Collection Track. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 91–104, November 1998.

[HCTH99] David Hawking, Nick Craswell, Paul Thistlewaite, and Donna Harman. Results and Challenges in Web Search Evaluation. In *Proceedings of WWW8*, pages 243–252, 1999.

[HT99] David Hawking and Paul Thistlewaite. Methods for Information Server Selection. *ACM Transactions on Information Systems*, 17(1):40–76, 1999.

[Hul93] David Hull. Using Statistical Testing in the Evaluation of Retrieval Experiments. In *Proceedings of the 16th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–338, June 1993.

[HVCB99] David Hawking, Ellen Voorhees, Nick Craswell, and Peter Bailey. Overview of the TREC-8 Web Track. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, November 1999.

[JSBS98] Major Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. Real Life Information Retrieval: A Study of User Queries on the Web. *SIGIR Forum*, 32(1):5–17, 1998.

[KMT+82] J. Katzer, M. J. McGill, J. A. Tessier, W. Frakes, and P. DasGupta. A Study of the Overlap Among Document Representations. *Information Technology: Research and Development*, 2:261–274, 1982.

[Lar74] H. J. Larson. *Introduction to Probability Theory and Statistical Inference, (2e).* John Wiley & Sons, Inc., 1974.

[LCC96] Zhihong Lu, James P. Callan, and W. Bruce Croft. Measures in Collection Ranking Evaluation. Technical Report TR-96-39, Computer Science Department, University of Massachusetts, 1996.

[Lee95] Joon Ho Lee. Combining Multiple Evidence from Different Properties of Weighting Schemes. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 180–188, July 1995.

[Lee97] Joon Ho Lee. Analyses of Multiple Evidence Combination. In *Proceedings of the 20th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–276, July 1997.

[LG98a] Steve Lawrence and C. Lee Giles. Context and Page Analysis for Improved Web Search. *IEEE Internet Computing*, 2(4):38–46, 1998.

[LG98b] Steve Lawrence and C. Lee Giles. Inquirus, The NECI Meta Search Engine. In *Seventh International World Wide Web Conference*, pages 95–105, 1998.

[LG99] Steve Lawrence and C. Lee Giles. Accessibility of Information on the Web. *Nature*, 400:107–109, 8 July 1999.

[Los95] R. M. Losee. Determining Information Retrieval and Filtering Performance without Experimentation. *Information Processing & Management*, 31(4):555–572, 1995.

[LXLN99] Yong Lin, Jian Xu, Ee-Peng Lim, and Wee-Keong Ng. ZBroker: A Query Routing Broker for Z39.50 Databases. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, pages 202–209, November 1999.

[LYM+99] King-Lup Liu, Clement Yu, Weiyi Meng, Wensheng Wu, and Naphtali Rishe. A Statistical Method for Estimating the Usefulness of Text Databases. Technical report, Department of EECS, University of Illinois at Chicago, July 1999.

[MB97] Udi Manber and Peter A. Bigot. The Search Broker. In *Proceedings of the First Usenix Symposium on Internet Technologies and Systems*, December 1997.

[MF94] James T. McClave and Frank H. Dietrich, II. *Statistics.* Dellen, 6th. edition, 1994.

[Mil95] George A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[MLY+98] Weiyi Meng, King-Lup Liu, Clement Yu, Xiaodong Wang, Yuhsi Chang, and Naphtali Rishe. Determining Text Databases to Search in the Internet. In *Proceedings of the 24th VLDB Conference*, pages 14–25, 1998.

[MLY+99] Weiyi Meng, King-Lup Liu, Clement Yu, Wensheng Wu, and Naphtali Rishe. Estimating the Usefulness of Search Engines. In *15th International Conference on Data Engineering*, pages 146–153, March 1999.

[MMF00] Gary A. Monroe, David R. Mikesell, and James C. French. Determining Stopping Criteria in the Generation of Web-Derived Language Models. Technical Report CS-2000-30, Department of Computer Science, University of Virginia, 2000.

[Mon00] Gary A. Monroe. Investigating Language Models of Web Databases Generated Using Query-Based Sampling Techniques. Master's project report. Department of Computer Science, University of Virginia., 2000.

[MZ95] Alistair Moffat and Justin Zobel. Information Retrieval Systems for Large Document Collections. In *Proceedings of the Third Text Retrieval Conference (TREC-3)*, pages 85–94, 1995.

[ODL93] Katia Obraczka, Peter B. Danzig, and Shih-Hao Li. Internet Resource Discovery Services. *IEEE Computer*, 26(9):8–22, 1993.

[OF00] Edward K. O'Neil and James C. French. A Description of the LAMB Web-Derived Language Model Builder. Technical Report CS-2000-31, Department of Computer Science, University of Virginia, 2000.

[Ott93] R. Lyman Ott. *An Introduction to Statistical Methods and Data Analysis.* Duxbury Press, 4th. edition, 1993.

[PF98] James Powell and Edward A. Fox. Multilingual Federated Searching Across Heterogeneous Collections. *D-Lib Magazine*, September 1998. `http://www.dlib.org`.

[PFC+00] Allison L. Powell, James C. French, Jamie Callan, Margaret Connell, and Charles L. Viles. The Impact of Database Selection on Distributed Searching. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval*, pages 232–239, July 2000.

[RC95] T. B. Rajashekar and W. Bruce Croft. Combining Automatic and Manual Index Representations in Probabilistic Retrieval. *Journal of the American Society for Information Science*, 46(4):272–283, 1995.

[RC99] Judi Repman and Randal D. Carlson. Surviving the Storm: Using Metasearch Engines Effectively. *Computers in Libraries*, 19(5), 1999.

[RJ76] S. E. Robertson and K. Sparck Jones. Relevance Weighting of Search Terms. *Journal fo the American Society for Information Science*, 27(3):129–146, May-June 1976.

[RW94] S. E. Robertson and S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, 1994.

[RWJ+94] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, 1994.

[Sal71] Gerard Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[Sal81] Gerard Salton. A Blueprint for Automatic Indexing. *SIGIR Forum*, 16(2):22–38, 1981.

[Sal91] Gerard Salton. Developments in Automatic Text Retrieval. *Science*, 253(5023):974–980, August 1991.

[SAM+99] Amit Singhal, Steve Abney, Michiel Macchiani, Michael Collins, Donald Hindle, and Fernando Pereira. AT&T at TREC-8. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, November 1999.

[SB88] Gerard Salton and Christopher Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[SBM96] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, August 1996.

[SC97] Alan F. Smeaton and Francis Crimmins. Relevance Feedback and Query Expansion for Searching the Web: A Model for Searching a Digital Library. In *First European Conference on Research and Advanced Technology for Digital Libraries, ECDL'97*, pages 99–112, September 1997.

[Sch93] Michael F. Schwartz. Internet Resource Discovery at the University of Colorado. *IEEE Computer Magazine*, 26(9):25–35, 1993.

[SDW+94] Mark A. Sheldon, Andrzej Duda, Ron Weiss, Jr. James W. O'Toole, and David K. Gifford. Content Routing for Distributed Information Servers. In *Proceedings of the 4th International Conference on Extending Database Technology*, 1994.

[SDWG95] Mark A. Sheldon, Andrzej Duda, Ron Weiss, and David K. Gifford. Discover: A Resource Discovery System Based on Content Routing. *Computer Networks and ISDN Systems*, 27:953–972, 1995.

[SE95] Erik Selberg and Oren Etzioni. Multi-Service Search and Comparison Using the MetaCrawler. In *Proceedings of the 1995 World Wide Web Conference*, 1995.

[SE00] Atsushi Sugiura and Oren Etzioni. Query Routing for Web Search Engines: Architecture and Experiments. In *Proceedings of the 9th International World Wide Web Conference*, May 2000.

[SEKN92] Michael F. Schwartz, Alan Emtage, Brewster Kahle, and B. Clifford Neuman. A Comparison of Internet Resource Discovery Approaches. *Computing Systems*, 5(4):461–493, 1992.

[SHMM99] Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz. Analysis of a Very Large Web Search Engine Query Log. *SIGIR Forum*, 33(1):6–12, 1999.

[SK88] T. Saracevic and P. Kantor. A Study of Information Seeking and Retrieving. III. Searchers, Searches, Overlap. *Journal of the American Society for Information Science*, 39(3):197–216, 1988.

[SM81] Edward E. Smith and Douglas L. Medin. *Categories and Concepts*, pages 78–81, 89–93. Harvard University Press, 1981.

[SM83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, New York, 1983.

[SPM+00] Rashmi Srinivasa, Tram Phan, Nisanti Mohanraj, Allison L. Powell, and James C. French. Database Selection Using Document and Collection Term Frequencies. Technical Report CS-2000-32, Department of Computer Science, University of Virginia, 2000.

[SS97] Amanda Spink and Tefko Saracevic. Interaction in Information Retrieval: Selection and Effectiveness of Search Terms. *Journal of the American Society for Information Science*, 48(8):741–761, 1997.

[TC90] Howard R. Turtle and W. Bruce Croft. Inference Networks for Document Retrieval. In *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 1–24, 1990.

[TC91] H. Turtle and W. Croft. Evaluation of an Inference Network-Based Retrieval Model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[TC92] Howard R. Turtle and W. Bruce Croft. A Comparison of Text Retrieval Models. *The Computer Journal*, 35(3):279–289, 1992.

[TGL+97] Anthony Tomasic, Luis Gravano, Calvin Lue, Peter Schwarz, and Laura Haas. Data Structures for Efficient Broker Implementation. *ACM Transactions on Information Systems*, 15(3):223–253, 1997.

[VF95a] Charles L. Viles and James C. French. TREC4 Experiments Using DRIFT. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, November 1995.

[VF95b] Charles L. Viles and James C. French. Dissemination of Collection Wide Information in a Distributed Information Retrieval System. In *Proceedings of the 18th International Conference on Research and Development in Information Retrieval*, pages 12–20, July 1995.

[VGJL94] Ellen Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. The Collection Fusion Problem. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 95–104, November 1994.

[VGJL95] Ellen Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning Collection Fusion Strategies. In *Proceedings of the 18th International Conference on Research and Development in Information Retrieval*, pages 172–179, 1995.

[VH98] Ellen Voorhees and Donna Harman. Overview of the Seventh Text REtrieval Conference (TREC-7). In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 1–23, 1998.

[Voo95] Ellen M. Voorhees. Siemens TREC-4 Report: Further Experiments with Database Merging. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 121–130, November 1995.

[Voo96] Ellen Voorhees. The TREC-5 Database Merging Track. In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, November 1996.

[Voo00] Ellen M. Voorhees. Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness. *Information Processing & Management*, 36(5):697–716, 2000.

[VT97] Ellen M. Voorhees and Richard M. Tong. Multiple Search Engines in Database Merging. In *Proceedings of the Second ACM International Conference on Digital Libraries*, pages 93–102, July 1997.

[WFPS94] Nikolaus Walczuch, Norbert Fuhr, Michael Pollman, and Birgit Sievers. Routing and Ad-hoc Retrieval with the TREC-3 Collection in a Loosely Federated Environment. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 135–144, November 1994.

[XC98] Jinxi Xu and Jamie Callan. Effective Retrieval with Distributed Collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–120, August 1998.

[XC99] Jinxi Xu and W. Bruce Croft. Cluster-based Language Models for Distributed Retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 254–261, August 1999.

[XCLN98] Jian Xu, Yinyan Cao, Ee-Peng Lim, and Wee-Keong Ng. Database Selection Techniques for Routing Bibliographic Queries. In *Proceedings of the Third ACM International Conference on Digital Libraries*, pages 264–273, June 1998.

[YGM95] Tak W. Yan and Héctor García-Molina. Duplicate Removal in Information Dissemination. In *Proceedings of the 21st VLDB Conference*, pages 66–77, 1995.

[YL97] Budi Yuwono and Dik L. Lee. Server Ranking for Distributed Text Retrieval Systems on Internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications*, pages 41–49, April 1997.

[YLW+99] Clement Yu, King-Lup Liu, Wensheng Wu, Weiyi Meng, and Naphtali Rishe. Finding the Most Similar Documents across Multiple Text Databases. In *Proceedings of the IEEE Conference on Advances in Digital Libraries (ADL'99)*, pages 150–162, May 1999.

[YML+99] Clement Yu, Weiyi Meng, King-Lup Liu, Wensheng Wu, and Naphtali Rishe. Efficient and Effective MetaSearch for a Large Number of Text Databases. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, pages 217–224, November 1999.

[YR98] Ronald R. Yager and Alexander Rybalov. On the fusion of documents from multiple collection information retrieval systems. *Journal of the American Society of Information Science*, 49(13):1177–1184, 1998.

[Zob97] Justin Zobel. Collection Selection via Lexicon Inspection. In *Proceedings of the Second Australian Document Computing Symposium*, pages 74–80, April 1997.