LOOKING FOR A HAYSTACK

SELECTING DATA SOURCES IN A DISTRIBUTED RETRIEVAL SYSTEM

Ryan Scherle

Submitted to the faculty of the Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in Computer Science and Cognitive Science

Indiana University

November 2006

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Doctoral Committee

David B. Leake, Ph.D.
(Principal Advisor, Computer Science)

Michael Gasser, Ph.D.
(Principal Advisor, Cognitive Science)

Gregory J. E. Rawlins, Ph.D.

October 5, 2006

Javed Mostafa, Ph.D.

Acknowledgements

When I was growing up, many of my friends owned the Atari 2600, an early home video game system. I wanted one, too. But my parents thought I should "learn something" instead of just playing games, so they bought me a TRS-80 Color Computer instead. That single purchase changed the course of my life, and I have never been far from a computer since. My parents have always provided this sort of gentle guidance, pointing me in useful directions while supporting the decisions I made, even when the decisions took me in a different direction than they had originally intended. For this, I will always be grateful.

While attending high school, I came across James Gleick's book *Chaos: Making a New Science*. This was my first exposure to the world of scientific research. Del Steinhart and Richard Abell, two of my science teachers, indulged me as I spent an inordinate amount of time reconstructing the experiments described in this book.

In college, my interest in computers remained, but I also had a keen interest in electrical engineering. Fortunately, Frank Young convinced me to major in computer science, which allowed me to work on much more interesting problems. Cary Laxer and Claude Anderson provided a wealth of knowledge, advice, and friendship. David Mutchler was instrumental in my decision to pursue research, and provided critical advice along the way.

My primary advisor, David B. Leake, is amazing. He has an uncanny ability to provide a critical idea or reference just when it is needed. His wordsmithing abilities are incredible. All of my writings, including this dissertation, are greatly improved due to his involvement. He also displayed an immense amount of patience while the dissertation process took much longer than expected. The rest of my dissertation committee, Michael Gasser, Gregory J. E. Rawlins, and Javed Mostafa, also provided critical insights along the way.

Members of the Indiana University Cranium lab provided camaraderie and stimulating conversation. These include: Douglas Eck, Andrew Kinley, Adam Leary, Kyle Wagner, David C. Wilson, Ana Maguitman, Thomas Reichherzer, and Jocelyn Bauer. Travis Bauer deserves to be singled out. In addition to being a source of stimulating ideas, Travis was an outstanding model of time management, proving that it is possible to balance a growing family and a serious research career.

Josh Goldberg was instrumental in determining the best way to vary databases with respect to a particular topic, which opened the door to the bulk of the experiments in this dissertation.

I would like to thank Northwestern University and the Intelligent Information Laboratory for providing a place to work and intellectual support during my brief sojourn in Chicago. Many thanks go to Kris Hammond, Jay Budzik, Shannon Bradshaw, Dave Franklin, and Josh Flachsbart.

The support staff of the Indiana University Computer Science department is wonderful. I'm certain that each of them helped me at some time during the process, but Rob Henderson, Laura Reed, and Sherry Kay deserve special thanks.

I worked in the Indiana University Digital Library Program during much of the dissertation process. Switching between my dissertation research and my work in this department each day was like stepping into a parallel universe. Libraries contain different types of data than Web pages, and this data often adds a new level of complexity to the search process. Working in this environment gave me a wonderful new perspective on the process of search. I am deeply indebted to all of the people there, including: Jon W. Dunn, Mark Notess, Jim Halliday, Donald Byrd, Jenn Riley, John Walsh, Tamara Lopez, David Jiao, Randall Floyd, and Michelle Dalmau.

Finally, eternal gratefulness goes to my wife, Amanda, and my children, Kedric, Graham, and Adrian. They waited patiently through hundreds of hours of writing, always greeting me with happy smiles when I returned to the real world. They have been the driving force that kept me going through the darkest times, and this dissertation would not exist without their support.

Abstract

The Internet contains billions of documents and thousands of systems for searching over these documents. Searching for a useful document can be as difficult as the proverbial search for a needle in a haystack. Each search engine provides access to a different collection of documents. Collections may be large or small, focused or comprehensive. Focused collections may be centered on any possible topic, and comprehensive collections typically have particular topical areas with higher concentrations of documents. Some of these collections overlap, but many documents are available from only a single collection. To find the most needles, one must first select the best haystacks.

This dissertation develops a framework for automatic selection of search engines. In this framework, the collection underlying each search engine is examined to determine how properties such as central topic, size, and degree of focus affect retrieval performance. When measured with appropriate techniques, these properties may be used to predict performance. A new distributed retrieval algorithm that takes advantage of this knowledge is presented and compared to existing retrieval algorithms.

Contents

List of Tables

List of Figures

1 Introduction

"Only librarians like to search; everyone else likes to find." – Roy Tennant

The Internet contains billions of documents and thousands of systems for searching over these documents. Searching for a given document can be as difficult as the proverbial search for a needle in a haystack. Each search system provides access to a different collection of documents. Collections may be large or small, focused or comprehensive. Focused collections may be centered on any possible topic, and comprehensive collections typically have particular topical areas with higher concentrations of documents. Some of these collections overlap, but many documents are available from only a single collection. Thus, to find the most (and best) needles, one must first select the best haystacks. The process of retrieving documents from many search engines is called distributed information retrieval. Fortunately, the problem of distributed retrieval has some features that make it easier to solve than a search through actual stacks of hay. Unlike haystacks, search engines provide some indication of their contents. In many cases, the search engine's home page describes the type of content. In other cases, the content may be determined by performing some test queries and browsing the results.

The primary question we will consider is: Given an information need and a set of search engines, is it possible to predict which search engines will return the best sets of documents? To answer this question, we will focus on two more specific questions: Under what circumstances will topic-specific search engines outperform general-purpose search engines, and what information is necessary to determine whether those conditions have been met?

1.1 The Need for Resource Selection

Another way to look at the search problem is to compare it with the way a search is performed in a library. If a patron in a library has a problem, he can ask a librarian for help. The librarian may point the patron to various catalogs, books, Web sites, or other reference literature. Based on discussions with the patron, the librarian can adjust the search to include other types of material, keep searching after the patron has left, and notify the patron if a particularly relevant resource becomes available. In the patron's home, using the Web alone, this kind of detailed interaction is not possible. The user must determine his own information need, state it in appropriate language, select all information resources he will use, query them individually, store results of searches, and keep track of the performance of searches. When most people search for information on the Web, their first reaction is to send a query to their favorite general-purpose search engine. They do not analyze whether a more specific search engine might be appropriate.

Typical queries on the Web are short, often one or two words (Lesk 1997; Nichols 1997). These short queries are often ambiguous, resulting in poor results from general-purpose search engines. For example, consider the owner of a moving company interested in buying heavy-duty dollies to move furniture. As a first query, he may simply search for "dolly." In response to this query, the first page of a recent search on the popular search engine Google[1] provided pages for singer Dolly Parton, the musical "Hello Dolly," "Dolly" the cloned sheep, and links to various pages about children's dolls.

A common approach to this problem is to add information to the query, making the topic of the query more specific and enabling general search engines to gather better information. Some search engines support this process by enabling users to add terms to narrow subsequent queries. Other approaches allow users to select word senses for ambiguous terms (Cheng & Wilensky 1997), automatically add context-based terms to queries (Budzik & Hammond 2000), or exploit user profiles to anticipate user preferences (Billsus & Pazzani 1999; Chen & Sycara 1998). Unfortunately, even with this extra information for disambiguation, the usefulness of results from general-purpose information sources may be limited by their need to perform well over a broad set of topics. If the topic being searched is not popular, or has a name sufficiently similar to a popular topic, it will be difficult to find. In Google, for example, a page that matches a user's needs perfectly may be buried in the result listing simply because there are not enough other pages pointing to it. For queries including "dolly", for example, Google is biased towards results related to Dolly Parton, because of the popularity of Dolly Parton pages. If a system can determine that the context of a query is moving furniture, it can instead direct the query to the search engine at Handtrucks.com[2], which provides access to numerous types of moving dollies.

Searches that depend on time-sensitive information can also benefit from automatic search engine selection. A sports fan wanting to know the outcome of a baseball game may simply provide a query containing the names of two teams, perhaps "red sox yankees". When sent to a general-purpose search engine, this query produces pages describing the virtues of the two teams, pages detailing the history of their rivalry, and pages that provide general baseball information. But

[1]http://www.google.com
[2]http://www.handtrucks.com

general-purpose search engines have difficulty returning the score of a recent game. For that task, it would be more appropriate to route the query to a sports-oriented search engine[3] or a search engine that is frequently updated with current news stories[4].

The most obvious deficiency of large, centralized, general-purpose search engines is the fact that no one engine can contain all of the data in the world. Even though current search engines can crawl a sizable fraction of the Web, there is still data that is inaccessible. Many of the best documents are available from the "hidden Web". Bergman (2000) estimated that the size of the hidden Web was 500 times larger than the "surface Web" accessible by centralized search systems. Data on the hidden Web can be inaccessible for a variety of reasons, including:

1. The owner of the data wants to mediate access through their site, and uses the "robot exclusion protocol" (i.e., a robots.txt file) to declare that some documents should not be indexed. Most large centralized search engines follow the "robot exclusion protocol", and refrain from following links to content when the authors of the content have declared that it should not be indexed. For example, Ipeirotis and Gravano (2002) show that Google does not index documents in the database at the CANCERLIT[5] site. Users searching for detailed medical information regarding cancer would get much better results by using the CANCERLIT site's internal search engine than a more general-purpose engine like Google.

2. The owner of the data wants only paying customers to access the data. Access to the data may be restricted by a username and password, or by a set of allowed IP addresses. Even though a centralized search engine could pay for access to this type of data, it would be counterproductive to include the data in the centralized index, because most users of the search engine would not be allowed to access the data. This type of restriction has often

[3]e.g., http://www.espn.com
[4]e.g., http://news.google.com
[5]http://www.cancer.gov

been used for electronic journal articles and newspaper archives, but recently many of these organizations have begun making article summaries available for free.

3. The data is dynamically generated or based on a database, and cannot be crawled easily. Results of searches in library catalogs or real estate sites are examples of this.

4. The data is publicly available, but changes so frequently that the copy in a search engine is always outdated. This often happens for sites that report the news.

When content is not hidden, and is accessible via a centralized search system, it may still be difficult to retrieve. Even when a user has overcome the problems of off-topic results described above, the resultant documents may be at the wrong level of specificity. For many users, the documents returned by centralized search systems are at the appropriate level, but for novice users the broad summary documents returned by an encyclopedia-like site may be more appropriate. For users who are experts in a particular topic, highly-detailed journal articles may be more useful.

There are thousands of specialized search engines on the Web, each indexing a carefully-crafted set of documents relevant to a particular topic or audience. Many of these are listed in the weekly Internet Scout Report[6]. Unfortunately, the availability of specialized sources is not a panacea: to use them, the user must know which sources are available and where to locate them. Instead, users tend to focus entirely on general-purpose search engines, ignoring content that is available from more specific sites. Graham and Metaxas (2003), when researching the trust students have in Internet-based information, noted that

> If the survey question did not mention a particular Web site, almost all students immediately turned to a search engine. Many remained faithful to one search engine throughout the survey, even if it did not immediately provide the answer sought.

[6]http://scout.wisc.edu/

Similar results were found in a study of the mental models of novice Web searchers (Brandt & Uden 2003). It is unreasonable to expect a human searcher to remember the existence, contents, and addresses of thousands of topically-focused search systems. Human memories are not well suited to this type of information. Information retrieval systems, on the other hand, were designed to track information on thousands of documents. It is relatively simple to transform basic information retrieval algorithms to work at the level of collections rather than with individual documents. Information retrieval systems therefore are a form of "intelligence augmentation" (Engelbart 1962), extending the capabilities of the human mind,

If an automatic system can select the most appropriate set of search engines, the focus provided by these topically-focused search engines may provide a better set of result documents than using a single large, general-purpose search engine. Given sufficient bandwidth, it would be possible to skip the selection process, retrieve documents from all available search engines, and then filter the results according to user needs. However, that approach would destroy the benefits of the pre-focused document sets. It would require sophisticated filtering to substitute for the careful pre-selection already done in the creation of specialized search engines. Taken to an extreme, this filtering approach would simply provide another general-purpose search engine.

Automatically selecting sources requires considerable knowledge, and has been identified as an important problem in information retrieval (Baeza-Yates & Ribeiro-Neto 1999; Tenenbaum 2005). Even within a library, it can be difficult to find the correct sources. Libraries have a growing problem with managing multiple information providers. Library patrons should not need to remember which provider has which type of information (and which subscriptions each library holds). The library would like to integrate all of these providers into a single access point. The recent Open Archives Initiative (Lagoze & Van de Sompel 2001) is an attempt to make search possible across disparate collections, but uptake of the OAI standard is slow (Lagoze *et al.* 2006).

1.2 Project Outline

In a centralized retrieval system, the search engine simply tries to do the best it can with whatever query is presented. In a distributed system, however, a query can be analyzed to determine how well a source will be able to answer it. There are many factors that can affect whether a search system will be able to provide adequate results for a particular information need. The general approach taken throughout the dissertation is to create collections that vary on the property in question, determine methods for measuring the property, and then measure how variations in the property affect retrieval performance. Properties to be analyzed include:

- **Topic match between the query and collection:** Intuitively, a query should do well with a source that focuses on the query topic.
- **Absolute topic of the query:** Some topics are "easy", either because they deal with widely-available information, or because they deal with highly specific, unambiguous terms.
- **Size of collection:** In the early days of Web search, general-purpose search engines were constantly competing for the title of "largest collection". Increasing collection size allows for coverage of more topics, but at the possible expense of accuracy. Topically focused collections will typically be a small fraction of the size of general-purpose collections.
- **Collection focus:** Sources that are highly focused on a particular topic should be able to perform well for queries closely associated with that topic. Queries that are slightly off-topic may perform better with a source that is less focused, even if the topic does not match as closely.

Many of these factors are related. For example, large collections tend to have less focus. Collections that are highly focused may perform better for "difficult" topics, but be useless for "easy" topics. These factors will be examined individually, as well as in combinations.

1.3 Contributions of This Work

This research makes four main contributions. First, it demonstrates that distributed information retrieval can work with realistic collections composed of actual Web pages, rather than the more artificial collections used in the past.

Second, this work develops a methodology for evaluating how changes in a collection affect retrieval performance. Methods for generating collections that vary systematically on each property in question are examined. Measures that best reflect retrieval performance in a Web-based system are selected. Methods for interpreting the results are discussed.

Third, this research establishes how the basic properties of collections affect retrieval performance. When a collection is small and topically focused, high performance will be obtained only if there is a high degree of similarity between the central focus of the collection and the user's need. As collections become larger and/or less focused, there is a greater chance that they will perform well. If a search engine is known to be very large or contain a wide array of documents, there is little point in measuring its "central" topic. However, when a collection is smaller or more focused, the collection's central topic is much more important than the collection's size or the amount of focus in the collection.

Fourth, this work examines the effect of contextual information on retrieval from both centralized and distributed collections. Adding a small amount of contextual information to the query can aid retrieval of documents from any collection, but contextual information is unlikely to aid in the selection of collections in a distributed system.

To illustrate these contributions in a practical setting, an algorithm for a complete distributed retrieval system is presented, combining existing techniques with the insights gained from this work. In this algorithm, search engines are automatically sampled to obtain a representative set of documents. The sampled documents are analyzed to estimate the essential properties of the full collection. When a new query comes into the system, it is compared with each search engine's representation. The query is forwarded to all search engines that are predicted to return good result sets, and results from all engines are merged before display to the user.

1.4 Structure of the Dissertation

Chapter 2 describes the information retrieval process in more detail, for both centralized and distributed systems. Then Chapter 3 looks at how the central topic of a collection affects query performance. Chapters 4 and 5 examine how the size and focus of a collection affect query performance. In Chapter 6, the effects of automatic query expansion query are examined. Chapter 7 describes how the results of the preceding chapters can be used to build a distributed search system and examines how well such a system will perform. Chapter 8 summarizes the results of this work and provides a roadmap for future work.

2

Information Retrieval

Information retrieval is the process of taking a user's query and returning a set of documents that are expected to be relevant to the query. Before we get into a detailed description of information retrieval, though, it is useful to establish a common set of definitions for the terms that will be used throughout this dissertation.

Depending on the community one comes from, the terminology for the type of work presented in this dissertation may differ. "Distributed information retrieval" in the information retrieval community may be called "federated search" in the digital library community, although the problems being described are nearly identical. I will use the term "distributed information retrieval" to better separate it from the "centralized information retrieval" common in most search engines on the Web. I will also maintain the following definitions:

Document: A single Web page, referenced by a URL

Collection: A set of documents

Database: A set of documents

Query: A statement of information need as expressed by a user or automated searching system, represented as a set of keywords

Search algorithm: A process for ranking documents in a collection with respect to a query

Search engine: The combination of a particular collection and search algorithm

Within distributed information retrieval, the problem of selecting an appropriate search engine for a query has been given many names. "'Database selection" and "collection selection" have been used commonly.

These terms are useful for describing systems in which all individual engines use the same search algorithm, but on the Web, there are a wide variety of search algorithms in place. Since all access to topic-specific collections must be through their search systems, the database and search algorithm must be treated as an inseparable unit, and thus I will use the term "search engine selection" to mean not only the collection, but also the search algorithm and Web interface that give access to the collection. Within this dissertation, the search algorithm will be held constant, so we can focus on the effects of different collections. Therefore, the terms "collection" and "search engine" may be used interchangeably for convenience.

2.1 Centralized Information Retrieval

In a centralized information retrieval, a set of documents is processed to create an index. Processing steps typically include parsing each document to break it into a set of terms, performing a calculation to assign weights to the terms, and storing the terms in an index that can be efficiently searched. The details of weighting terms and retrieving documents from the index depend on the search model being used. The boolean and probabilistic search models have been used heavily, but by far the most popular is the vector space model created by Salton (1975). As described by a recent information retrieval textbook (Baeza-Yates & Ribeiro-Neto 1999):

> A large variety of alternative ranking methods have been compared to the vector model but the consensus seems to be that, in general, the vector model is either superior or almost as good as the known alternatives. Furthermore, it is simple and fast.

Many search engines use a processing engine based on simple vector space techniques, with only minor configuration changes to meet local needs. Typical modifications to the basic retrieval process include:

Stopword removal: Words that are common in English (or common within a particular domain) but typically contain no informational value are excluded, reducing the chances of documents falsely matching based on these words alone. Examples of common stopwords are "a", "the", "begin", and "whether".

Term stemming: Word suffixes are removed so that different version of the same word (e.g., "author" and "authored") are treated as the same term.

Phrase detection: The documents may be parsed to identify noun phrases and proper names. These phrases may be treated as single terms during retrieval to boost precision.

Two more substantial modifications to the vector space retrieval process have been used in limited settings. Latent Semantic Indexing (Berry, Dumais, & Letsche 1995) is a process of analyzing a document set to identify clusters of co-occurring terms, so documents may be retrieved even if they use language that differs slightly than the language used in the query. The major drawback of Latent Semantic Indexing is that it cannot handle incremental updates. When a new document is added to the collection, scores for all terms in the collection must be recomputed, which is a time-consuming process. For this reason, Latent Semantic Indexing is only used in a few applications where the document set changes very infrequently.

The second substantial modification to vector space retrieval is using link analysis to adjust the weights of documents. The popular search engine Google made a name for itself by leveraging the highly-connected nature of documents on the Web. Google's PageRank algorithm (Brin & Page 1998) takes advantage of the fact that documents on the Web cluster into "hubs" and "authorities" (Kleinberg 1999) that provide an indication of a page's popularity. In this dissertation, we will not be working with link analysis, because the collections held by small, topically focused search engines do not have enough interconnection for this type of analysis.

Evaluating Information Retrieval Systems

Each year, a Text REtrieval Conference (TREC) is held, allowing teams from various institutions to compare their retrieval systems on a common set of documents and information needs (Harman 1992). Evaluation in TREC conferences proceeds as follows: A collection of documents is given to the participating teams, and at some later point in time, a set of "topics", or statements of information needs. (These topics will be described in more detail in Chapter 3.) Each team uses their search engine to retrieve a set of result documents for each topic. All result documents for the topic are pooled, and a panel of human judges evaluates the relevance of each document with respect to the information need expressed in the topic. The resulting relevance scores are then used to determine how well each search engine performed. Many evaluation measures have been used over the years, but the most popular are:

Precision: the percentage of retrieved documents that are relevant

Recall: the percentage of relevant documents that are retrieved

P@N: examining only the first N documents retrieved, the percentage that are relevant.

Precision and recall can often be traded for each other. A search engine that is restricted to returning only a few documents will typically exhibit high precision and low recall, while a search engine that simply returns its entire collection in response to any query will exhibit high recall and low precision.

In the early years, TREC conferences focused on document sets that came from a single source (like a newswire service). These document sets often had idiosyncratic language structures, and did not reflect the diversity of documents available on the Web. As the prominence of the Web grew, the types of competitions falling under the TREC umbrella were expanded. The first TREC competition to feature a document set based on actual Web pages was TREC-8, in 1999 (Hawking *et al.* 1999).

2.2 Distributed Information Retrieval

Distributed information retrieval (DIR) uses multiple centralized retrieval systems, and employs a process for choosing which centralized systems are queried for any particular information need. The first distributed information systems grew out of work in distributed databases (Marcus 1983; Arens *et al.* 1993; Levy, Rajaraman, & Ordille 1996).

The typical steps followed by a DIR system include:

1. Locate search engines.

2. Determine a method for communicating with each search engine.

3. Form a representation of each search engine to store in the selection system.
4. When a query is received, apply a selection algorithm to the set of collection representatives, and forward the query to the collections that are expected to return the best results.
5. Apply a merging algorithm to combine results from the selected collections, and present the results to the user.

Early work on DIR systems (Gravano & García-Molina 1995; Xu & Callan 1998) focused on the question of whether collections could be searched more efficiently if they were stored in multiple (local) search indexes, rather than one large index. These systems worked primarily with collections generated based on the organization that created the documents. In some sense, this is a useful distinction, but it ignores the fact that some organizations (like news agencies) will produce documents with no discernable topical focus. This leads many of the collections under consideration to be less focused than actual collections in the wild (where even within an organization, there may be several search systems for documents pertaining to different topics). Another difference between these early experiments and actual Web collections is that no overlap was allowed between the experimental collections. The distributed collections were a strict partitioning of the centralized collection.

Using this type of configuration, Callan *et al.* (1995) found that DIR produces the same results as centralized IR when half of the distributed collections are searched. That is, when an appropriate selection algorithm is used, and collections comprising half of the documents of the centralized system are searched, overall retrieval results are similar to retrieval from a centralized collection containing all of the documents. Voorhees *et al.* (1995) found that when the number of documents to be used from each collection is guided by relevance judgments on a set of training queries, DIR is only 10 percent worse than CIR. However, this approach is expensive, as it requires the results of training queries to be manually judged for each search engine.

Distributed retrieval techniques have been used in other types of information retrieval with some success, including peer-to-peer retrieval systems (Lu & Callan 2005; Akavipat *et al.* 2006), text classification systems (Fu, Ke, & Mostafa 2005), and case-based reasoning systems (Leake & Sooriamurthi 2004).

2.3 Distributed Search on the Web

A number of DIR systems have been adapted for use with collections on the Web, including GlOSS (Gravano, García-Molina, & Tomasic 1994; Meng *et al.* 1999), EMIR (Kulyukin 1999), and OBIWAN (Zhu *et al.* 1999). These systems provide methods for routing queries to a collection of sources, but they depend on the sources to cooperate by providing indices or other data to a central distribution system, and require costly updating of central information as their contents change. On the Web, the assumption that search engines will want to cooperate with a centralized system has not held true. Search engines on the Web are often competing for traffic, and while providing information about their contents may help boost traffic, they are very reluctant to provide this type of information to a central authority. In fact, some search engines actively misrepresent their contents (using high-demand terms like "sex") in order to boost the amount of activity they receive. See (Callan, Connell, & Du 1999) for a good discussion. No distributed system on the Web has achieved the full cooperation of all sources it indexes. Unless the sources have an overriding economic reason to cooperate, they will not. In general, non-interference is the best that can be hoped for. Recent work in the information retrieval community has started to address the problem of representing sources without explicit cooperation. Most notably, Callan and Connell (2001) have developed methods for approximating the content of a server's collection by sending repeated small queries and analyzing the results.

When search engines have a desire to cooperate with a central authority, they must follow a standard protocol. Harvest (Bowman *et al.* 1995) and STARTS (Gravano *et al.* 1997) were early popular protocol, but most search engines that wish to make their content explicitly available for distributed retrieval are publishing records that follow the recent OAI-PMH protocol (Lagoze & Van de Sompel 2001). Any record published via this protocol may be picked up by an OAI harvester, such as the University of Michigan's OAIster[1] service (Hagedorn 2005). NCSTRL[2], one of the oldest DIR systems (Davis 1995), has recently begun conversion of its services to use the OAI-PMH protocol.

Meta Search

DIR systems that do not assume cooperation from their sources fall under the heading "meta-search engines". This approach was first taken by the MetaCrawler (Selberg & Etzioni 1995) to access search engines without explicit cooperation by simply forwarding queries to them and collating the results. Popular metasearch engines include Dogpile[3], Mamma[4], and IxQuick[5]. There are also a growing number of topical metasearch engines that collect results from several smaller topical search engines. For example, the WindowsSecrets[6] search engine collects tips on using Microsoft Windows from several other sites. Metasearch systems often ignore the differences in coverage of topics by search engines they index. Bandwidth constraints limit the number of search engines that can be queried, restricting the list of available search engines to a relatively small number of general-purpose engines.

[1]http://oaister.umdl.umich.edu/o/oaister/
[2]http://www.ncstrl.org/
[3]http://www.dogpile.com/
[4]http://www.mamma.com/
[5]http://ixquick.com/
[6]http://www.windowssecrets.com/

Metasearch systems with more manual control have been developed. The Sherlock tool (Montbriand 1999) included with Apple's MacOS, ProFusion (Gauch, Wang, & Gomez 1996), and Inquirus2 (Glover 2001) allow users to select a topic area, and the query is automatically sent to three or four search engines covering that subject. While these systems are useful, the user cannot be expected to correctly classify every query. Many users do not have the knowledge or desire to perform this task.

Some "personal" metasearch systems can be extended by the user to search new sources. These include Apple's Sherlock, Magellan Metasearch[7], and Watson[8] (Budzik & Hammond 2000). The creation of custom "plug-ins" allows search engines to be added to the system and configure them into searchable groups. In order to create such a plug-in, the users must be aware of a search engine's existence, and must manually code settings that allow searches to be performed. Adding too many plug-ins to a system like this can degrade performance, as additional bandwidth is used by each new search engine added to the system.

A few metasearch systems have attempted to intelligently select search engines. SavvySearch (Dreilinger & Howe 1997) addressed this problem by recording how well particular search engines handled past queries, and using vector-space retrieval to match new queries to search engines that did well with similar queries. ProFusion (Gauch, Wang, & Gomez 1996) used a category hierarchy to automatically categorize queries and select relevant search engines. Q-Pilot (Sugiura & Etzioni 2000) used a combination of home page keywords, back-link keywords, and database sampling to determine which search engines were most appropriate for each query. However, since these systems use hand-coded or computation-intensive representations, they are still restricted to using a relatively small number of search engines.

[7]http://sourceforge.net/projects/magellan2/
[8]http://www.intellext.com

Another approach to adding intelligence to metasearch systems is post-processing of results. Both Clusty[9] and KillerInfo[10] attempt to cluster search results into meaningful categories. Ask.com[11] takes a slightly different approach, using a pre-generated set of categories that are matched to the queries being answered.

2.4 Intelligent Information Systems

Various enhancements have been added to information retrieval systems in an effort to make them more "intelligent". Just-in-time information systems attempt to provide information as the user needs it, based on user behavior. The Remembrance Agent (Rhodes & Starner 1996) watches the information a user types in a text editor, and sends related queries to local databases including the local file system and email folders. The system has recently been adapted for use with the Web (Rhodes 2002). The Lumiere project monitors user behavior in Microsoft Office to predict user questions and provide information (Horvitz *et al.* 1998). The Watson system (Budzik & Hammond 2000) analyzes applications that are running on the user's desktop, automatically generates queries, and processes results in the background, so that information is available whenever the user decides to look at it. Code Broker (Ye & Fischer 2002) retrieves useful pieces of code as the user writes a computer program.

Tracking the interests of users has been shown to increase the effectiveness of information retrieval systems. Profile information can be used to automatically rank documents (Kaplan, Fenwick, & Chen 1993; Pazzani & Billsus 1997; Bollacker, Lawrence, & Giles 1999), or to select categories that interest the user (Mostafa *et al.* 1997; Pretschner & Gauch 1999). Calvin (Leake *et al.*

[9]http://clusty.com
[10]http://www.killerinfo.com
[11]http://www.ask.com

2000) tracks the ways in which users access documents, so the documents can be more easily retrieved in the future.

2.5 Cognitive Perspectives on Information Retrieval

Cognitive science an interdisciplinary field focusing on study of the human mind and the nature of intelligence. Often, cognitive science research confines itself to modeling the processes that occur solely within the mind. However, this approach has been questioned. Donald Norman (1988) notes that people often supplement knowledge in the mind by transferring portions of that knowledge to objects in the world. One of the simplest ways this can be accomplished is by writing a note. Information retrieval systems supplement knowledge in the mind by storing documents that are of interest to a user and making the documents easily accessible via simple keyword searches. If a user knows that the information he needs is available in a search engine, he may not make the effort to memorize this information.

Edwin Hutchins (Hutchins 1995) goes a step farther than Norman, arguing that any study of cognitive science should treat the mind as a single part of a complete system. The system may include the person, the physical environment, and any tools (physical, electronic, or mental) that the user has at his disposal. An information retrieval task should be viewed as a cooperation between the user and the information retrieval system.

The process of searching for information on the web is usually complex, with users performing many queries that increase in specificity. Users may make multiple visits to pages that result from a search, as the sites may point to each other and/or appear multiple times in the search results. Initial searches often target a relatively broad topic as the user solidifies their information need.

More specific searches occur when the user has a good idea what they want, and they are comparing several options. Males and females have slightly different search patterns, with females taking more time to digest each page, and males skipping around a bit more (Hotchkiss 2003).

Rhodes argues that each user implicitly performs cost/benefit analysis before performing a search. Depending on the situation, the user's expected cost/benefit ratio for searching the web may be higher or lower than the expected cost/benefit ratio of other search methods, such as visiting a library. The user constantly updates their cost/benefit evaluation as the search progresses, and newly discovered information, such as the availability of a relevant book in the local library, may cause a change of strategy midway through the search process (Rhodes 2000, pp. 41–44). Rhodes describes search systems in terms of a "ramping interface", which allows the user to easily manage the cost/benefit ratio (Rhodes 2000, pp. 56–59). A ramping interface displays search results in the most unobtrusive manner possible until the user decides that they are interested in obtaining more detail. At any time, the user can progress through steps that take slightly more effort and produce more detail about the search results. For example, in a typical search system, document titles are displayed in a larger font than document summaries, allowing the user to quickly skim the results. The user can expend a small amount of extra effort by reading summary information for the items that are of interest. In cases where the title and summary appear to be relevant, the user can expend a small amount of extra effort to click on the result and review the contents of the full document. In a distributed retrieval system, a ramping interface may start by providing the names of the selected search engines and a count of the number of documents returned by each engine, allowing the user to view results from individual engines or merged results from all engines.

Distributed retrieval systems face several challenges. Queries on the Web are relatively short, averaging just over 2 words in length (Lesk 1997; Spink & Jansen 2004). DIR systems typically need large queries (30 words or more) generated by experts to accurately select sources and retrieve relevant information. Because of the cost/benefit evaluations ratio mentioned above, users expect search engines to return results quickly, often in less than two seconds (Miller 1968). Another compounding factor is that users examine very few results from each search. More than 70 percent of the time, users only view the top 10 results from a search (Spink & Jansen 2004). Consider a typical Web search engine that returns 10 hits on each page of results. If none of the results on the first page are relevant, a user may check the second page, but if there are no relevant hits on that page, the user will often change their query or look elsewhere. This reasoning suggests that the threshold is at or above an initial precision of 0.05. Any time precision drops below this, users are likely to look elsewhere.

2.6 Information Retrieval in This Dissertation

The evaluations in this dissertation will be somewhat different from previous evaluations of DIR systems. Previous evaluations (French *et al.* 1999; Xu & Croft 1999) have focused on recall metrics (proportion of total relevant documents retrieved). On the Web, where there may be millions of documents relevant to a given query, recall has little meaning. This work will focus on obtaining high precision (proportion of documents retrieved that are relevant).

Most previous experiments on DIR systems assume that the collections are a method of partitioning a centralized collection. That is, each document from the centralized collection appears in exactly one collection in the distributed system. In the real world, this is not the case. While the “hidden Web” generally contains independent, non-overlapping databases, the public Web has opportunity for much overlap between the contents of differing databases. Some topically focused search engines may have a high degree of overlap, as they are indexing the same small set of documents available on their target topic. Other focused search engines provide access to documents internal to a particular site, none of which will be indexed by any other site on the Internet.

In this work, we will focus primarily on the act of selecting a collection, but the final system presented in Chapter 7 will combine all of these steps. To make the experiments manageable, some simplifications must be made. In a more general, fielded system, these simplifications will not always hold. However, it is assumed that they will hold often enough, or can be controlled enough, to make the experimental results applicable. The simplifications are:

1. **Context is always relevant:** The context and/or user profile generated by a user-tracking system may not be compatible with the query presented. For example, if a user working on financial statements suddenly remembers he was supposed to order a present for his niece, he may send a query to the system that has nothing to do with the currently available context, or his recorded preferences. For the purposes of this dissertation, it is assumed that any available context will always be relevant to the query. In a fully fielded system, this could be assured by allowing the user manual control of the system's use of context.

2. **All IR systems use vector-space retrieval:** There are several competing models for information retrieval systems. However, they have been shown to produce very similar results. Vector-space retrieval is the most popular model for Internet search engines, so it is reasonable to use this as a starting point.

3. **Bag-of-words is an adequate representation:** The basic representation being used for this project is a set of weighted keywords. This type of representation is traditional for information retrieval systems, and has been shown to provide adequate results in both single-source and distributed systems. It is assumed that adding more complex, knowledge-intensive representations would only improve effectiveness. In fact, the possibility of performing more

intense calculations on small sets of documents is one of the advantages of distributed retrieval. The weighted keyword representation will be used for all documents, contexts, and queries.

3

Choosing Collections Using Topic Similarity

An obvious way to choose search engines is by finding the engine whose collection most closely matches the topic of the current query. For example, if the user's query contains many terms related to sports, it seems reasonable to direct the query to a sports-oriented search engine, rather than a general-purpose search engine.

In general, distributed information retrieval (DIR) systems assume that an "on topic" collection will perform better than an "off topic" collection. That is, a collection containing many documents similar to the query will produce better results than a collection containing few documents similar to the query (for example, see Callan, Lu, & Croft 1995 or Gravano & García-Molina 1995). While this assumption sounds reasonable, it does not necessarily hold true.

The algorithms used by information retrieval systems have been designed for the express purpose of identifying relevant documents from a collection containing a wide variety of documents. It is possible that collections with a narrower focus will not improve the capabilities of a searching system. If we are searching for a document on batting averages, will it be easier to pick this document out of a collection of baseball documents, or a collection with broader focus, such as a collection of sports documents? As long as a collection has a few documents similar to the query, these documents could potentially be retrieved by the search engine, and be useful to the searcher.

Even if topically-focused collections can perform better than a non-focused collections, there may still be a problem with the accuracy of the system for selecting target collections. Sometimes documents (or collections) that appear to match the user's topic will turn out to be irrelevant. If the selection system cannot accurately choose focused collections that contain relevant documents, retrieval from a broader collection may produce better results. In this chapter, we will investigate several questions:

1. Will a search engine with a topically-focused collection perform better than a collection of documents covering a wide range of topics?
2. Will a search engine perform better with a collection that contains many documents similar to the query, or is it better for the collection to be focused on a different topic?
3. Can an automatic system select topically-focused collections with enough accuracy to provide better overall results than a centralized collection?

In order to answer these questions, we will generate collections that are focused on particular topics. The performance of search engines based on these collections will be compared to the performance of a search engine based on a more general-purpose collection. The topically-focused collections will be varied in their similarity to a given query, allowing us to study how their retrieval performance changes at different similarity levels. Document collections "in the wild" will not always be perfect matches (or even as-good-as-possible matches) to the query, so we will need to determine whether there are levels of similarity that accurately predict the answers to the questions above.

3.1 Previous Work on Topic Similarity

The typical approach taken by distributed information retrieval systems is to generate a summary of the topical content of a database, and use this summary as a surrogate for the actual database contents when calculating similarity between the query and the database. This approach allows similarity between a query and a collection to be computed in much the same manner as similarity between a query and a document in a centralized IR system.

While the technique of choosing collections that are topically similar to the query has been used in many distributed retrieval systems (Gravano, García-Molina, & Tomasic 1994; Gravano *et al.* 1997; Callan, Lu, & Croft 1995), there has been little formal tests of the theory behind topic matching. Most previous work consists of ad hoc tests that revealed interesting trends. A group of researchers led by David Hawking analyzed several methods for topic-based selection (Hawking & Thistlewaite 1999). Many of these methods outperformed a simple random selection of collections. The topic-based methods sometimes outperformed retrieval from a centralized collection, but not always. Later, these results were augmented with further indications that centralized retrieval was generally more effective than distributed retrieval using topic-based selection (Craswell, Bailey, & Hawking 2000). In a limited test (Voorhees, Gupta, & Johnson-Laird 1995), it was determined that centralized retrieval is better than distributed retrieval. While these tests are useful steps towards answering the questions posed above, they were primarily focused on selecting the best among several competing selection algorithms, rather than on determining the conditions that make topic match useful.

Ipeirotis and Gravano (2002) performed one of the only DIR experiments on actual search engines in the wild, using an algorithm for automatically categorizing the central topic of a search engine. Their experiments resulted in very low performance scores. In part this was due to the difficulty of distributed search, but another major factor was the minimal overlap between the topics represented in the search engines they used and the topics of the target queries. This indicates the need to have a large selection of search engines covering a wide variety of topics for a distributed searching system to work well.

Most previous DIR work has focused on artificially generated collections, often consisting of non-overlapping partitions of an existing collection. In the experiments that follow, we will start with artificial collections, but generate them in a way that is similar to the way actual Web collections are generated.

3.2 Creating a Collection

To test the usefulness of matching collections on topic similarity, we need to start with a collection, and determine how its performance can be evaluated. Rather than randomly selecting a collection from the Web, we will use a more controlled document set as a basis for building collections. This standard collection of Web pages, called WT2g, has many advantages for experimental use.

The WT2g Collection

WT2g is a 2-gigabyte sample of Web pages created for testing information retrieval systems. It was originally created for use with the TREC information retrieval conferences (Hawking *et al.* 1999), and is now available from the Australian research organization CSIRO[1]. It is composed of

[1]http://www.ted.cmis.csiro.au/TRECWeb/access_to_data.html

247,491 documents that were originally published across 956 Web servers. This collection is more representative of the Web than collections often used to evaluate information retrieval systems (like the widely-used TIPSTER corpus). While there are larger, more comprehensive collections available (such as the 10-gigabyte WT10g collection and the 100-gigabyte VLC2 collection), the reasonable size of WT2g allows for more flexible experimentation.

The WT2g collection was used for retrieval competition at the eighth Text REtrieval Conference (TREC-8). As a result of this use, documents in the collection have been evaluated by human judges regarding their relevance to 50 TREC topics. The TREC topics are statements of "information needs" that a user may bring to a search system. Each topic contains a description of the information need at three levels of specificity. First is a title, which contains a 2-3 word summary of the topic. This is the sort of information a user might type into a typical search engine. The description section of the topic contains 1-2 sentences with a bit more detail, such as a user might bring to a reference desk in a library. Finally, there is a narrative description, which discusses in greater detail the types of documents that would be considered relevant and non-relevant to the topic. Figure 3.1 shows topic 401, which is the first topic for which relevance scores are available with respect to WT2g documents. The topic's ID number is 401 due to the fact that topics used in TREC conferences are given sequential numbers, and new topics are created for use at every conference.

Relevance scores for TREC topics are binary, so a document cannot be considered "partially" relevant. The human judges are instructed to count a document as relevant if it meets the criteria set out in the narrative portion of the topic. The number of documents in the WT2g collection that are considered relevant to a given topic ranges from 6 to 148, with an average of 45.6. Topic 401 falls very near the average, having 45 relevant documents in the collection. Figure 3.2 contains the titles of some documents the human judges considered relevant to topic 401.

```
<top>
<num> Number: 401
<title> foreign minorities, Germany
<desc> Description:
What language and cultural differences impede the integration of foreign
minorities in Germany?
<narr> Narrative:
A relevant document will focus on the causes of the lack of integration
in a significant way; that is, the mere mention of immigration difficulties
is not relevant.  Documents that discuss immigration problems unrelated to
Germany are also not relevant.
</top>
```

Figure 3.1: Topic 401

```
WT02-B12-172 Germany-Kohl: Immigration Affects Unemployment
WT02-B12-220 The third Migration Dialogue Seminar on Integration Issues
             and Immigration Policy
WT02-B13-128 MIGRATION NEWS  Vol. 1, No. 2    March, 1994
WT09-B19-132 The Forced Migration Alert: 20 September 1996: Germany Refugees
```

Figure 3.2: Documents relevant to topic 401

Documents in the WT2g collection have a small amount of topical overlap. Some documents cover multiple topics, but most documents focus on a single topic, in a manner similar to documents typically found on the Web. Over the 50 topics used in the TREC-8 Web competition, 2149 WT2g documents are relevant to at least one topic (this is less than 1% of the entire collection). 116 documents are relevant to two or more topics, 13 are relevant to 3 or more topics, and only one document (WT02-B15-8) is relevant to 4 topics.

The Search Engine

Once we have selected a document collection, we need a way to search over it. For this task, we will use Lucene[2], an open-source toolkit for information retrieval written in Java. Lucene uses

[2]http://lucene.apache.org/java/

a simple variant of the popular vector-space model (Salton, Wong, & Yang 1975) for ranking documents relative to a query. The vector-space model treats both documents and queries as high-dimensional vectors, where every term in the dictionary corresponds to a dimension of the vector. The vectors are used to compute a score that represents the "similarity" between each document and the user's query. These similarity scores are then used to rank documents in order of their estimated relevance to the query.

Most vector-space retrieval systems, including Lucene, use a metric known as TFIDF to calculate scores for each term in the vector. TFIDF metrics estimate the amount each term contributes to the "meaning" of a document or query. TFIDF is an abbreviation for "term frequency, inverse document frequency". Under a TFIDF metric, each term in a query or document is given a score based on two factors:

Term Frequency (TF): Terms that occur frequently in a document are good indicators of the document's central topic. When a term appears many times in a document it is likely that the document is "about" that term. A term's TF score varies with its frequency within the document. In other words, higher frequencies within the document result in higher TF scores. A document about water skiing may have a high term frequency for the terms "water", and "ski", among others.

Inverse Document Frequency (IDF): Terms that occur frequently throughout the collection of documents do not assist in discriminating between documents. A term's score varies inversely with its frequency in the collection. That is, higher frequencies within the collection result in lower IDF scores. In a collection of documents about water sports, "water" may appear in nearly all of the documents, giving it a low IDF score, while "polo" may appear in only a few of the documents, resulting in a high IDF score for that term.

When the similarity is computed, these two scores are multiplied, so only terms that score highly on both will make a large contribution to the final score. In fact, for any individual query or document, most dictionary terms will have a term frequency of zero, and therefore a TFIDF score of zero.

To calculate the similarity between a document and a query, the two vectors are normalized, or scaled to have a length of one. Then, the dot product of the two vectors is taken. The dot product operation multiplies corresponding scores in the two vectors, and adds all resultant values. This means that the TFIDF scores from corresponding terms in the query and document will be multiplied, and only terms that the two vectors have in common (with non-zero values) will contribute to the similarity score.

The result is equivalent to finding the cosine of the angle between the two vectors, and is often called the "cosine similarity measure". The "closer" the two vectors are to each other, the higher the value of the similarity measure. At the extremes, this measure will have a value of 0 for vectors that have no terms in common, and a value of 1 for vectors that are equivalent.

The actual computation used by Lucene, described in (Lucene: Frequently Asked Questions Web site 2006), is slightly more complex than a standard TFIDF calculation. Lucene's similarity formula is, for a query q, a document d, and a dictionary of terms T:

$$\begin{aligned}
\text{similarity}_L(d,q) &= \sum_{t \in T} \frac{\text{TF}_L(t,q) \cdot \text{IDF}_L(t)}{\text{norm}(q)} \cdot \frac{\text{TF}_L(t,d) \cdot \text{IDF}_L(t)}{\text{norm}(d)} \cdot \text{boost}(t) \cdot \text{coord}(q,d) && (3.1)\\
\text{TF}_L(t,x) &= \sqrt{\text{frequency of } t \text{ in } x} && (3.2)\\
\text{IDF}_L(t) &= \log\left(\frac{\text{number of documents in collection}}{\text{number of documents containing } t+1}\right) + 1 && (3.3)\\
\text{norm}(x) &= \sqrt{\sum_{t \in T} (\text{TF}_L(t,x) \cdot \text{IDF}_L(t))^2} && (3.4)\\
\text{boost}(t) &= \text{user-specified boost factor for } t && (3.5)
\end{aligned}$$

$$\text{coord}(q, d) = \frac{\text{number of terms } q \text{ and } d \text{ have in common}}{|q|} \tag{3.6}$$

The square root of a term's frequency is used to compute the TF score, to keep very long documents from skewing the the results. For similar reasons, a logarithm is applied to calculate the IDF score. Lucene also allows users to specify a "boost" in the weight of terms for their query. Setting the boost parameter to a higher values is equivalent to repeating a term within the query. We will be ignoring the boost for now. Finally, Lucene applies an extra "coordination factor" to increase the score of documents that have more terms in common with the query. This ensures that documents that contain three terms of a four-term query will score higher than documents that only contain two of the query terms (regardless of how high those two terms score).

Creating a Collection From WT2g

To investigate the effects of topically-focused collections, we will begin by comparing two types of collections. The first collection, representing the contents of a large general-purpose search engine, will consist of all documents in the WT2g collection. The second collection will be a smaller collection focused on the topic described previously, topic 401.

There are many ways to create a topically-focused collection, but one of the simplest is to search a larger collection for documents that have keywords of interest. This process is similar to the method of collection creation used by some search engines, called focused crawling (Chakrabarti, van den Berg, & Dom 1999).

We will build the focused collection by searching the full WT2g collection, using the description portion of topic 401 as our search criteria. The top 4000 documents will be selected from the result list, and this will form the collection. While large centralized search engines contain billions of documents, the number 4000 was chosen as representative of the size of smaller, more topically focused search engines. This is in line with collection sizes for previous distributed retrieval systems (Yuwono & Lee 1997; Xu & Croft 1999; Powell *et al.* 2000), and is assumed to be typical of topic-specific search engines on the Web.

3.3 Evaluating Collection Performance

After placing the collection into a search engine, we need a way to measure the performance of the search system as a whole. Although there are many evaluation metrics used by the information retrieval community, most metrics are variants of three basic measures:

Precision: the percentage of retrieved documents that are relevant

Recall: the percentage of relevant documents that are retrieved

P@N: examining only the first N documents retrieved, the percentage that are relevant.

The most common method of examining search system performance is to graph precision as a function of recall. A graph is created by gathering documents until a particular recall level is met, measuring precision at that point, and repeating for multiple recall levels. A good retrieval algorithm will place most of the relevant documents near the beginning of the retrieved set, and precision will typically decrease as higher levels of recall are sought. While it is possible for precision to increase from one recall level to the next, most precision-recall graphs decrease monotonically (similar to the one shown in Figure 3.3).

If the precision value is noted after each relevant document is seen (rather than at specific levels of recall), these values can be averaged to give an evaluation measure called **average precision**.

The average precision measure indicates the area underneath the precision-recall curve, and is a useful summary of the overall effectiveness of a search system for a given query.

There are many other evaluation measures, but most of them are simply variants of those discussed above. For example, **weighted precision** is similar to average precision, but adds extra weight to the first few precision scores, boosting the overall score of search systems that have high precision early in the retrieved set.

These evaluations will be somewhat different from previous DIR evaluations. Evaluations on DIR systems (French *et al.* 1999; Xu & Croft 1999) have previously focused on recall metrics (proportion of relevant documents retrieved). On the Web, it is generally not possible to calculate recall, because the number of relevant documents in the universe is not known. For many queries, the Web contains millions of relevant documents, which can not all be read by a single searcher, further decreasing the utility of recall as a performance measure. Instead, we will focus on two precision-based measures, P@20 and average precision. P@20 (precision at 20 documents) is particularly important for Web search, because users often look at only the first page of results from a search engine (Spink & Jansen 2004). A system that boosts initial precision at the expense of low precision later in the result set is acceptable. In fact, early precision is even more important for DIR, because a typical DIR system will take the first few results from each individual engine and combine them to present the final result list. Average precision, on the other hand, and is a good measure of the general usefulness of an IR system. Both P@20 and average precision have a history of use for comparing Web search engines (for example, in Hawking *et al.* 1999).

Lucene's Overall Performance

To illustrate the use of these measures, and verify the basic functionality of the search system, Lucene was applied to the Web retrieval task from TREC-8, retrieving documents from the WT2g

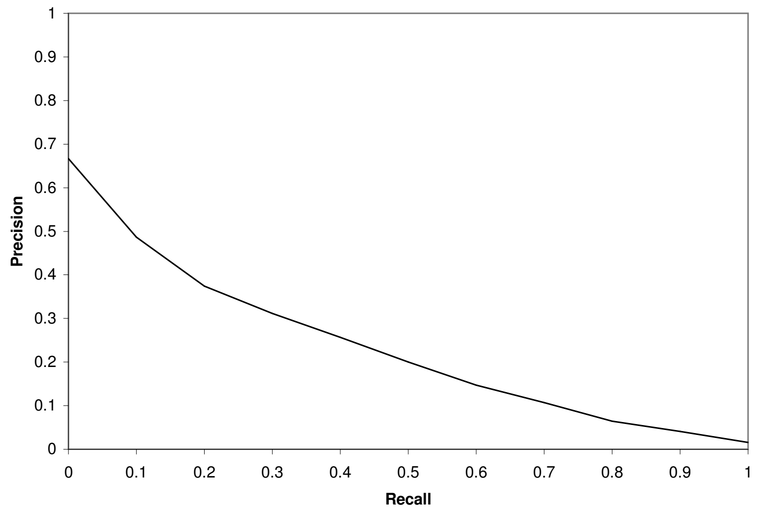

Figure 3.3: Lucene's average performance on the TREC-8 Web topics.

collection for each of the 50 target topics.

For all of the experiments that follow, the search engine itself was held constant, while the contents of the document collection were varied. Lucene was set up with a typical configuration. A basic set of "stopwords", common words with little informational content (like "the", "and", and "begin") were removed from the documents. No other pre-processing was performed. Queries were automatically generated from the "Description" portion of the topic, weighting all terms equally. For example, the query generated for topic 401 was: "what language cultural differences impede integration foreign minorities germany".

Figure 3.3 shows Lucene's average precision-recall curve for the 50 topics when retrieving documents from the full WT2g collection. This curve was created by averaging the precision scores for the 50 topics at each recall level. An "overall" average precision score was calculated, averaging the precision scores for all recall levels over all 50 topics. This score was 0.2175, which would have placed in 11th position against the 17 groups that participated in the TREC-8 Web competition (see

Hawking *et al.* 1999). Although this performance was not exceptional, and the state of the art has obviously advanced in the years since the TREC-8 competition was held, our primary concern is not that Lucene outperform the other systems, but rather to affirm that it is not significantly lacking. We are interested in comparing the difficulty of queries across various collections, not in comparing the details of the search engine being used. Lucene's search algorithm is typical of the engines that underly a large number of topic-specific search services present on the Web. In fact, many of these search services are actually based on Lucene.

Comparing the Topical Collection to the Full Collection

Now that our evaluation metrics have been established, we can compare retrieval results from a topically-focused collection with results from a centralized collection. The topically-focused collection will be the collection based on topic 401, as described in Section 3.2. The centralized collection will be the full set of WT2g documents. Both collections will be evaluated using topic 401 as the query. While this is somewhat unrealistic (collections are seldom custom-built for a single query), it can be thought of as a "best case" for a topical search engine. What better place to look for information on topic 401 than a search engine that was custom built with this topic in mind?

For simplicity, I will be referring to a search system based on a collection simply by the name of the collection. The search engine (Lucene), and all other processing will remain the same, using the configuration described in Section 3.3. Therefore, the search system based on the full WT2g set will be referred to simply as "the full collection", while the search system based on topic 401 will be called collection t401-t1 (the t1 designation is added to distinguish this collection from variants that will be described later).

There are several differences between the collections that may affect retrieval performance. First, there is a great difference in the size of the collections. The full collection contains nearly

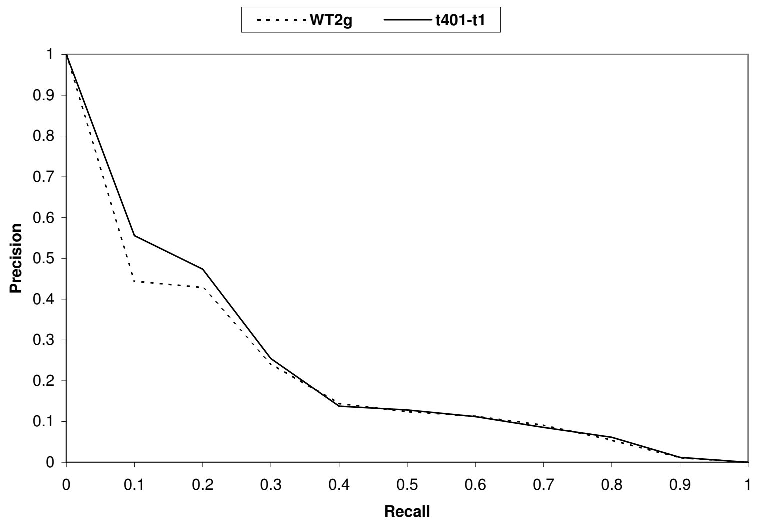

Figure 3.4: A comparison of precision-recall curves using a search engine based on topic 401 and a search engine based on the full WT2g set.

250,000 documents, while the topical collection contains only 4000 documents. One effect of this size difference is that the topical collection does not contain as many “distractor” documents as the full collection. Intuitively, this should improve the performance of the topical collection, since it only has to identify relevant documents from a small pool of documents, rather than the much larger pool of documents in the full collection. Another difference between the collections is in the number of documents that are actually considered relevant. While the full WT2g collection contains 45 documents relevant to topic 401, the topical collection only contains 42 of these documents. This difference does not immediately suggest a difference in the performance of the two engines. It does mean that the number of documents needed to achieve 100% recall is less for the topical collection than for the full collection, but this does not matter much, since we are primarily concerned with measures of precision.

Figure 3.4 compares the precision-recall curves produced by the full collection and the t401-t1 collection. The two curves are very similar, particularly at high levels of recall. This is to be

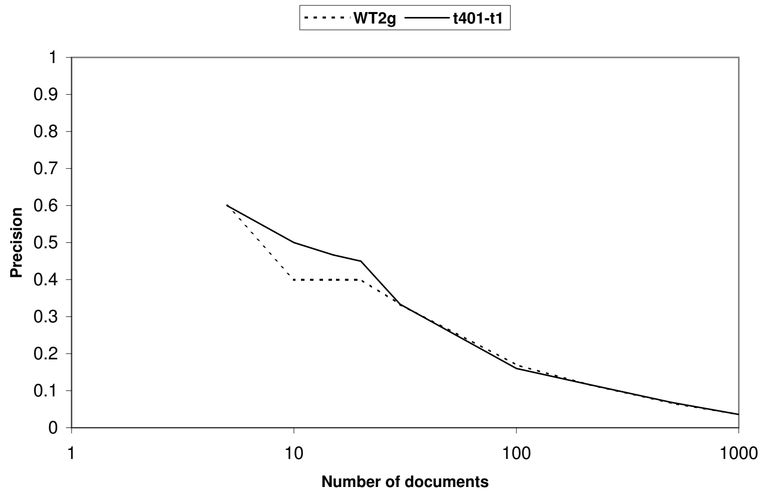

Figure 3.5: A comparison of P@ curves using a search engine based on topic 401 and a search engine based on the full WT2g set.

expected, since the two collections contain nearly the same sets of target documents. It is promising to see that for low recall levels, the topical collection has higher precision scores.

However, this is not really a fair comparison. Precision-recall curves are used extensively in the information retrieval literature when comparing algorithms and holding the collections constant. In our case, where the collection itself is being varied, comparing recall scores can distort the results. Since there is a difference in the total number of relevant documents contained in the two collections, the number of relevant documents required to reach a given level of recall is no longer constant. We will dispense with this problem by ignoring recall-based measures. As discussed previously, recall is not a particularly useful measure when dealing with Web search, so we will focus on measures of precision.

The graph shown in Figure 3.5 more accurately displays the difference between the two collections, comparing P@ values for the first 1000 documents retrieved from each collection. As in the previous comparison, the t401-t1 collection outperforms the full collection, particularly when small

Collection	P@20 (Topic 401)	Average P (Topic 401)
Full WT2g	0.40	0.2094
t401-t1	0.45	0.2267

Table 3.1: Comparison of one-dimensional precision measures.

numbers of documents are retrieved, the case most typical for Web searching.

In the next section, when we start comparing many collections, it will be cumbersome to make these comparisons between P@ curves, so we will focus on one-dimensional measures of precision. Scores for the one-dimensional measures on these two collections are shown in Table 3.1.

In this case, the topical collection slightly outperforms the full collection, but this will not always be the case, as we shall soon see.

3.4 Varying the Topical Focus of a Collection

The comparison above indicates that small, topically-focused collections can outperform larger, more general collections. But this comparison was somewhat contrived, using a collection that was custom built for the topic in mind. In fact, the query used for testing was exactly the same as the query used to generate the collection. Collections "in the wild" tend not to be built like this. It is extremely rare for a collection to directly correspond with a specific user need. It is easy to understand this–just look at topic 401 (in Figure 3.1). It is unlikely that any real search systems on the Web that cater to this topic. However, there are many Web search sites that focus on information *related* to this topic, such as search engines specific to Germany, search engines dealing with immigration, etc.

Creating Collections With Varied Focus

To investigate the effects of collections that are not specifically created for the topic in question, we will look at collections that have varying degrees of similarity to topic 401. These collections were created from the full WT2g collection in the same way the t401-t1 collection was created, by sending a query to the "full collection" search engine and selecting the first 4000 documents returned.

We will look at collections representing four levels of similarity. The first collection is the t401-t1 collection we have been using. Two additional collections were generated by making manual modifications to the topic 401 query. These modifications consisted of deleting terms or substituting terms with similar terms, often resulting in a collection that covered a broader array of topics while remaining the same size. The fourth collection is intended to be as off-topic as possible. This collection was generated from a query based on a completely separate topic (topic 413, regarding steel production). Details about the specific queries used can be found in Appendix A.

The following designations will be given to the collections based on topic 401:

- t401-t1 = documents retrieved using topic 401's description as a query
- t401-t2 = documents retrieved by modifying the topic description slightly
- t401-t3 = documents retrieved by modifying the topic description greatly
- t401-t4 = documents retrieved using the description of a completely different topic

Initial Results

Figure 3.6 shows a comparison of the four collections based on topic 401, using both of the one-dimensional precision measures. Keep in mind that each pair of columns on this graph is an

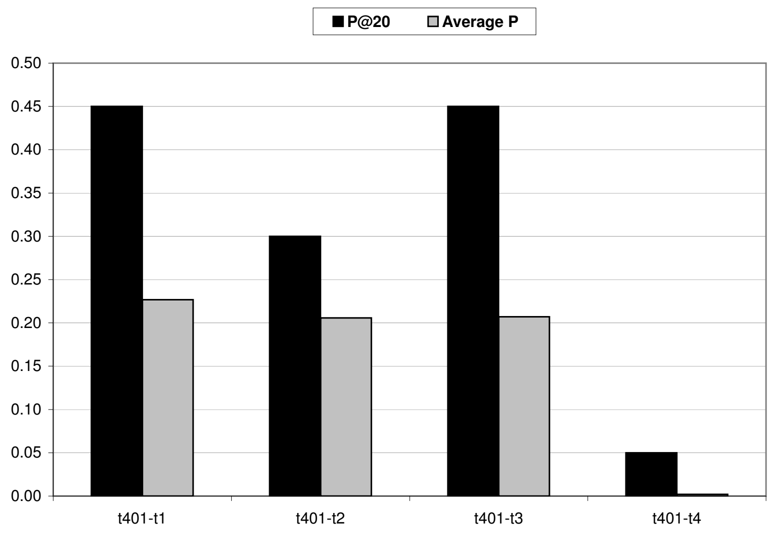

Figure 3.6: Comparison of P@20 and Average Precision scores for the collections based on topic 401.

attempt to summarize a precision graph. For example, the leftmost pair of bars is a summary of the solid line from Figure 3.5.

For both measures, there is a definite decrease between the t401-t1 collection and the t401-t4 collection. However, the scores for the intermediate collections are a bit more confusing. The average precision measure drops off gradually as the collections are moved more off-topic, but the P@20 measure has an unexpected high value for the t401-t3 collection, illustrating the greater variability of the P@20 measure.

3.5 Calculating Similarity Between Queries and Collections

The results above are consistent with our intuition. As similarity between the query and the collection decreases, performance decreases. We will now try to establish a method for predicting performance based on the topic of a collection. To do this, we must establish a method for quantifying the amount of similarity between a query and collection.

One of the most successful algorithms for calculating this type of similarity is CORI, first expressed in (Callan, Lu, & Croft 1995), and later refined (Callan 2000; French *et al.* 1999). In comparisons of the most popular DIR algorithms, CORI came out on top (French *et al.* 1999; Powell *et al.* 2000).

CORI is a modification of TFIDF to work on the collection level instead of the document level, treating each collection as a "superdocument". CORI's core formulas are, for a term t, collection c, and set of collections C:

$$\begin{aligned}
\mathrm{TF}_{CORI}(t,c) &= \frac{\text{frequency of } t \text{ in collection } c}{\text{frequency of } t \text{ in collection } c + 50 + 150 \cdot \mathrm{cw}(c)/\mathrm{avgcw}} && (3.7)\\
\mathrm{IDF}_{CORI}(t) &= \log\left(\frac{|C| + 0.5}{\text{number of collections containing } t}\right) \cdot \frac{1}{\log(|C|+1)} && (3.8)\\
\mathrm{avgcw} &= \sum_{c\in C} \frac{\mathrm{cw(c)}}{|C|} && (3.9)\\
\mathrm{cw}(c) &= \text{total number of terms in collection } c && (3.10)
\end{aligned}$$

In much of the literature, CORI's algorithm is described as TFICF ("term frequency, inverse *collection* frequency"), but we will keep the term TFIDF for simplicity of comparison. By comparing with formulas 3.2-3.6 in section 3.2, some obvious similarities can be seen. Although there are differences in the exact method for calculating weights, the essential properties of these formulas are the same. The TF values rise with increasing frequency of the target term in the item (whether the item is a document or a collection), and the IDF values rise with decreasing frequency of the term throughout the set (whether the set is a collection of documents or a set of collections). All of these formulas make use of functions that reduce the effect of very large values (e.g., logarithms, square roots, asymptotes) to place large documents or collections on a relatively equal footing with smaller ones.

One major difference between the equations here and those in section 3.2 is that the CORI equations were designed to work with a probabilistic retrieval system, rather than a vector space retrieval system. CORI uses these TF and IDF values to compute a "belief" that a given query will be satisfied by the contents of a collection. Although the probabilistic model allows for complex combinations of queries, the belief calculations for a standard "bag of words" query are roughly equivalent to the similarity calculations in the vector space model. In this case, CORI's belief that a query q will be satisfied by a collection c is:

$$\mathrm{p}(q|c) \quad = \quad \sum_{t \in q} \frac{0.4 + 0.6 \cdot \mathrm{TF}_{CORI}(t, c) \cdot \mathrm{IDF}_{CORI}(t)}{|q|} \tag{3.11}$$

It seems that the CORI equations are perfectly suited for our purpose of estimating the "similarity" between a query and a collection. However, we run into a small problem. The TFIDF metric was designed for distinguishing between documents with high and low similarity *within a collection*. CORI, which is meant for calculating similarity between a query and a collection, therefore needs to distinguish a collection *from a set of collections*. Thus far, we have been considering collections in isolation, and for the time being, it is necessary to continue looking at them in isolation. It is difficult to artificially generate a set of collections that would be similar to the set of collections used by a fielded system. We cannot accurately predict the sorts of collections that will exist on the Web in the future, nor the collections that would be sought and discovered by a distributed searching system working on the behalf of a particular user. This is likely one of the reasons that CORI and other distributed information retrieval algorithms have thus far been used almost exclusively on simple test collections.

In the absence of other collections, weighting schemes like CORI that rely on statistics of the entire set of collections must be modified before use. While the IDF factor is very helpful in pulling documents out of a particular set, it is not particularly useful when the background set is unknown.

The Nature of IDF

When dealing with individual documents, the IDF portion of the TFIDF calculation serves mostly to balance the effect of “noise words” in the query. It can be viewed as adjusting the weights of the query terms, giving higher weights to words that have better discriminatory power. For example, in a collection of documents on water sports, we may have a document on the topic of “water polo”, and a document on “water skiing”. The frequency of the terms might be:

Document A	water=100
	skiing=50
	polo=0
Document B	water=20
	skiing=0
	polo=20

If we only take TF into account, a query for “water polo” would rate document A higher, because the occurrence of the term “water” is much higher, even though the document B is obviously more likely to contain references to the sport of water polo. The IDF factor identifies words like “polo” that tend to be descriptive of documents on a particular subject, and weights these words higher than words like “water” that will match most of the documents in the collection.

However, the IDF score is less important when comparing collections. Collections contain a variety of documents, not all of which will have exactly the same focus, and not all of which will use the same terminology to describe a particular topic. This makes collections less sensitive to minor changes in vocabulary usage than individual documents. The term frequencies of the two documents described above would be averaged in the collection’s TF score, making the score more indicative of usage in an "average" document on that topic.

Similarity Estimation

Because the effect of IDF is small on the collection level, and because there is no good way to estimate the IDF values for a single collection in the absence of other collections, we will initially have to modify CORI. In the modified version, the IDF calculation is ignored. That is, IDF(t) is always set to 1. The TF score (equation 3.7) also has an adjustment for collections that have a larger or smaller number of terms than the average. Since this adjustment (cw(c)/avgcw) cannot be made in the absence of other collections, its value is also set to 1. With these adjustments, CORI's similarity calculation simplifies to:

$$\text{similarity}_{CORI}(c,q) = \sum_{t \in q} \frac{0.4 + 0.6 \cdot \text{TF}_{ModCORI}(t,c)}{|q|} \tag{3.12}$$

$$\text{TF}_{ModCORI}(t,c) = \frac{\text{frequency of } t \text{ in collection } c}{\text{frequency of } t \text{ in collection } c + 200} \tag{3.13}$$

Since we are forced to modify the CORI formula, it is possible that another method would give better results. We will compare the CORI scores with two alternative similarity calculations. The first alternative method is a centroid, which finds a term vector representing the "average" document in a collection. Instead of using the collection-wide TF measure, this calculation uses the document-based TF and IDF values, as indexed by Lucene (equations 3.2 and 3.3):

$$\text{similarity}_{Cent}(c,q) = \sum_{t \in q} \frac{\text{TF}_{Cent}(t,c) \cdot \text{IDF}_{Cent}(t,c)}{\sqrt{\sum_{t \in T} (\text{TF}_{Cent}(t,c) \cdot \text{IDF}_{Cent}(t,c))^2}} \cdot \frac{\text{TF}_L(t,q)}{\text{norm}_{Cent}(q)} \tag{3.14}$$

$$\text{TF}_{Cent}(t,c) = \text{average TF}_L(t,d) \text{ over all documents in collection } c \tag{3.15}$$

$$\text{IDF}_{Cent}(t,c) = \text{IDF}_L(t) \text{in collection } c \tag{3.16}$$

$$\text{norm}_{Cent}(q) = \sqrt{\sum_{t \in q} \text{TF}_L(t,q)} \tag{3.17}$$

The other alternative is based on document frequency. That is, the score of a term t is the number of documents in the collection that contain t. This type of score is very crude, because it does not include any scaling to correct for large documents or large collections. A large document may contain many terms, perhaps matching every term in the query even though it is not a good match for the query. However, this measure has the advantage that a collection will not match the query based on the high score of a single document. Multiple documents in the collection must contain the term in order to generate a high similarity score.

$$\begin{aligned}
\text{similarity}_{DF}(c,q) &= \sum_{t \in q} \frac{\text{df}(t,c)}{\sqrt{\sum_{t \in T} (\text{df}(t,c))^2}} \cdot \frac{\text{TF}_L(t,q)}{\text{norm}_{DF}(q)} && (3.18)\\
\text{df}(t,c) &= \text{number of documents in collection } c \text{ containing } t && (3.19)\\
\text{norm}_{DF}(q) &= \sqrt{\sum_{t \in q} \text{TF}_L(t,q)} && (3.20)
\end{aligned}$$

None of these approaches will provide the “optimal” score for each collection, but we are more interested in obtaining reasonable estimates to identify trends. In Chapter 7, when we are actually choosing collections from a specific set of collections, it will be possible to use the full CORI score, but we will see that that is not necessary.

There is little difference between the qualitative results of the three measures (see Figure 3.7). The document frequency and centroid measures result in scores between 0 and 0.1. CORI, in part due to the added 0.4 in the similarity calculation, resulted in scores that were much closer to 1. There was a greater difference between the highest and lowest CORI values (.1161) than with the

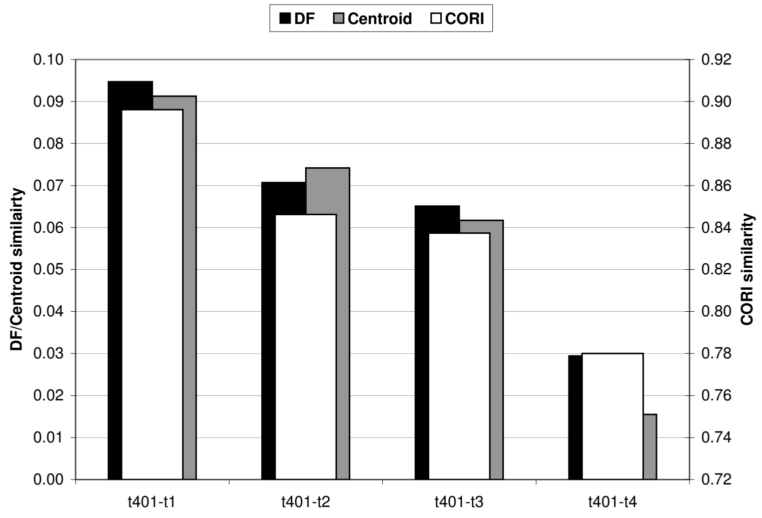

Figure 3.7: A comparison of the three candidate similarity measures.

other measures (DF .0653, centroid .0758), but all three displayed the same trend of decreasing similarity values as the collection was moved farther off topic. The same comparison was performed on two other topics, with similar results. (For a description of the other topics, see section 3.7.) Since all three measures display the same trends, we will only use the CORI measure in future calculations.

Now that we have both (a) a method for evaluating a collection's performance on a query and (b) a method for calculating similarity between a query and collection, we can examine the accuracy of the similarity measure in predicting the collection's performance. Figure 3.8 combines the data from Figures 3.6 and 3.7. There is a definite relationship between the similarity scores and the precision scores ($r = 0.85$).

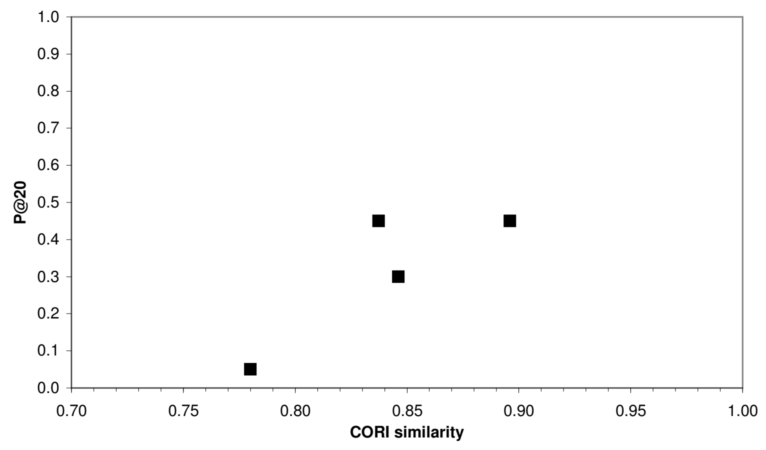

Figure 3.8: Comparison of the similarity scores to measured collection performance.

3.6 The Robustness of Similarity Scores

It appears that CORI similarity is a good indicator of how well a collection will perform with respect to a given query. Unfortunately, this relationship is not guaranteed. It is possible that we might encounter a collection that looks like it matches the query, but does not contain any relevant documents. How well can similarity scores detect these types of collections?

The collections described to this point can be considered the "typical" case. These collections are representative of the type of collection created by an automatic system that crawls the Web looking for certain keywords (Chakrabarti, van den Berg, & Dom 1999; Qin, Zhou, & Chau 2004). They are also assumed to be representative of databases created from a single source of homogeneous documents, such as a company or a department within a university.

It is possible to manipulate the contents of a collection so that its contents (relative to the query) are changed greatly without having much affect on the similarity score. To illustrate this, two new sets of collections were generated. A set of collections representing the "worst possible case" was

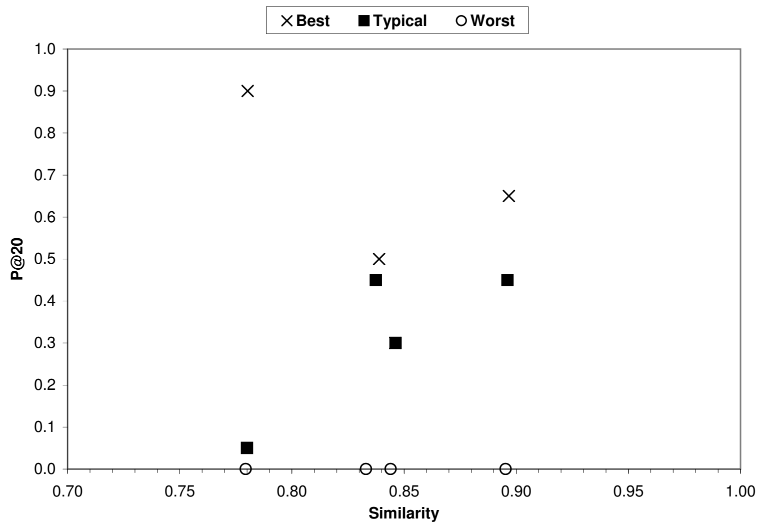

Figure 3.9: Effect of changing similarity on topic 401.

created by removing all relevant documents from the "typical" databases, and replacing them with non-relevant documents. For example, all documents marked relevant that appeared in the t401-t1 collection were replaced to create the t401-t1 worst case collection. A third set of collections, the "best possible case" collections, was created by substituting documents relevant to the target query for non-relevant documents until all relevant documents were included in the collection.

Figure 3.9 compares the similarity scores and resultant precision scores for all 12 collections based on topic 401. The worst case scores are disappointing. While the similarity values have changed little from the typical case, the precision scores have all dropped to zero, because the collection no longer contains relevant documents. An automatic system based on comparing similarity scores would not be able to tell worst case collections from typical collections.

The best case scores are along the top of the graph, always performing as well as or better than the typical collections. (Note that one of the x's on the graph is obscured by a square, as both the best case collection and typical collection have a precision of 0.3 in this case.) The best case scores actually fall off a bit as similarity increases. The usefulness of a "best possible" collection decreases as it becomes more similar to the query, due to the fact that when all documents in the collection are similar, it is more difficult to pick out the relevant documents. When the relevant documents are sufficiently different from the other documents in the collection (e.g., in the t401-t4 case at the far left) it is much easier to pick them out.

Even though the worst possible case performance is disappointing, it is an extremely unlikely case. Just as it is difficult for our similarity score to tell the difference between a collection with relevant documents and a collection without relevant documents, an automatic process for building collections would have a difficult time excluding relevant documents, even if it were to attempt this task. Nonetheless, we will continue to examine the best case and worst case performance as we investigate other possible areas of variation.

3.7 Varying Topic Difficulty

It would be foolish to assume that the results discussed previously will hold for any topic. "Easy" topics may be less sensitive to changes in the underlying collection, because a search algorithm will always be able to discriminate relevant from non-relevant documents. Likewise, "difficult" topics may produce poor results regardless of the collection being searched.

It is not possible to determine in advance whether a particular topic will be easy or difficult. The perceived difficulty of a query is highly dependent on the number of relevant documents in the collection, the match between language used in a query and language used in relevant documents, and the use of query terms in non-relevant documents. The TREC community briefly addressed this issue (Buckley & Walz 1999), but did not come to any meaningful conclusions. However, since the WT2g collection includes relevance scores, we have a handy measurement for the "difficulty" of a query: the performance of our search system when searching over the entire WT2g collection.

Selecting Target Topics

To investigate the effect of varying topic difficulty on the predictive power of the similarity calculation, three topics were chosen from the TREC-8 Web task. These topics were chosen based on the precision scores that resulted from running the queries through the base Lucene configuration, searching over the full WT2g set (see Section 3.3). The topics used were:

1. Easy topic (#441) - This topic requested information on the prevention and treatment of Lyme Disease. The system was able to identify relevant documents with relative ease due to the word "Lyme", which rarely appears in non-relevant documents. However, since the desired result set was a subset of the information available on Lyme disease (prevention and treatment only), this topic proved to be a bit more difficult when using queries based on the topic's title only.

2. Medium difficulty topic (#401) - This topic, described previously, requested information on the integration of minority groups into Germany's culture. As with the easy topic, the title did not fully specify the intended documents, and title-only queries proved more difficult than queries derived from the description.

3. Difficult topic (#437) - This topic requested information regarding the effects of gas and electric deregulation. It is difficult to pinpoint exactly what made this topic difficult, but it is likely that the individual terms in the query were common enough to make it difficult to single out documents relevant to this topic.

Figure 3.10 shows the precision-recall curves for the three topics. This graph is presented to provide a comparison with earlier precision-recall graphs, even though it is somewhat misleading.

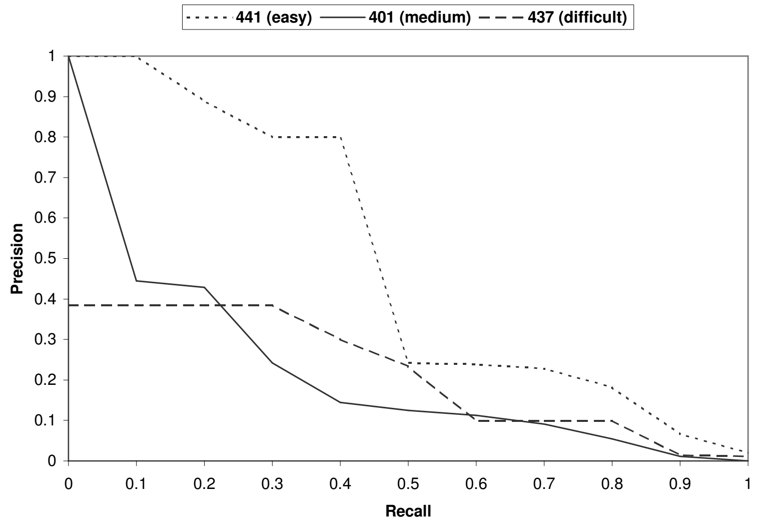

Figure 3.10: Lucene's precision-recall curve for each of the test topics using the centralized collection.

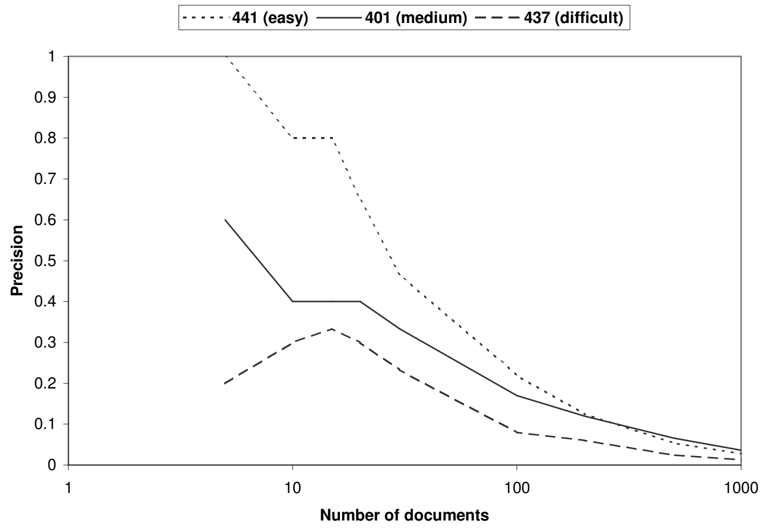

Figure 3.11: Lucene's "P@" performance on the test topics using the centralized collection.

Topic	**TREC ID#**	**# Relevant**	**P@20**	**Average P**
Easy	441	29	0.65	0.5055
Medium	401	45	0.40	0.2094
Difficult	437	14	0.30	0.1878

Table 3.2: Summary of the three target topics.

The three curves on this graph are not directly comparable, since the three topics had differing numbers of relevant documents. For example, the difficult topic had 14 relevant documents, so the search engine would only need to find 5 documents to reach the 30% recall level, while the medium topic, with 45 relevant documents, required finding 14 documents to reach the 30% level.

Figure 3.11 shows the results from Lucene according to the P@ measure, which gives a better comparison, because it mirrors the way documents are typically viewed by a user. Remember that our primary purpose is to evaluate the effectiveness of the retrieval system from a user's perspective, and users hardly ever look at more than the first 20 documents in a result set. Table 3.2 summarizes the three topics and the effectiveness with which Lucene was able to retrieve them from the full WT2g collection.

Results of Changing Topic Difficulty

To complement the initial 12 collections shown in Figure 3.9, an additional 24 collections were generated, consisting of typical, worst-case, and best-case collections for both the easy topic (441) and the difficult topic (437). These collections were then evaluated by comparing the CORI similarity scores with the P@20 precision values.

Figure 3.12 shows the scores for collections built from the easy topic. Comparison with the medium-difficulty topic (Figure 3.9) reveals that both the similarity scores and precision scores are higher for the easy topic. Figure 3.13 shows the same comparison for the difficult topic. In this case,

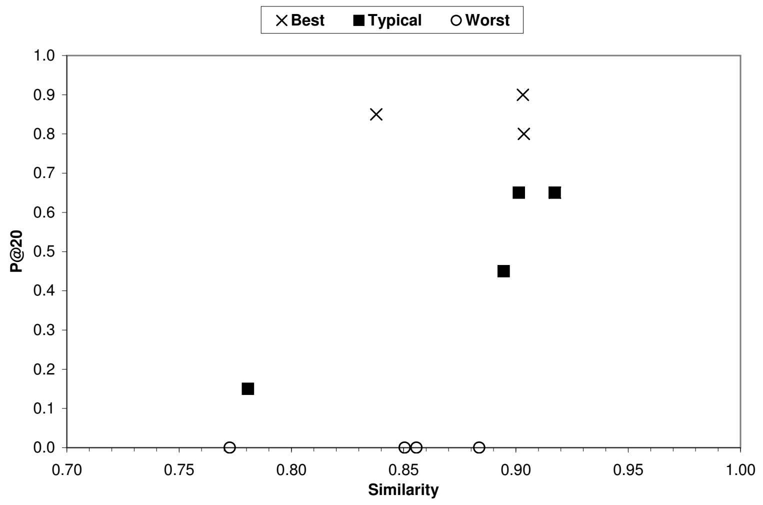

Figure 3.12: Effect of changing similarity on topic 441 (easy).

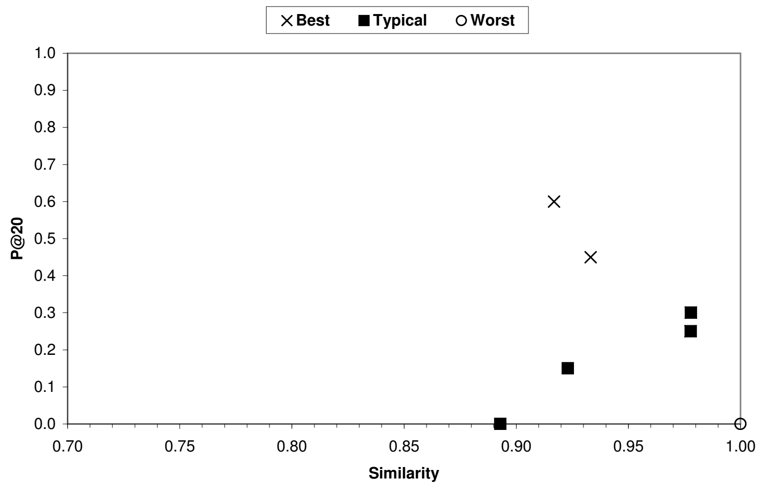

Figure 3.13: Effect of changing similarity on topic 437 (difficult).

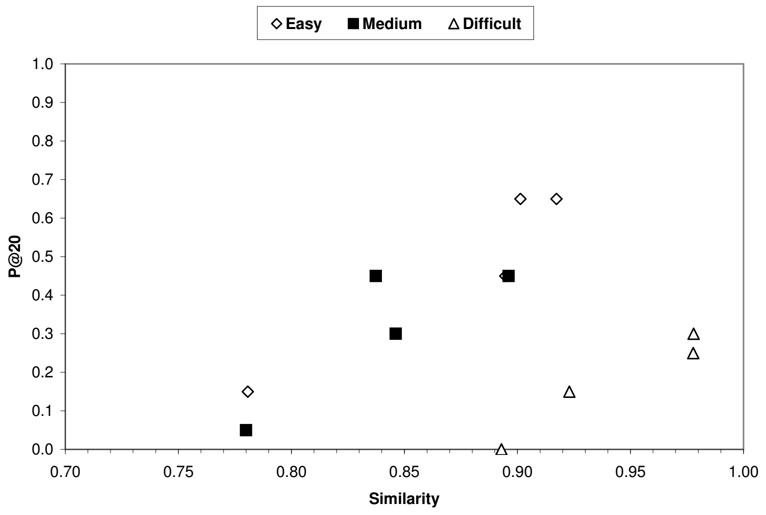

Figure 3.14: Summary of changing similarity.

the similarity scores are higher than collections from both of the other topics. Even though some of the documents in these collections had high similarity, very few of them were actually relevant. The precision scores are the lowest so far, indicating the true difficulty of this topic.

The precision scores, as expected, increased or decreased based on the difficulty of the topic. The lack of a clear trend in the similarity scores is initially surprising, but it makes sense, since the process of building a collection chooses the 4000 documents most similar to the seed query, regardless of each document's absolute similarity level.

The collections built from the easy topic and the difficult topic display a bit more variation in similarity scores between best-case, typical-case, and worst-case collections than collections built from the medium-difficulty topic. The data points for a given level of similarity (t1, t2, etc., as described in Section 3.4) do not always end up in vertical columns. However, the overall trend is the same as before. The best-case collections always perform at an equal level as the typical collections, and often better. The worst-case collections always have a P@20 value of 0.

Figure 3.14 compares the precision scores obtained from all three topics. For clarity, the best-case and worst-case scores are not shown, leaving only the results of the 12 typical-case collections (four similarity levels for each topic). Looking at these graphs, it is immediately apparent that there is a relationship between the database's similarity to the target topic and its precision score. When the typical collections of all three topics are considered together, there is a positive correlation (r=0.64), even though the three topics are not directly comparable. When each of the three topics is considered on its own, the correlation is much stronger (easy r=0.92, medium r=0.88, difficult r=0.83).

3.8 Conclusions

Now it is time to return to the questions posed at the beginning of this chapter:

Will a search engine with a topically-focused collection perform better than a search engine containing documents with a wide range of topics? We have limited data, but the answer to this is currently no. Table 3.3 summarizes the P@20 values for our target topics using the Full WT2g collection and the t1 (most similar) collection created for each topic. Even though the t1 collections were created specifically for the topic in question, retrieval performance was only improved in one case. However, when half of the documents are removed from the centralized collection (WT2g-half), the topical collection outperforms the centralized collection. This is an important fact, because topical search engines will often contain many relevant documents that are not available to general-purpose search engines.

Will a search engine perform better with a collection that contains many documents similar to the query, or is it better for the collection to be focused on a different topic? It is better for the collection to be similar to the query. There is a definite correlation between similarity scores and

Collection	**Topic 441**	**Topic 401**	**Topic 437**
Full WT2g	.65	.40	.30
Topical	.65	.45	.25
WT2g-half	.65	.30	.05

Table 3.3: Summary of P@20 values for the topical and centralized collections.

retrieval performance. Even though it is possible to artificially construct collections which are off-topic, yet perform well (the best-case t4 collections), this is nearly impossible to accomplish without prior knowledge of the documents that are relevant to the query.

Can an automatic system select topically-focused collections with enough accuracy to provide better overall results than a centralized collection? If the centralized collection contains a superset of the documents in the topically-focused collection, this will be very difficult, because even when a collection is specifically built for the query in question, the centralized collection often performs better than the topically-focused collection. To perform better, a topically-focused collection must not only be centered on the correct topic, it must be highly focused on that topic. Even with topic 401, only the t1 collection (most on-topic) outperformed the full WT2g collection. As noted above, though, topically-focused collections can be a good source of relevant documents that are not available through centralized collections, which could make an automatic selection system more useful.

One possible explanation for the the relatively poor performance of the topical collections is their small size. The full WT2g collection is much larger than the topical collections. It is possible that the topical collections did not include enough relevant documents to allow higher performance. When relevant documents were explicitly added (the best-case collections), retrieval performance always increased. In the next chapter, we will manipulate the size of collections, investigating whether the presence of a larger selection of documents (both relevant and non-relevant) improves performance.

4

The Effects of Collection Size

The previous chapter suggests that choosing collections based on topic alone will not provide a significant improvement in retrieval performance. A large general-purpose collection almost always outperformed a smaller collection that was focused on the target topic.

It is possible that the negative results of the previous chapter were solely due to the small size of the focused collections. Even though 4,000 documents is typical of the collections used by small search engines, it may not be the optimal size for distributed retrieval. The WT2g collection that was used to represent a centralized collection contains "only" a quarter of a million documents. This collection could be considered "small" when compared to the billions of documents indexed by large general-purpose search engines on the Web today.

A moderately-sized collection may provide a "happy medium" between the small collections used in the previous chapter and the full WT2g collection. The larger a collection is, the more likely it is to contain relevant documents, but as a collection grows very large, it is also more likely to contain "spurious matches", documents that match the query by chance combinations of words rather than actual topical similarity.

In this chapter, we will investigate how variations in collection size affect retrieval performance.

These experiments will follow the same model as the experiments on topical focus, using a standard searching system and standard set of documents, varying only the size of the collection.

4.1 Previous Work on Collection Size

There has been very little work on the effect that collection size has on collection selection. Most notably, French and Powell (1999; 2000) found that distributed information retrieval (DIR) algorithms tend to choose larger collections more often than smaller collections. Even so, selection algorithms that are based on size alone do not perform as well as DIR algorithms that take other factors (such as topic) into account.

Collection size is often a factor when calculating similarity between a query and a collection. Algorithms like CORI include factors that are affected by collection size. For example, the raw term frequency score "frequency of t in collection c" will be higher if there are more documents in the collection. Algorithms like this typically include compensating factors that dampen the effects of larger collections (see equation 3.7). Frequently, the size of a collection is employed to adjust the weights of individual terms rather than to modify the similarity score as a whole.

4.2 The Effect of Size on Essential Properties

In the previous chapter, it was argued that IDF scores have less impact when dealing with collections than they do when dealing with individual documents. When the size of a collection is increased, IDF scores become even less important. To illustrate this, consider a set of trivial collections, where each collection holds a single document. Using CORI's IDF formula (equation 3.8, repeated below), these collections will have IDF scores very similar to the IDF scores of individual documents in a large collection.

$$\mathrm{IDF}_{CORI}(t) \quad = \quad \log\left(\frac{|C| + 0.5}{\text{number of collections containing } t}\right) \cdot \frac{1}{\log(|C| + 1)} \tag{4.1}$$

Now, consider a set of collections where each collection holds two documents. The total number of collections remains the same, so the only change in equation 3.8 is that the “number of collections containing t” increases for certain terms, slightly lowering the IDF values for these terms. As the number of documents per collection increases, the number of terms included in each collection will increase. At the extreme, each collection in the set will hold every document in the world. This type of collection would contain every term in the dictionary, making all of the IDF values identical. As collection size increases, it is increasingly likely that the collection will contain every term *somewhere*, making the IDF score even less relevant than in smaller collections.

As collections become larger, they naturally become more general. For any given topic, there are a limited number of documents in the world focused on the topic. If the process that creates a collection is very restrictive, it will eventually run out of documents to include in the collection, because it can no longer find novel documents that meet its requirements. On the other hand, if the process that creates the collection is more lax in its selection, it will continually accept documents that are only tangentially (or worse yet, superficially) related to the collection’s target topic. As more documents are gathered that are not strictly focused on the target topic, the overall contents of the collection become more general.

4.3 Performance of Smaller Collections

The collections used by search engines on the Web can range in size from just a few documents to billions of documents (e.g., Google). DIR systems typically contain several thousand documents in each collection (e.g., Yuwono & Lee 1997; Xu & Croft 1999; Powell *et al.* 2000). The collections created in Chapter 3 were based on the typical size of 4000 documents. To investigate the effect of collection size on similarity scores and retrieval performance, we will compare the typical-size collections to collections both an order of magnitude larger and smaller. That is, we will compare collections containing 400, 4000, and 40,000 documents.

We will use the same TREC topics described in Chapter 3. For each of these three topics, we have been using 12 collections with varying topical focus and number of relevant documents. Now, we will add a parallel set of "small size" collections (each containing 400 documents). There will be one "small size" collection for every collection that exists so far. That is, 3 topics $\times$ 4 similarity levels $\times$ 3 relevance levels = 36 new collections.

As with the typical-size collections, the smaller collections are given names based on the properties used to create them. The only difference in the names is that the penultimate character is replaced. Instead of letter t (for "typical"), it will be a letter s (for "small"). The small collections based on topic 401 are:

- t401-s1 = documents retrieved using topic 401's description as a query
- t401-s2 = documents retrieved by modifying the topic description slightly
- t401-s3 = documents retrieved by modifying the topic description greatly
- t401-s4 = documents retrieved using the description of a completely different topic

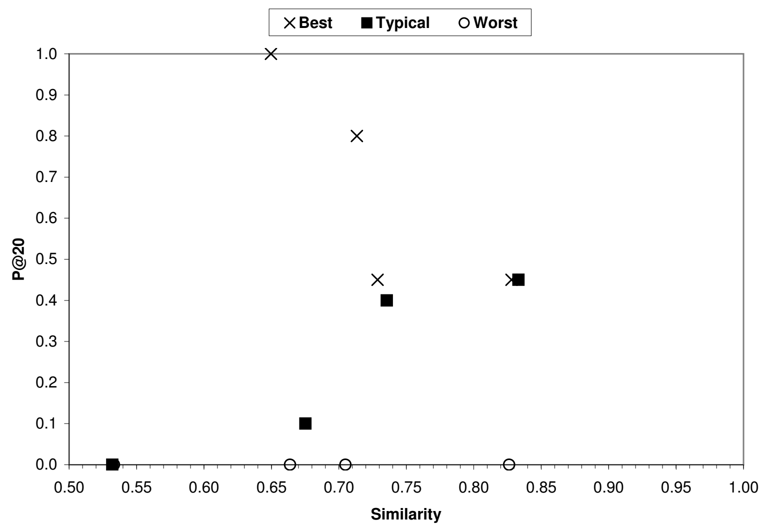

Figure 4.1: Performance of small databases on the medium-difficulty topic.

One difference that is immediately obvious upon creation of the small collections is that they do not contain as many relevant documents as the typical size collections. For example, the t401-s1 collection contains 32 documents considered relevant to topic 401, while the typical size collection t401-t1 contains 42 relevant documents. This discrepancy may cause degraded retrieval performance.

Figure 4.1 shows the performance of the small collections based on the medium-difficulty topic (topic 401). When compared with Figure 3.9, many of the same trends are evident. For the typical-case collections (denoted by squares), similarity score is closely correlated with performance ($r = 0.92$).

As with the typical size collections, only the most on-topic of naturally generated collections (t401-s1) has a performance score higher than the full WT2g collection. The t401-s1 collection has a P@20 score of 0.45. The full WT2g collection scored 0.40 on this topic, which is the same as the t401-s2 collection.

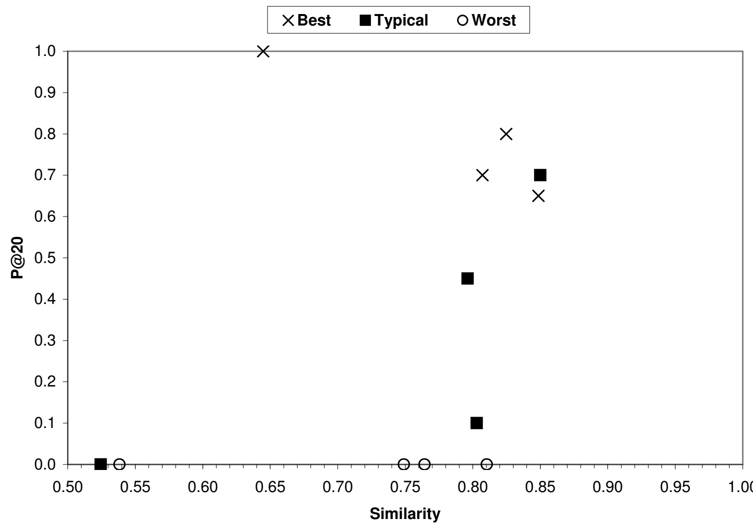

Figure 4.2: Performance of small databases on the easy topic.

The small best-case collections display the same trends as the typical size best-case collections. The similarity score of each new collection is near that of the typical-case collection it is based on, but the best-case collection always has an equal or greater performance score. These performance scores increase as the collection's similarity to the topic decreases (again, likely due to the fact that relevant documents are easier to pick out from a collection of mostly off-topic documents). One difference is that the best-case scores for the small collections increase in performance even more rapidly than the best case scores for the typical size collections. It seems that the smaller size of the collections enhances this trend. The worst-case collections, as before, display similarity scores in line with the typical case collections and have P@20 scores of zero.

Figures 4.2 and 4.3 display the performance of small size collections on the easy (441) and difficult (437) topics, respectively. As with the medium-difficulty topic, the basic trends observed in the previous chapter remain.

When we compare the typical-case collections from each of the three topics (Figure 4.4), we see

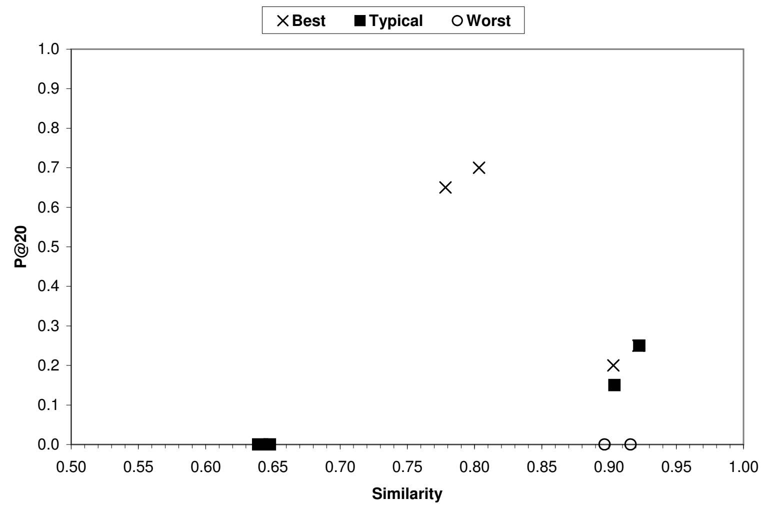

Figure 4.3: Performance of small databases on the difficult topic.

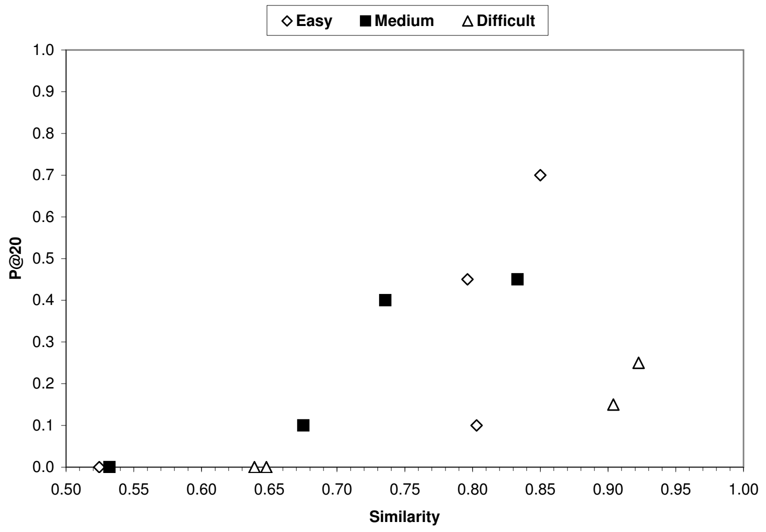

Figure 4.4: Performance of small databases on the three test topics.

that the easy topic produces the highest performance scores. Collections based on the difficult topic have the lowest performance scores, as expected.

As we observed with the typical size collections, the difficult topic produces collections with higher similarity scores than the other topics. While it is possible that there is something about difficult topics that tends to create this type of relationship, we expect that this behavior is simply due to the particular combination of topic (437) and base collection (WT2g) being used.

We have discovered that making collections smaller does not make any major changes in the trends we observed in the last chapter. Will this hold as we increase the size of collections?

4.4 Performance of Larger Collections

Now we turn our attention to "large" collections. These collections are an order of magnitude larger than the collections in Chapter 3, with each collection containing 40,000 documents. As with the small collections, there is one new collection for every existing typical collection at the typical-case relevance level. There will be a total of 3 topics $\times$ 4 similarity levels = 12 new collections created at the large size.

We will not examine best-case and worst-case large collections. Our investigations with the best-case and worst-case collections thus far have produced very consistent results. Since we do not expect the large collections to change our conclusions regarding the best and worst cases, and since the large collections can take considerable time to generate, it was deemed unnecessary to continue this line of investigation.

As with the small and typical size collections, the large collections are named based on their target topic, size (l for "large"), and similarity level. For example, the collection based on topic 441, large size, and using a slightly off-topic query is denoted t441-l2.

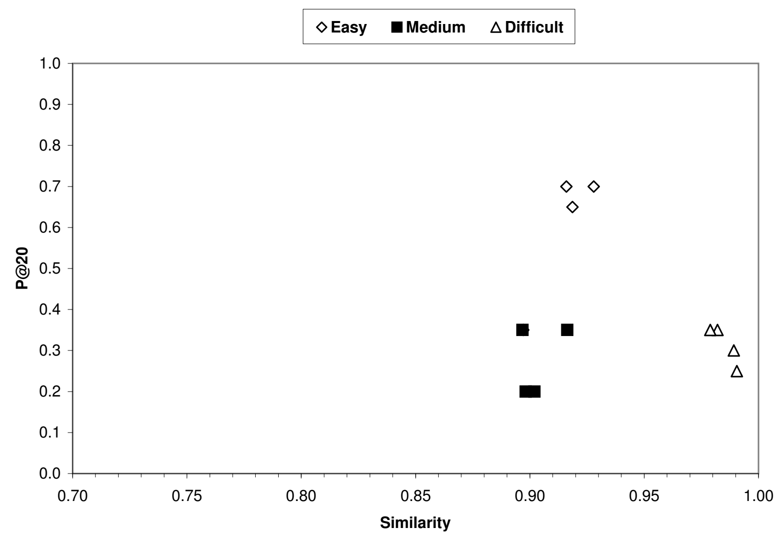

Figure 4.5: Performance of large databases on the three test topics.

Large size collections could not always be generated with the full 40,000 documents, because there were not always 40,000 documents in the WT2g set matching the target query (for actual collection sizes, see Appendix B). This is representative of the way real collections are built on the Web. If an automatic system (like a "focused crawler") were building a search engine with a particular topic in mind, it would need to stop when it ran out of documents that matched its topic.

It would be possible to simply add random documents to guarantee that each large collection contained 40,000 documents, but this would not affect the performance results. In the case of the collections that are specifically built for the query being evaluated (t401-l1, t441-l1, and t437-l1), none of the extra documents would match the query, and therefore would not appear in a result set. The documents retrieved from the collection would remain the same. In the off-topic collections (similarity levels 2 through 4), documents that were randomly added to the collection may "accidentally" match the target topic, but this would be unlikely. For search engines on the Web, accidental matches would be *extremely* unlikely, due to the sheer number of off-topic documents available.

Collection	**Similarity**	**P@20**
t437-l1	0.9905	0.25
t437-l2	0.9892	0.30
t437-l3	0.9821	0.35
t437-l4	0.9789	0.35

Table 4.1: Similarity and performance scores for the large t437 collections.

The results of evaluating the 12 large collections are shown in Figure 4.5. As usual, the easy topic (441) produces the best performance scores. The trend of the difficult topic generating the highest similarity scores continues as well.

Surprisingly, the medium-difficulty topic and the difficult topic have performance scores very near each other. Another surprising result that is difficult to see on the graph is the progression of performance scores for the difficult topic, detailed in Table 4.1. The most on-topic collection has the lowest performance score, while the off-topic collections produce higher scores. The cause of this discrepancy is unclear.

Each of the three topics produce a cluster of dots in Figure 4.5. As noted earlier, increasing the size of a collection increases its generality. Since each of these collections is based on gathering documents surrounding a central topic, increasing the size of the collections tends to increase the amount of overlap between them. The greater overlap tends to drown out the differences in the queries that were used to generate the collections.

4.5 Trends as a Factor of Size

Now that we have examined the performance of all three collection sizes separately, we can look at the data from another point of view, holding topic constant and comparing only variations in collection size.

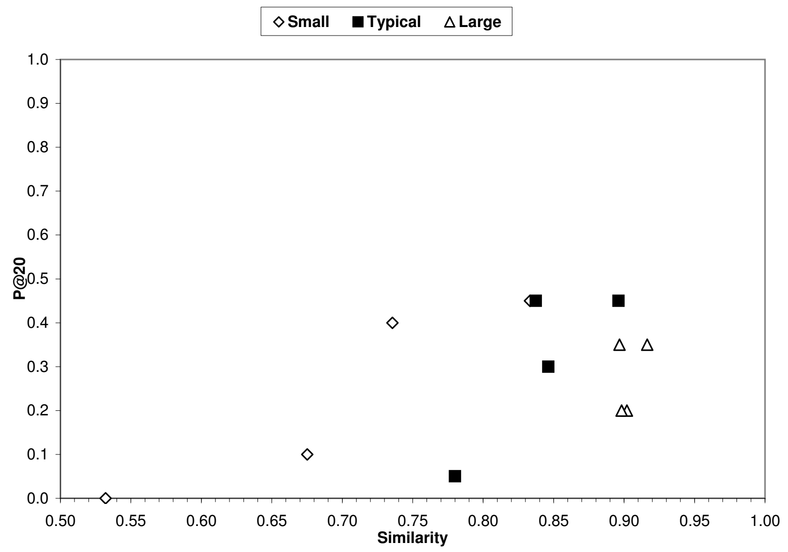

Figure 4.6: Summary of database performance on the medium-difficulty topic.

The results of the three collection sizes with respect to topic 401 are compared in Figure 4.6. This graph indicates that collection size can have an effect on similarity scores, while only moderately affecting performance. Although the effect is not nearly as large, the same trend can be seen in comparisons of topic 441 and 437 (Figures 4.7 and 4.8).

Now we see a pattern emerging. The overall theory looks like Figure 4.9. At small collection sizes, there is a great difference between on-topic collections and off-topic collections. As collection size increases (and the collection becomes more general), this variability decreases. Both similarity and performance scores tend to cluster. The larger a collection is, the more likely it is to produce good results, even if it is somewhat off topic.

At the extreme, the largest possible collection for a given topic will contain all of the documents in the full WT2g set that match any of the topic's keywords. Searching one of these collections is equivalent to searching the centralized collection. The centralized collection can be thought of as the largest possible collection on the target topic, although it also happens to contain thousands of

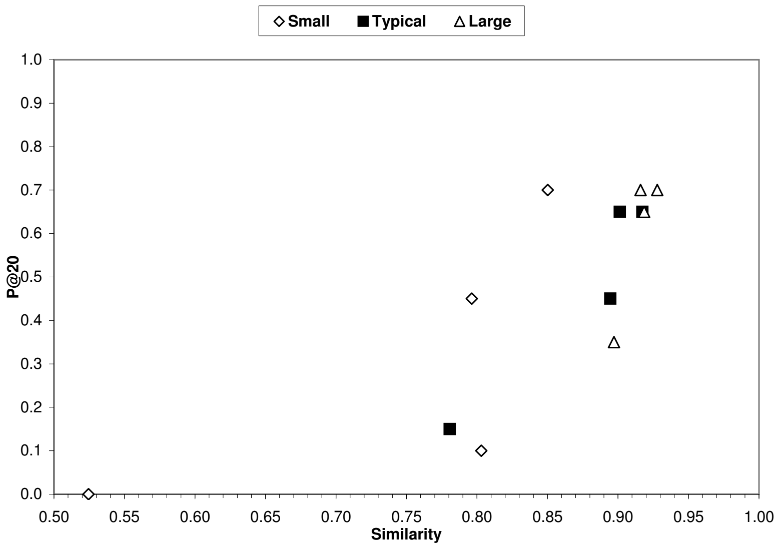

Figure 4.7: Summary of database performance on the easy topic.

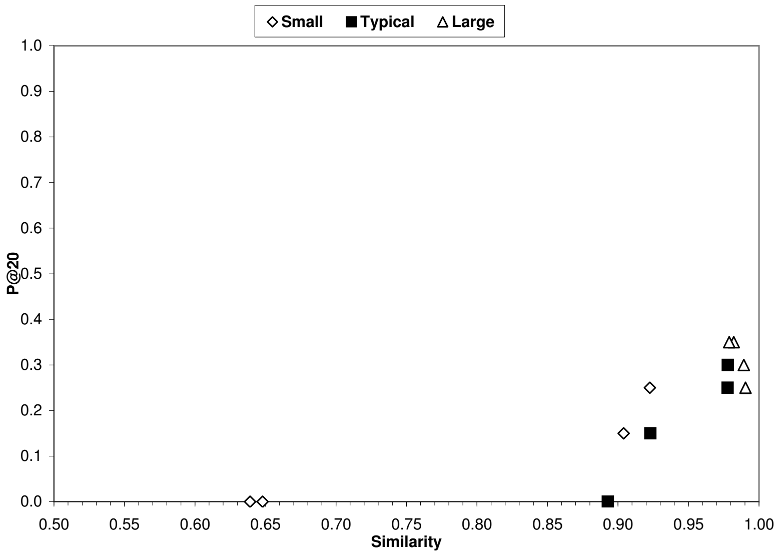

Figure 4.8: Summary of database performance on the difficult topic.

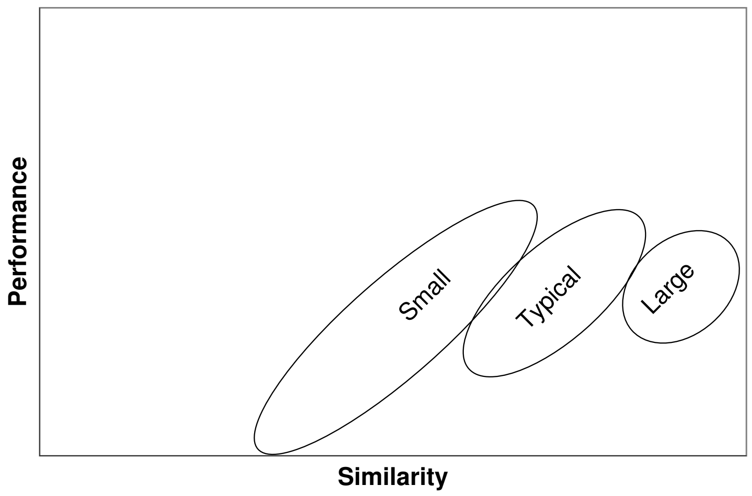

Figure 4.9: Summary of the effects of changing size on database performance.

off-topic documents that do not affect the query (because they do not contain any of the query's keywords). As we saw in the previous chapter, a centralized collection produces good results, even though it is not focused on any particular topic.

One reason for the more robust performance of large collections is the raw number of relevant documents in these collections. As mentioned previously, larger collections will naturally contain more relevant documents than smaller collections. Regardless of the retrieval algorithm used, a small collection will not be able to retrieve documents it does not contain, resulting in lowered performance scores. Figure 4.10 compares the number of relevant documents contained in each of the 12 collections based on topic 401.

These results reinforce the conclusions from the previous chapter. In general, better performance can be obtained from centralized collections than from topically-focused collections. It is important to remember, though, that centralized collections do not always contain every document that might be of interest to a user. There are many topically-focused search systems on the Web that

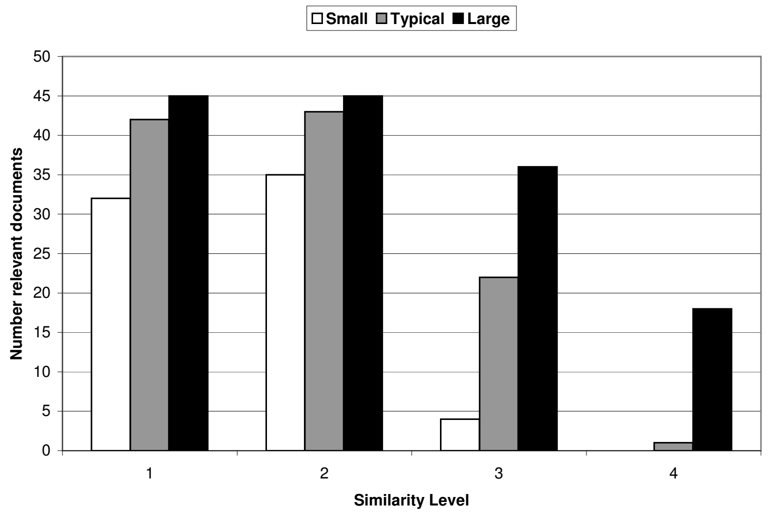

Figure 4.10: Number of relevant documents in collections based on topic 401.

contain important documents but do not allow their contents to be indexed by centralized search engines (e.g, a search engine for bugs in a particular software package).

It seems that a successful distributed retrieval system could be built by trading off size and topical focus, selecting only collections that would fall in the upper half of the graph in Figure 4.9. The distributed system could take results from large collections without worrying about topical focus. When the system encountered smaller collections, it would only accept results from these collections if they were highly focused on the topic of the current query.

But absolute collection size does not guarantee good results. The results presented in this chapter rely on the fact that collections become more general as they grow in size. While this is true, the amount of added generality depends on the algorithm used to generate the collection as well as the number of relevant documents available. It is certainly possible to create "large" collections that do not contain a general set of documents. The WT2g collection we have been using to represent the available universe contains roughly a quarter of a million documents. A topic-specific site on the Web (like espn.com, money.cnn.com, or support.microsoft.com) can contain many more documents than this and still not contain documents on a wide variety of topics. It is unlikely that such a site would contain a single document relevant to German minority integration, topic 401.

In addition to the size and central topic of a collection, we will also have to account for the amount of focus or generality in a collection. This factor will be addressed in the next chapter.

5 The Effects of Collection Focus

So far, we have looked at the predictive power of the central topic of a collection and the size of a collection, but we have not directly investigated the amount of focus that a collection has. A collection may have a central topic that is in line with the user's need, but the contents of the collection may be spread in many directions, with no documents that address the need directly.

For our topic- and size- based investigations, it would have been preferable to hold the focus of the collections constant, selecting documents within a given distance from the collection's target topic. However, these three factors are not independent. Often, a change in one factor will force a change in another.

As explained in Section 4.2, in practice there is a relationship between collection focus and collection size. Adding documents to a collection will typically decrease the collection's focus. Restricting a collection to contain documents within a certain distance from the central topic would have limited the number of documents available, restricting the maximum size of the collection. If a collection is built on a "popular" topic that is covered by millions of documents, the collection can be very focused, regardless of the size. But if the central topic of the collection is not addressed by many documents on the Web, even a small collection will fail to be highly focused on the topic.

In the preceding experiments, the “farther away” databases (txxx-x2 through txxx-x4) were created by manually tweaking the query to use alternative terms that were synonyms of the original terms and/or terms that were more general than the original terms. In addition to moving the topic away from the original (txxx-x1) information need, this type of manipulation could broaden the focus of the collection, which may have an effect on retrieval performance.

In this chapter, we will look at the effects changes in the focus of a collection have on retrieval performance. We will compare several methods for measuring a collection’s focus, including measures based on document similarity and measures based on entropy. We will create a few new collections to aid in the comparison of these measures, but most of the evaluations will be based on the collections described in the preceding chapters.

5.1 Previous Work on Collection Focus

Researchers in distributed information retrieval (DIR) have rarely investigated the effects of a collection’s focus. One possible explanation for this is that (to my knowledge) there has been no previous work in which the properties of collections were explicitly manipulated. This means the researchers had very little control over collection focus, an were forced to rely on topical similarity measures to account for any differences in focus.

As part of the development of the vector-space model, Salton showed (1975) that the most effective retrieval is obtained when documents within a collection are spread as widely as possible, reducing the likelihood of false positives. That is, a collection with a broad focus will produce better results than a highly-focused collection, because the relevant documents will be easier to distinguish from the non-relevant documents. We have already confirmed this with our investigations of the “best possible” collections (Section 3.6). In those collections, which were guaranteed to contain all documents relevant to the target query, performance increased as the central focus of the collection was moved more off topic. That is, as more off-topic documents were added to the collection, the collection became less focused, making it easier to distinguish between the relevant and non-relevant documents.

While not often used to evaluate the focus of a collection, entropy measures have been used for many purposes in information retrieval. One of the most popular uses is Kullback-Leibler divergence, also called "relative entropy" (Kullback, Keegel, & Kullback 1987). This is a similarity measure that is sometimes used to calculate query-collection similarity in DIR systems. It often works as well as the CORI algorithm described in Section 3.5 (better under certain circumstances), but is not as robust (Larkey, Connell, & Callan 2000). Another similarity measure, cue-validity variance (CVV) (Yuwono & Lee 1997) calculates the entropy of a single term with respect to a set of collections. Terms with higher entropy are given higher weights, under the assumption that these terms can more effectively distinguish between collections.

A measure called "clarity", which is a form of relative entropy, has been used to measure the focus of result sets from searches (CronenTownsend, Zhou, & Croft 2002). Highly focused result sets are deemed more likely to contain relevant items. Clarity correlates well (r between 0.368 and 0.577) with actual query performance, although this correlation is not as high as the correlations between 0.64 and 0.92 found in Section 3.7. The primary disadvantage of the clarity measure is that the query must actually be executed before the measure can be computed. This expensive operation is exactly what we are trying to avoid by finding measures that predict the performance of a query with a given collection.

5.2 Measuring Collection Focus

We will evaluate several different measures of collection focus to determine whether any of them have the ability to improve predictions of collection performance. These measures fall into two classes, measures based on entropy and measures based on similarity.

Entropy Measures

One way to measure the focus of a collection is to use a measure of entropy. Entropy literally means "randomness" or "lack of organization", so a high entropy score indicates that a collection lacks focus. Low entropy scores indicate a highly-focused collection. Claude Shannon (1948) defined the entropy of a random event X as

$$\mathrm{H}(X) = -\sum_{x \in X} \mathrm{p}(x) \cdot \log(\mathrm{p}(x)) \tag{5.1}$$

$$\mathrm{p}(x) = \text{probability of } X \text{ being in state } x \tag{5.2}$$

Essentially, if $\mathrm{p}(x)$ is low, the contribution of x to entropy will be low. If $\log(\mathrm{p}(x))$ is low (meaning $\mathrm{p}(x)$ is close to 1), the contribution of x to entropy will be low. Medium values of $\mathrm{p}(x)$ will result in higher entropy. That is, states that occur very frequently or very infrequently contribute little to an entropy score, while states that are more "random" make the entropy score higher. To treat a collection of documents as a random event, we will define a very simplistic probability measure, indicating the degree to which each term is distributed across the documents of the collection. For a collection c and a dictionary of terms T:

$$\begin{aligned} \text{term-entropy}(c) &= -\sum_{t \in T} p(t,c) \cdot \log(\mathrm{p}(t,c)) \quad (5.3) \\ \mathrm{p}(t,c) &= \text{percentage of documents in } c \text{ containing } t \quad (5.4) \end{aligned}$$

Unfortunately, term-entropy is not a "true" entropy measure. Although $\mathrm{p}(t,c)$ expresses the probability that a random document selected from the collection will contain term t, any given document will contain many terms, and the "states" in our random event will overlap. In a "true" entropy measure, the sum of all probabilities must be 1. We need something more like "the percentage of term-document occurrences that term t is responsible for":

$$\begin{aligned} \text{term-entropy2}(c) &= -\sum_{t \in T} \mathrm{p2}(t,c) \cdot \log(\mathrm{p2}(t,c)) \quad (5.5) \\ \mathrm{p2}(t,c) &= \frac{\text{number of documents in } c \text{ containing } t}{\sum_{t \in T} \text{number of documents in } c \text{ containing } t} \quad (5.6) \end{aligned}$$

Terms that appear in no documents or that appear in all documents will have low scores. Terms that appear in some documents (distinguishing terms) will have high scores, contributing to a higher term-entropy score. The more terms that are shared among all documents, the lower the overall entropy score.

One possible problem with a score like this is that there may not be enough overlap between terms across the documents. In a very unfocused collection, it is possible that many terms will appear in only a single document, causing the term-entropy score to be low (when we would expect an unfocused collection to have a high "entropy" score). In a somewhat focused collection, many terms may have high entropy scores, because they appear in only a subset of the documents. If this occurred, moderately-focused collections would have higher entropy scores than unfocused collections, and the entropy measures would not be useful in determining a collection's focus.

To test the reasonableness of the two entropy measures, several new collections were created, all with the "typical" size of 4000 documents. They are listed below, in an order that is expected to reflect decreasing collection focus:

- all440 = A collection in which all documents in the collection are copies of the same document, WT27-B27-440, which describes classroom technology at various colleges. This is the "most focused" collection possible.
- all440-435 = A collection composed of 2000 copies each of two documents, WT27-B27-440 and WT27-B27-435. This collection is highly focused, as both documents describe classroom technology at different institutions.
- all440-400 = A collection composed of 2000 copies each of two documents, WT27-B27-440 and WT27-B27-400. This collection has "mixed focus", as the two documents describe widely different topics and have few terms in common. Document WT27-B27-400 describes areas of research that fall under the heading "clean combustion of coal".
- all10 = A collection composed of 400 copies each of 10 documents randomly selected from the WT2g set. In one sense this collection is very focused, because the documents cluster into 10 tightly focused sets. In another sense, this collection is very unfocused, because there is little or no relationship between the document sets.
- germany = A collection built from the results of the query "germany", a moderately specific word that happens to appear in topic 401. This collection is expected to be relatively focused, but not as tightly focused as the collection built from topic 401.

Collection	term-entropy	term-entropy2
all440	0	8.0637
all440-435	1076	8.0633
all440-400	1107	8.0743
all10	2390	8.9497
t401-t1	3005	9.8518
germany	1311	9.7498
water	930	9.4238
WT2g	1167	10.0756

Table 5.1: Entropy-based scores.

- water = A collection built from the results of the query "water", one of the most common words in the English language. This collection is expected to be relatively unfocused, but it is expected to have more focus than the full WT2g collection.

The term-entropy and term-entropy2 measures were applied to each of these collections, along with the topically focused collection t401-t1 and the full WT2g collection. In Table 5.1, the collections are ordered by decreasing expected focus.

The scores for the original term-entropy measure are confusing. It is perfectly reasonable for the all440 collection to receive an entropy of 0, since every document contains exactly the same terms. But the score for all10, a collection created from only 10 documents, is higher than the score for the full WT2g collection. And the highest score goes to t401-t1, a collection that we expect to be relatively focused. The term-entropy2 score is much more promising. These scores generally correspond with the expected ordering, increasing from the top of the table to the bottom of the table. The main abberation is that t401-t1, germany, and water received scores that are the reverse of the expected order. Neither of the proposed entropy measures is an obviously good indicator of a collection's focus, so we will turn our attention to measures based on document similarity.

Similarity-Based Measures

Another way to compute a collection's focus is to see how "close" the collection's documents are to the collection's center. We will compare two different measures. The first measure, called average-similarity, simply averages the distances from the collection's center to each document in the collection:

1. For each document in the collection, generate list of terms in the document, and use this list as a query.
2. Calculate the similarity between each query and the collection, using the "modified" CORI formulas (from section 3.5, reproduced below):

$$\text{similarity}_{CORI}(c, q) = \sum_{t \in q} \frac{0.4 + 0.6 \cdot \text{TF}_{ModCORI}(t, c)}{|q|} \tag{5.7}$$

$$\text{TF}_{ModCORI}(t, c) = \frac{\text{frequency of } t \text{ in collection } c}{\text{frequency of } t \text{ in collection } c + 200} \tag{5.8}$$

3. Average the resultant similarity scores.

Every document in the collection contributes to the final score, giving a result that reflects the focus of the collection as a whole. A disadvantage of this method is that if the collection has a tightly focused center cluster of documents, but also contains a large number of documents that are not related to the central cluster, the score will be lowered. It may be better to only have documents in the central cluster contribute to the score.

The second method, called retrieved-focus, is based on the similarity scores that Lucene reports as documents are retrieved from a single, central query. This method ensures that only documents related to the collection's central focus are included in the score. It is calculated by the following algorithm:

Collection	average-similarity	retrieved-focus
all440	0.9089	0.6048
all440-435	0.8454	0.4699
all440-400	0.8789	0.5013
all10	0.8034	0.2149
t401-t1	0.7297	0.2059
germany	0.6289	0.0836
water	0.6182	0.0878
WT2g	0.7793	0.0657

Table 5.2: Similarity-based scores.

1. Generate a term vector representative of the collection's center. This vector consists of the 100 highest-weighted terms in the collection, using the modified CORI TF score (equation 3.13, reproduced below) to calculate weights:

$$\mathrm{TF}_{ModCORI}(t, c) = \frac{\text{frequency of } t \text{ in collection } c}{\text{frequency of } t \text{ in collection } c + 200} \tag{5.9}$$

2. Weighting all 100 terms equally, submit the term vector as a query to the collection.

3. For each document returned, obtain the similarity score reported by Lucene (see equation 3.1).

4. Average the resultant similarity scores.

Table 5.2 compares the similarity-based measures over the same collections that were used to compare the entropy-based measures. Collections are listed in expected order of decreasing focus.

Both of the similarity-based measures appear to be good indicators of a collection's focus. The primary differences are scores for the full WT2g collection, and the fact that the retrieved-focus scores drop off more drastically as collection focus is decreased. These differences make the retrieved-focus measure more useful than the average-similarity measure.

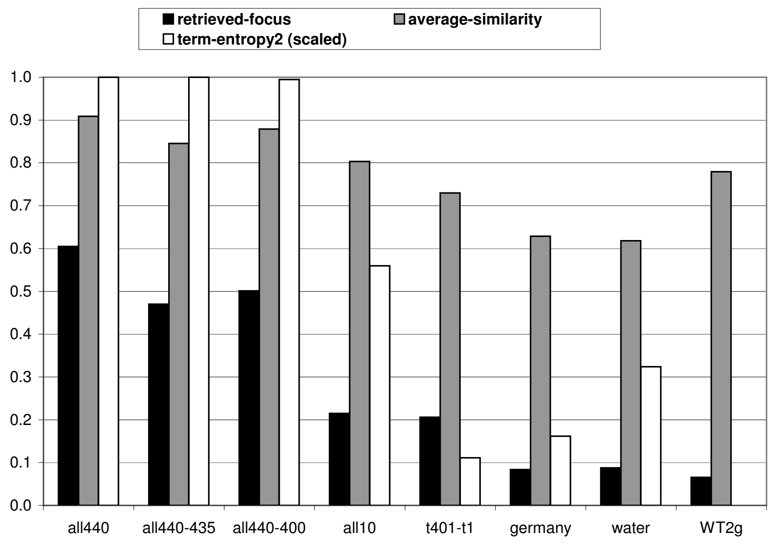

Figure 5.1: Comparison of the collection focus measures.

For better comparison, the three most promising measures are displayed together in Figure 5.1. Normally, for entropy-based measures, lower scores indicate higher focus. To provide a usable comparison, the term-entropy2 measure has been reversed (by negating the values) and scaled so that its highest value (all440) is 1 and its lowest value (WT2g) is 0. As noted previously, the term-entropy2 scores do not always have the expected values, particularly among the "moderately focused" collections t401-t1, germany, and water.

5.3 Collection Focus as a Predictor of Performance

Now we will examine the relationship between the focus and collection performance. Although qualitative assessment of Figure 5.1 indicated that the retrieved-focus measure would be the most useful, there was no correlation ($r = 0.006$) between the retrieved-focus values and the performance scores of the 36 topical collections used in the experiments of the previous chapters. Moderate correlations were obtained with both the average-similarity ($r = 0.31$) and term-entropy2 ($r = 0.33$)

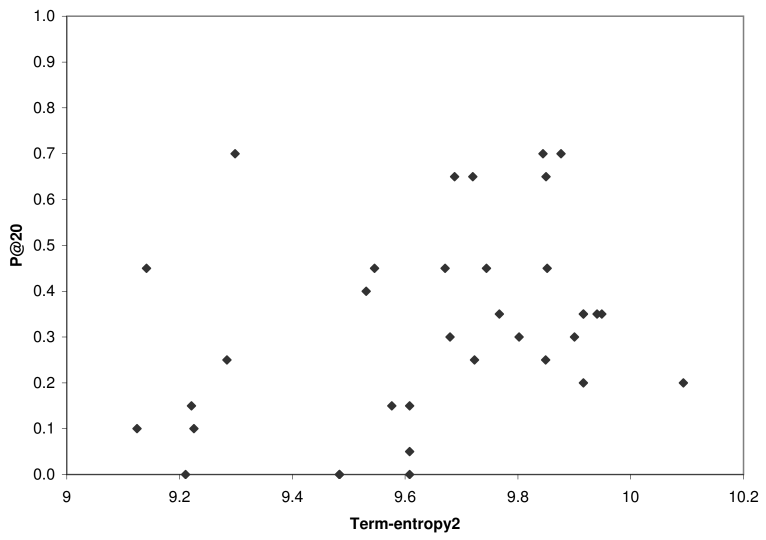

Figure 5.2: The relationship between collection focus and collection performance.

measures. Figure 5.2 shows the performance of the 36 topical collections as a function of their term-entropy2 score.

Figures 5.3 through 5.5 display the same comparison, separating out the collections based on their size and topic difficulty. For a complete listing of the focus scores for these collections, see Appendix C. It is clear that focus decreases (term-entropy2 increases) as the collection becomes larger ($r = 0.67$). This is not surprising, because these collections were built around a target topic, and the smaller collections only contain those documents that have the highest similarity to the target topic. It is unlikely that this result would hold for collections that were not topically focused.

5.4 Combining the Topic, Size, and Focus Measures

It is now possible to combine the results of the topic, size, and focus information to predict the performance of a collection. A multiple linear regression analysis was performed on the data for the 36 collections. The results are shown in Table 5.3.

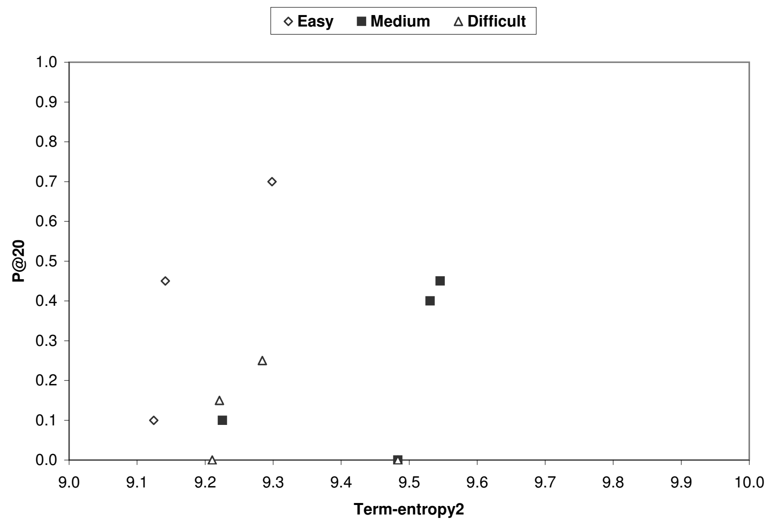

Figure 5.3: The relationship between focus and performance for the small collections.

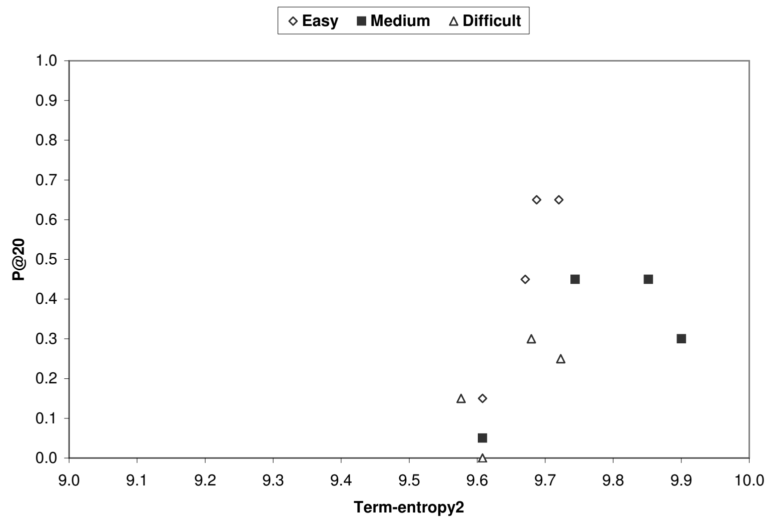

Figure 5.4: The relationship between focus and performance for the typical-sized collections.

	Topic (CORI)	**Size**	**Term-entropy2**	**Constant**
Coefficient	0.8967	-0.000003	0.1509	-1.8799
Std. error	0.3356	0.000003	0.1762	1.6296

Table 5.3: Results of multiple linear regression on candidate features.

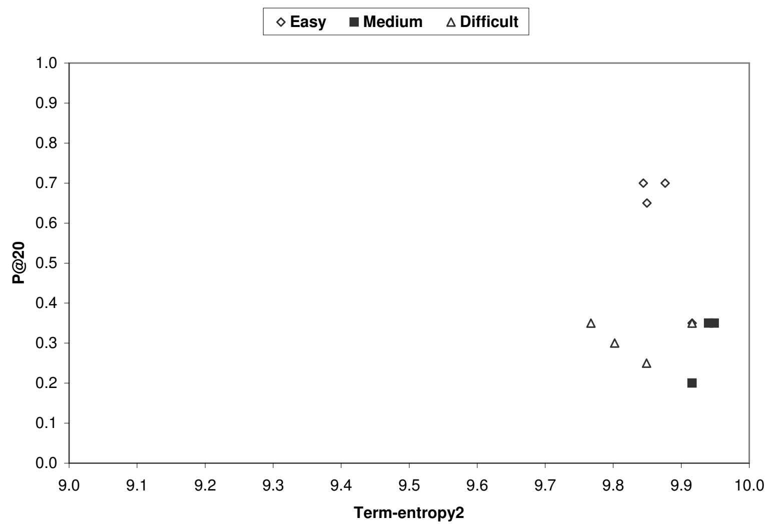

Figure 5.5: The relationship between focus and performance for the large collections.

It is clear that topic has the greatest influence on the ability of a collection to perform well. The small size of the coefficient for term-entropy2 means that affects the predicted performance, but not greatly. Although the raw entropy values typically larger than the topical similarity values, in practice they do not vary much, lessening the effect of their larger values. A surprising result, though, is that collection size matters very little. The extremely small coefficient shown is partially a result of the large collections having a size of 40,000, which is much greater than the topic similarity scores (ranging from 0 to 1). However, even collection sizes of 40,000 have little effect on the performance predicted using the indicated linear combination.

This analysis indicates that (if our only method for combining the scores is linear) the best collections will be those that have a high topic match, and are not highly focused. Based on the results of section 4.5, we will be able to apply a non-linear modification to the prediction mechanism: use the linear combination indicated above, but if the collection is very large and not highly focused, predict that it will do well regardless of the other scores.

6

Improving Topic Match With Context

We have now verified that topical focus is the best predictor of a collection's performance with respect to a query. Can we do anything to improve a distributed information retrieval (DIR) system's ability to identify collections that match the topic of a query? Algorithms for measuring the topical similarity between queries and collections have been studied for over a decade. It is unlikely that the approach of comparing term vectors that summarize queries and collections can be radically improved. While it is possible that some alternate technique will yield improvements, matching techniques developed in the last decade have failed to improve significantly on the performance of CORI.

If we do not expect the matching algorithm itself to become much better, we must look for ways to maximize the effectiveness of the matching algorithm. One way to do this is to introduce more information into the matching process. Methods for adding information include:

1. Adding formal metadata to documents in the collection, including subject headings from standardized lists of terms (e.g., see Svenonius 2000). To ensure consistency, formal metadata is often created by trained librarians, making it relatively expensive.
2. Adding informal metadata "tags" to documents in the collection (Golder & Huberman 2006).

Tags may be added by the document's author, or by readers of the document who are interested in making the document easier to find. Because tags can be created more quickly than formal metadata, they are sometimes used when formal metadata is not available.

3. Expanding the query by asking the user to submit more (or more specific) information. This is often accomplished via an "advanced search" system or by allowing users to enter their query using a more complex syntax than a simple set of keywords. This approach places a greater burden on the user, and most users do not take advantage of it. (Spink & Jansen 2004)

4. Expanding the query by an automatic process acting within the search system. This often takes the form of an iterative process known as "relevance feedback". Another method is to pre-compute information about language usage within the collection and expand queries based on this information (completing common noun phrases, for example).

5. Expanding the query by an automatic process acting on information external to the search system. This external information may be built from a set of preferences the user has submitted to the system, the user's past search history, or activities the user has recently performed. In the best case, a system will take all of these into account when determining the appropriate external information to submit with the query. For convenience, we will refer to all of these types of information as "contextual information".

The first three approaches require some amount of human effort, which means the desired information will not always be available. The maintainers of search systems (especially smaller, topically focused systems) will not want to shoulder the expense of augmenting documents with extra metadata. Individual users of search systems are not likely to change their querying habits, even with extensive encouragement and/or training. The fourth approach is interesting, but it falls somewhat outside the scope of our current investigations. If this approach improves performance, it can always be added on top of the regular matching algorithm (and this may already be the case within some of the topical search systems we want to access). In this chapter, we will focus on the fifth approach, automatic query enhancement based on contextual information. This approach has the potential to increase the length and specificity of queries without extra effort from users or search engine maintainers.

6.1 Background

A query submitted to a search system can be interpreted in many different ways, particularly if it only contains 2 words, as is typical (Lesk 1997; Spink & Jansen 2004). Researchers in cognitive science and information retrieval have long recognized that context plays an important role in interpreting information.

Marvin Minsky's concept of "frames" (Minsky 1975) allows language-understanding systems to store and use contextual information. The frame contains a summary of a stereotyped situation, indicating roles that people and objects play in the situation. As a story is read, terms from the story are assigned to roles in the frame. When an ambiguous reference is encountered, the reference can be resolved by looking at the contents of the frame and determining what term was placed in the appropriate role.

The system of scripts developed by Schank and Abelson (1977) extends the concept of frames by allowing the stereotyped situations to be more complex. For example, a script may represent the situation of eating at a restaurant. The script would contain roles for patrons, waiters, and chefs. Scenes in the script would include getting a table, ordering food, eating, and paying the bill. When a system encounters a situation for which an appropriate script can be found, the contextual information in the script can greatly improve the system's ability to interpret the situation and act appropriately.

The task of searching for information is relatively simple, and could likely be represented with a few simple scripts (e.g., "User converts information need into query. User submits query to search engine. Search engine returns results."). Unfortunately, to fully understand the query, the retrieval system would need a script describing the user's information need. Not only is it impractical to create and manage scripts covering every possible information need, but the level of detail present in a typical query would often not provide enough information to select the correct script and fill its roles. Because of these problems, information retrieval systems typically represent context with a simple set of keywords, not a more complex representation like a script.

Contextual information may be used in many ways in information retrieval systems (see Lawrence 2000 for an overview). Our focus will be on the use of contextual information for query expansion. One of the earliest systems to make use of context was HYPERFLEX (Kaplan, Fenwick, & Chen 1993), which guided users through a hypertext system, based on a manually set context. HYPERFLEX employed a learning algorithm to improve its knowledge about documents relevant to the context as users interacted with the system. Adaptive HyperMan (Mathé & Chen 1996) and Calvin (Leake *et al.* 2000) go a step farther, automatically gathering information about the context present at the time a document is accessed, and using a learning algorithm to suggest the document later when a similar context is entered.

A class of systems called "proactive information agents" collect contextual information from a user's activities and perform queries without explicit direction from the user. These systems include the Remembrance Agent (Rhodes & Starner 1996), Fab (Balabanović 1997), Watson (Budzik & Hammond 2000), Margin Notes (Rhodes 2002), and IACS (Leake, Maguitman, & Reichherzer 2005). Proactive information agents extract a context (a set of keywords) from applications running on a user's computer such as Web browsers, email clients, and word processors. The context is updated as the user reads and/or types, and queries formed from the context are periodically forwarded to a centralized search system. Result documents are constantly displayed on a portion of the user's screen, allowing relevant information to be found before the user issues an explicit query.

IBM recently deployed a system for customer support centers that can extract keywords from a conversation and use these keywords to retrieve relevant technical documents (Graham-Rowe 2004).

The systems listed above indicate that contextual information can be effective in generating queries when an explicit query is not present. Can context be used to effectively augment an existing query? The 8th TREC conference included a track devoted to investigating the effects of variations in queries (Buckley & Walz 1999). This track concluded that longer queries produce higher variability in retrieval results, but not necessarily a higher overall effectiveness. However, the "longer queries" used in this investigation had many distractor terms that could have affected the results. In another study of the use of context with a centralized search system, Kraft *et al.* (2006) compared three different methods for augmenting queries with contextual information, and found that the greatest improvements in query performance could be obtained by adding between 3 and 5 keywords to the query.

There have been some investigations of the effect of expanded queries in DIR, although not using contextual information to perform the expansion. Xu and Callan (1998) determined that a query expanded with up to 20 terms could improve retrieval performance if it was used for both collection selection and querying the collections. Ogilvie and Callan (2001) found that expanding short queries improved retrieval performance. Expanding longer queries did not work as well, presumably because the longer queries already contained enough information. Query expansion provided only small improvements for collection selection. French *et al.* (2002) found that automatic query expansion based on a controlled vocabulary could benefit a search system (both DIR and centralized) *if* the target documents were tagged with controlled vocabulary terms.

The only previous work that combines the use of context and distributed retrieval is Inquirus2 (Glover 2001). Inquirus2 asked the user to choose a topical category when a query is submitted. These categories covered extremely broad topics, such as "personal home pages" and "research papers". It is unclear whether this type of contextual information improved selection of collections, but it did increase the number of relevant documents retrieved from the collections that were selected.

6.2 Varying the Amount of Information Available in a Query

To see how the availability of contextual information affects retrieval performance, we will simulate the effects of a context-tracking system by expanding the query with extra keywords from the TREC topic. It is possible to represent context with more sophisticated structures than simple sets of keywords, but search engines operate primarily on keywords, making it difficult to utilize additional complexity in a contextual representation.

For a moment, we will step back from our focused collections, concentrating on the centralized collection built from the entire WT2g set and the full set of topics from the TREC-8 Web Track. Recall that Web Track participants were given a set of Web pages (WT2g) and 50 topic descriptions. Each topic description consists of a two- or three-word title, a one-sentence description, and a multi-sentence longer narrative section. If we consider the title the "basic query", the description and narrative sections represent two levels of additional context available for each topic.

Figure 6.1 compares queries that were formed using 5 different levels of simulated context. All

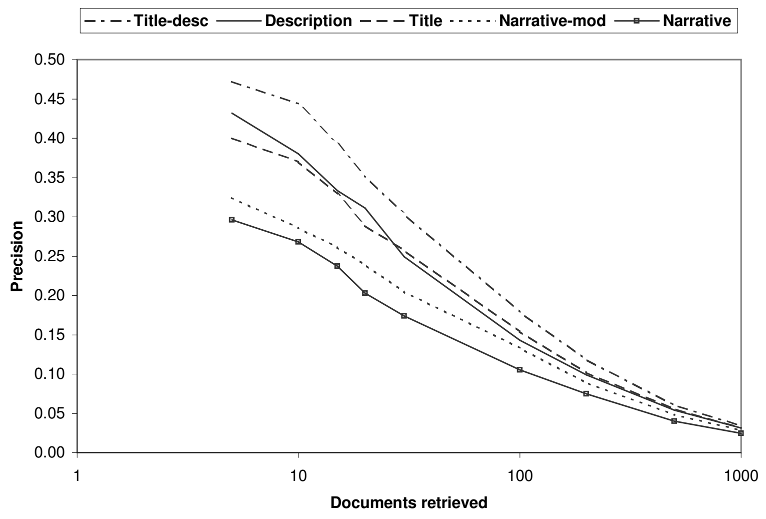

Figure 6.1: The effect of varying query context over the full set of Web Track topics.

queries had a basic set of stopwords removed, but no other processing was performed on them. Documents were retrieved from the full WT2g collection, and the resultant precision scores from all 50 topics were averaged. The 5 levels of simulated context are:

- Title = The title only
- Description = The description only
- Narrative = The narrative only
- Narrative-mod = The narrative, with an additional set of stopwords removed.
- Title-desc = The description, augmented with two copies of the title, to give the most important terms higher weight.

Queries based on the description section of the topic fared better than queries based on the title, showing that a adding a small amount of context to short queries does indeed help. Surprisingly, the narrative portion of the query, which provides the most context available for these topics,

```
<top>
<num> Number: 401
<title> foreign minorities, Germany
<desc> Description:
What language and cultural differences impede the integration of foreign
minorities in Germany?
<narr> Narrative:
A relevant document will focus on the causes of the lack of integration
in a significant way; that is, the mere mention of immigration difficulties
is not relevant.  Documents that discuss immigration problems unrelated to
Germany are also not relevant.
</top>
```

Figure 6.2: Topic 401

performed poorly. A closer investigation of the description for topic 401 (Figure 6.2) reveals the reason. The narrative portions of the topics provide specific descriptions of documents that are considered relevant and non-relevant. These descriptions often contain terms like "relevant", "significant", "documents", and "unrelated", which are not normally considered stopwords, but have nothing in common with the topic. When these terms are removed (Narrative-mod), performance is increased. The modified narrative performance scores are still not as high as those for titles or descriptions, indicating that too many distractor terms remain in the narrative descriptions. Many of the narratives specify types of documents that are not relevant to the topic, and these descriptions often contain terms that decrease query performance. A highly-intelligent system would be required to parse the narrative descriptions and determine which terms truly apply to the topic at hand. I believe that this behavior is a peculiarity of the TREC topics. Automatic methods for gathering contextual information (like the methods used by the "proactive information agents") would not typically include these kinds of distractor terms.

Since the terms contained in the topic title are always a more succinct version of the topic description, it was expected that improved results could be obtained by using the title to augment the description, indicating which terms should receive the highest weighting. As expected, the Title-desc queries consistently performed better than any of the other query formulations. This indicates that a system could be very successful if it took a query from the user and augmented it with contextual information, as long as the system weighted the human-generated query terms higher than terms that came from the automatically-generated context.

6.3 The Effect of Context on Focused Collections

In a DIR system, context can be used in two ways: as additional information to use in selecting collections to query, and as additional information that can be sent with the query along to the selected collections.

Using Context to Select Collections

First, we will investigate the effect of contextual information on collection selection. Adding contextual information has no effect on the documents in the collection itself, and so will not affect the collection's score on measures of collection size or focus. Only the query-collection similarity is affected. In the preceding chapters, collections were built that explicitly varied each property under investigation. It does not make sense to build collections with varying amounts of context. A collection, by the simple fact that it contains many documents, contains a large amount of contextual information. Here, we focus on context available in the query.

The five levels of simulated context described above were used as queries to generate query-collection similarity scores for each of the 12 "typical size" focused collections. Figure 6.3 compares the scores for the collections based on the medium-difficulty topic (Topic 401). It is clear that adding or removing context has an effect on the similarity score, but it is unclear whether that effect will

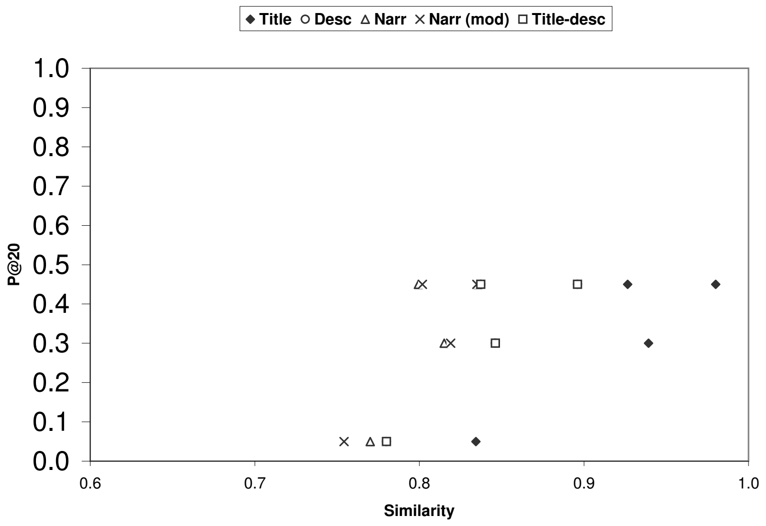

Figure 6.3: The effect of varying context on similarity, for the medium-difficulty topic.

be helpful in selecting collections to query. Note that scores fall along horizontal lines, because we are only using the contextual information while determining query-collection similarity, not while submitting the query to each collection. This has the effect of varying the similarity scores while holding the P@20 score for each collection constant. The queries based on the title alone produced the highest similarity scores. Adding more context to the queries lowered the scores in a relatively uniform manner.

Although the ability of context to raise and lower the overall similarity scores is an interesting effect, it is more useful to see whether context will improve our ability to discriminate between collections that perform well and collections that do not. The title-only queries provide the greatest separation between the scores for the most on-topic (t1) collection and the most off-topic (t4) collection. This separation is lowest for the queries with the most context (those based on the narrative portion of the topic), indicating that a large amount of context may actually interfere with the process of selecting collections.

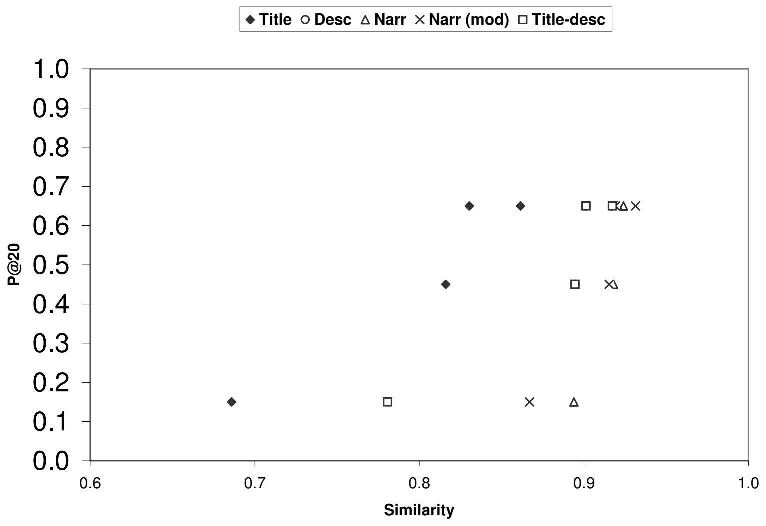

Figure 6.4: The effect of varying context on similarity, for the easy topic.

Figures 6.4 and 6.5 display query-collection similarity scores for the five levels of context using the easy collections and the difficult collections, respectively. The scores are ordered somewhat differently for the easy collections, and no ordering is evident for the difficult collections. However, the same trend remains: queries based on title and description provide a greater spread in similarity scores between the most on-topic and the most off-topic collections, indicating that shorter, more concise queries are better for calculating topical similarity.

One oddity in these graphs is that the CORI measure showed no difference between the description and title-desc queries. This is due to the fact that CORI (in either the original form or our modified form) doesn't take the weight of query terms into account. CORI simply treats a query as a list of terms, and calculates scores for those terms based on their frequency within the collection. To determine whether the title-desc queries would provide more useful similarity scores, the document frequency (DF) similarity measure was used to generate a parallel set of scores. This measure produced similar results, with the title-desc queries receiving scores very similar to the description queries (exact scores may be found in Appendix D).

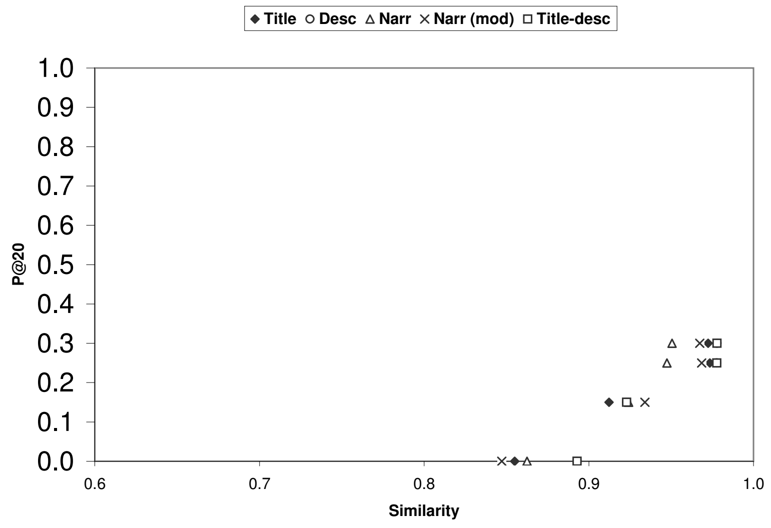

Figure 6.5: The effect of varying context on similarity, for the difficult topic.

Using Context to Query Collections

If context is not helpful in the process of selecting a collection, will it be helpful when the query is sent to the collection? Previous systems that used context (Budzik & Hammond 2000; Ogilvie & Callan 2001; Kraft *et al.* 2006) have shown that adding context is useful when querying general-purpose collections. We want to determine whether that result will hold for more focused collections.

In the previous section, we varied context in the similarity calculations only, holding the performance scores constant (with the values from description-based queries, which we have been using thus far). In this section, we are varying context when querying the target collections, which only affects the performance scores. Similarity scores are held constant for each particular collection, so all values end up arranged in horizontal lines.

Scores for the five levels of simulated context across the 12 topically-focused collections are shown in Figures 6.6 through 6.8. On the medium-difficulty topic, scores based on the narrative and

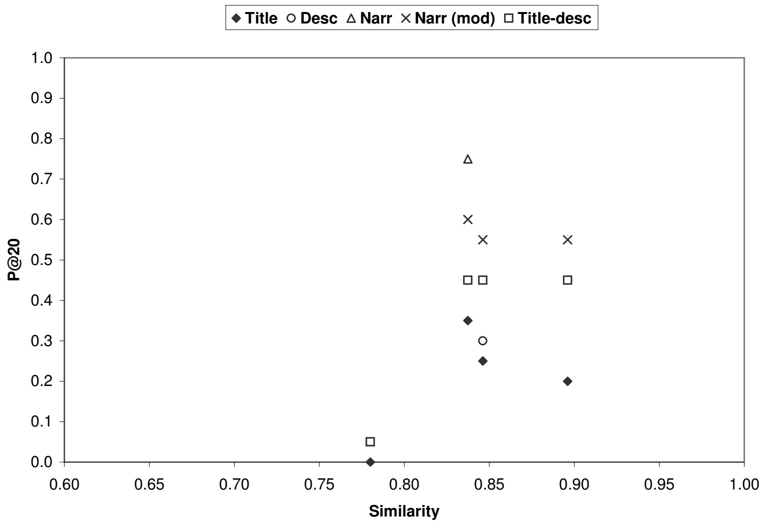

Figure 6.6: The effect of varying context on query performance, for the medium-difficulty topic.

modified narrative are the highest, indicating that context can provide improvements. On the easy topic, the narrative queries produced the worst scores of all. This could have been caused by the presence of the generic terms “reports” and “research” in the narrative query (which do not appear in the narratives for the other topics). But even when these terms are removed to produce the modified narrative query, the scores are not as high as those produced by the other types of queries. For collections based on the difficult topic, queries based on the description section performed better than other queries, although all of the queries performed poorly.

It is possible that the erratic performance of queries with greater context can be explained by the way the TREC topic descriptions are constructed. The topic descriptions often describe the information need in a formal, abstract way. The “easy” topic (441), regarding the prevention and treatment of Lyme disease, uses the terms “prevention” and “treatment”. These terms will not necessarily occur in documents that describe methods for preventing and treating Lyme disease.

In the narrative portions of the TREC topics, significant terms often appear a single time, making

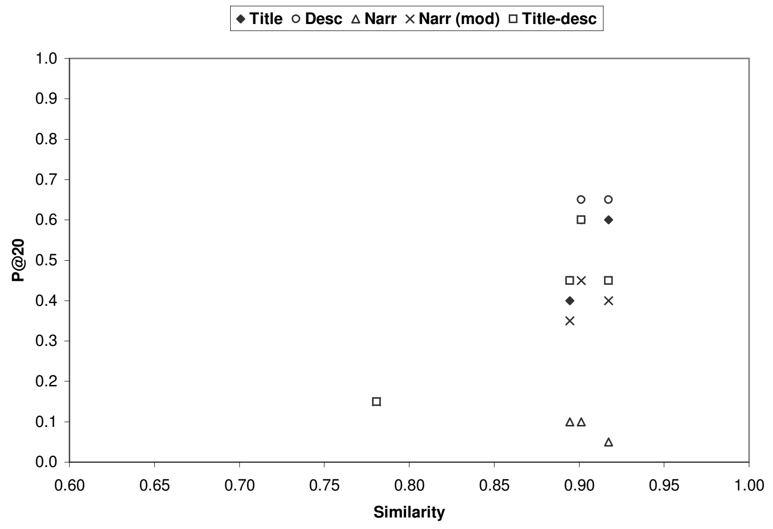

Figure 6.7: The effect of varying context on query performance, for the easy topic.

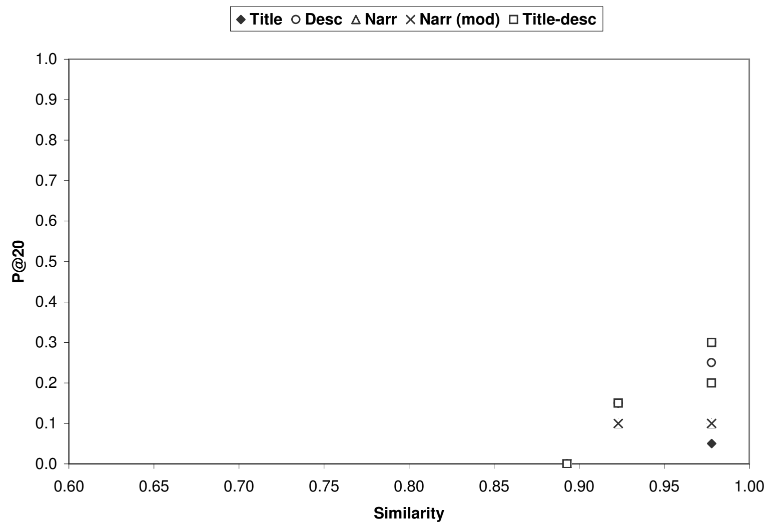

Figure 6.8: The effect of varying context on query performance, for the difficult topic.

it difficult to distinguish the important terms from unimportant terms. The documents used in other context-generation systems are typically longer, and it is often easier to pick significant words out of them. A "proactive information agent" would be expected to identify the most significant terms in a document, giving them high weights, and include a sampling of other, less significant terms. This is very similar to the content of the title-desc queries. In our experiments, the title-desc queries performed well. They performed the best on the difficult topic, near the top on the easy topic, and in the middle of the pack on the medium-difficulty topic. It appears that using a small amount of context (5 terms or less) may be a way to provide some "safety" in the results, decreasing the variability of the resultant performance scores.

6.4 Discussion

It appears that a small amount of contextual information can produce better performance from general-purpose collections, and can produce better performance from focused collections under the right circumstances. However, there is still much work to be done before contextual information can reliably be used to enhance search results.

Expanded queries derived from the sections of the TREC topics may not be an accurate reflection of the type of queries that would come from an actual context-tracking system. Since the TREC topic descriptions are manually created, we would expect them to be indicative of the "true" context that surrounds a query. But due to the succinct nature of the descriptions as well as the tendency to include both positive and negative examples in the narrative sections, it is possible that an automatic context-tracking system could produce queries that would outperform the queries generated from the TREC topics.

There are still problems in selecting the correct context that fits with a query. For any given query, there are many available contexts, including all of the applications the user has open at the time, all of the user's past search history, and even the "ambient context" of conversations occurring in the room (typically unavailable to a search system). The user's information need may occur within any of these contexts. It may be difficult to tell which of the available contexts (if any) is relevant to the query. There has been some work in this area (Horvitz *et al.* 1998), but the problem has not been solved.

Once the correct contextual information is selected, it may not be usable with all search systems. Some search engines (including Google) only return documents to the user that contain all of the terms in the query. If the entire context is used as a query, the results may be artificially limited. Queries with more than a few terms are unlikely to return many results from such a system. To mitigate this problem, Google places an absolute limit on the number of terms allowed in a query (currently 32). These constraints work well for general-purpose collections containing billions of documents. With more focused collections, constraints like this make the use of contextual information much more difficult.

7

Putting It All Together

The previous chapters have given some indication that under certain conditions, distributed information retrieval can be successful. However, determining these conditions requires knowledge of essential properties of the collections being evaluated. Measuring these properties can be problematic. A distributed information retrieval (DIR) system will seldom have access to the collection backing a search engine, for reasons including:

1. The web is large, and there are many search engines to deal with. It is not a process that can be accomplished manually.

2. Search engine operators have little incentive to cooperate with a DIR system, and may in fact see the system as competition. For example, in March 2005, Agence France-Presse (AFP) sued Google for $17.5 million because Google was making AFP's photos and headlines available on the aggregated Google News service.

3. Even in cases where search engine operators want to cooperate (e.g., with OAI-PMH), doing so can take significant effort (Lagoze *et al.* 2006).

Now that we know how the properties of a collection can be used to predict retrieval performance, we will examine a prototype DIR system that demonstrates how well this prediction works in a realistic situation. Chapter 2 described DIR in some detail. In this chapter, we will briefly discuss how the steps of a DIR system are implemented for testing. The steps are, in preparation for queries:

1. Locate search engines.
2. Determine a method for communicating with each search engine.
3. Form a representation of each search engine to store in the selection system.

And when a query is received:

1. Apply a selection algorithm to the set of collection representatives, calculating the suitability of each collection to the query.
2. Forward the query to the collections that are expected to return the best results.
3. Apply a merging algorithm to combine results from the selected collections, and present the results to the user.

Detecting the presence of a search system is an interesting problem. Users could submit pages for detection, or a crawling algorithm could be used to search the web for pages that have searches. The algorithm for taking a page and locating a possible search system would involve parsing the page and looking for a text entry box near the word "search". We will not address the problem of detecting search systems here. If the task of automatically locating search systems is too difficult, we can rely on manually-added systems.

7.1 Communicating With a Search Engine

Once a search engine has been located, there must be a way to interact with it. A DIR system interacts with each of its component search engines through a piece of code called a "wrapper". The wrapper knows how to translate between the search system's native input/output formats and the format required by the DIR system. Depending on the search engine, the wrapper may use one of three communication methods:

1. Standardized search protocol.
2. Protocol specific to the search engine.
3. "Screen scraping".

The need for interoperability between search engines has been recognized for some time, and some standard protocols for interacting with search engines have emerged. One of the oldest standard protocols is Z39.50[1], originally developed for communication between library catalogs. SRU (Search and Retrieve via URL)[2] is a re-engineered version of Z39.50 that has been gaining in popularity for digital library systems. The online store Amazon.com has recently published a standardized method of communicating with search engines called OpenSearch[3] in conjunction with its search engine, A9.com. These protocols provide standard ways to format queries and standardized result formats, allowing seamless interaction with the search engine.

Some search systems make their services available through proprietary communication protocols. The most notable example of this is the popular search engine Google, which allows programs to access services via the Google APIs[4].

[1]http://www.niso.org/z39.50/z3950.html
[2]http://www.loc.gov/standards/sru/
[3]http://opensearch.a9.com/
[4]http://www.google.com/apis/

When a search engine does not provide a protocol for direct use, the only option available to a DIR system is a "screen scraping" method. Screen scraping involves sending queries to the search engine using a URL of the same form that the search engine's graphical interface generates. The search engine returns results in its normal manner, using an HTML format that is appropriate for human viewing. This HTML must then be parsed (or scraped) to identify individual result items. One of the earliest Web-based DIR systems, Apple's Sherlock (Montbriand 1999), relied on manually-generated "screen scraping" wrappers. Screen scraping is error-prone, and the parsing algorithm must be updated any time the search engine's display format changes. Methods of automatic wrapper generation for screen scraping systems (Kushmerick, Weld, & Doorenbos 1997; Ashish & Knoblock 1997; Adelberg & Denny 1999; Fan & Gauch 1999; Zhao *et al.* 2005) are just beginning to achieve acceptable levels of accuracy.

A problem common to proprietary protocols and screen scraping methods is that query interpretation is dependent on the characteristics of the individual search system. Some search engines allow special operators to boost the weight of a query term, while others weight each term equally. Some search engines allow an unlimited number of terms to be used in a single query, while others limit number of terms. Even when the number of terms is unlimited, practical constraints may limit the number of terms that are sent to a search engine. For example, while some search engines employ a straightforward TFIDF algorithm (as described in Section 3.2), others limit result sets by only returning documents that contain all terms in the query. In this type of search engine, it is unlikely to find results with a query of more than 5 search terms. This is one of the primary reasons automatic systems like Watson (Budzik & Hammond 2000) do not use the popular Google search engine.

We will not worry about the problems of interacting with search systems here. The usage of standardized protocols like SRU and OpenSearch is increasing as more search engines operators realize the benefits of making their data available for search by others. For purposes of the prototype DIR system, we will assume that an appropriate communication method is available. However, we will not assume that we have unlimited access to the collection of documents held by a search engine, because this type of access is very uncommon. Even when search engines provide a standardized interface for searching, they are often unwilling to provide direct access to their documents. Therefore, the contents of a search engine must often be estimated.

7.2 Building a Collection Representation

So far, we have examined collections that allow us full knowledge of their contents. In "real" collections on the Web, this will not be the case. The method for building a representation of a search engine must be completely automated. Search engine contents change frequently, and the representation must be regenerated on a regular schedule to maintain accurate information about the search engine's contents.

On the Web, since we will be unable to analyze the contents of a collection directly, we must do it indirectly. An effective way to do this is by sampling the documents that the collection contains and using the sample as a representative for the full collection. A sample from a collection can be treated as a smaller copy of that collection to which all of the regular algorithms can be applied.

Query-based sampling (Callan, Connell, & Du 1999) is a method of sampling a database to retrieve a small set of representative documents. These documents can be used to calculate statistics of the full collection without having explicit cooperation of the database. Query-based sampling works by sending a random dictionary term to a search engine. If no documents are returned in response to this query, another term is selected. Once at least one document is retrieved from the search engine, all of the terms in the documents are added to a set of "known terms" for the

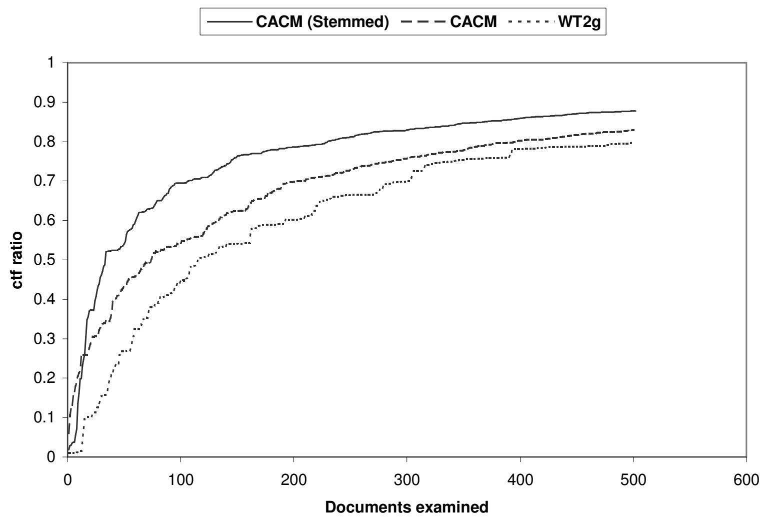

Figure 7.1: Convergence of query-based sampling.

collection. From this point on, single-term queries are randomly selected from the list of known terms. Callan and Connell (2001) found that by sampling 500 documents from a collection, a highly accurate representation of the entire collection was formed.

Note that when collections are small, the sampling process may not be able to retrieve all 500 documents required for true convergence. A sampling process may take a long time to identify 500 unique documents from a collection of 4000; and when the collection size is 400, it will not be possible to retrieve 500 unique documents. To avoid infinite searches, the process is limited to examining a total of 5000 documents (including duplicates) from any single collection.

Query-based sampling takes some time. Hundreds of searches must be performed. Hundreds of documents must be downloaded and processed. However, this process only needs to be done once for each collection (or once each time a collection's representative needs to be updated), and after that queries can be sent to the collection fairly rapidly.

Figure 7.1 shows the convergence of three collections as they are sampled. The CACM(Stemmed) collection is one of the collections used by Callan in the initial development of query-based sampling. In this collection, a stemming algorithm has been applied to combine similar terms. The CACM collection is the same collection, without stemming applied. Finally, the WT2g collection is shown.

The values on the x-axis represent the number of unique documents seen at each point in the sampling process. Values on the y-axis represent the percentage of total term occurrences in the collection that are represented by the terms seen in the sampled documents (collection term frequency ratio). For a collection c, a dictionary of terms T, and a dictionary of sampled terms T':

$$\text{ctf ratio} = \frac{\sum_{t \in T'} \text{ctf}(t)}{\sum_{t \in T} \text{ctf}(t)} \tag{7.1}$$

$$\text{ctf}(t) = \text{number of times term } t \text{ occurs in collection } c \tag{7.2}$$

In a large collection, 500 documents may contain only a small percentage of the total terms in the collection. However, the documents gathered by query-based sampling contain the terms that are used most throughout the collection, as evidenced by the high ctf ratios in Figure 7.1. Note that sampling is affected by stemming. The WT2g data set takes a bit longer to learn than the CACM data set. Whereas 300 sampled documents is sufficient for a stemmed database, 500 documents is a more reasonable number for databases where stemmed terms are not available.

Estimating Topic

How does sampling affect the properties we have examined previously? Since sampled collections have high ctf ratios, we expect the sampled documents to be highly representative of term usage within the full collection.

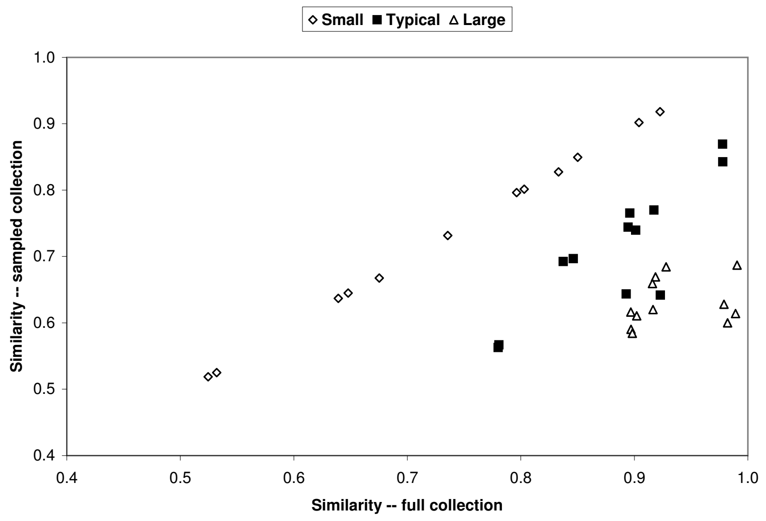

Figure 7.2: Affect of sampling on topical similarity scores.

The 36 topic-specific collections (3 topics, 3 sizes, 4 similarity levels) used in previous chapters were sampled using query-based sampling. Query-collection similarity scores were computed using the modified CORI measure, as in Chapter 3, and the similarity scores were compared. The results, shown in Figure 7.2, reveal some interesting patterns. If the sampled collection provided a perfect representation of the full collection, all values would lie along a straight line. Overall, the correlation is moderate ($r = 0.34$), but when the various sizes of collections are separated, the picture becomes clearer. As would be expected, there is a near-perfect correlation ($r > 0.99$) between the small collections and their sampled versions. This is due to the fact that the sampled collections contain almost all of the documents in the original collections. With the typical-size collections, the correlation drops a bit ($r = 0.86$). And the correlation is barely evident ($r = 0.24$) for the large collections. In other words, as a collection grows larger, our ability to accurately sample its contents diminishes.

At first glance, this appears to be a problem. But, upon closer inspection, we find that all of the points that are "out of line" come from collections based on the difficult topic. When the four large t437 collections are excluded, the large-collection correlation becomes much more acceptable ($r = 0.90$). That is, the topic for which we had difficulty retrieving quality results is the same topic that proves most difficult to sample. And, on the difficult topic, the sampled collections always have lower similarity scores than the original collections, which will *improve* performance of the selection algorithm (the topic-specific collections will not be selected as often for queries involving the difficult topic). Whether this result will generalize to other topics is still an open question.

Estimating Size

Very little work has been done on automatically estimating the size of a search engine's collection. In (Liu, Yu, & Meng 2001), a method for determining a collection's size based on sampling is described. The size estimate is based on the frequency with which duplicate documents are retrieved during the sampling process. Accuracy can be quite good (within 5 percent) when 2000 documents are sampled out of a database of 300,000. Since our analysis has indicated that size is not worth including in the calculations, we will not try to determine a collection's size for purposes of the prototype.

Estimating Focus

The sampled versions of our 36 collections were measured with the term-entropy2 measure (described in Section 5.2). The results are shown in Figure 7.3. As with the topical similarity measure, correspondence was quite good ($r > 0.99$) for the small collections, and not as good but still strong for the typical-size collections ($r = 0.68$) and for the large collections ($r = 0.70$). It is interesting to note that the focus scores overall are much more consistent than the topical similarity scores ($r = 0.87$).

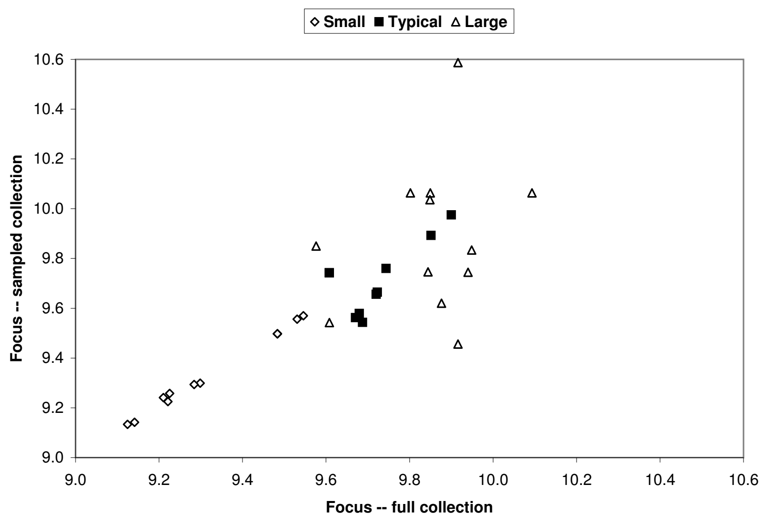

Figure 7.3: Affect of sampling on term-entropy2 scores.

Surprisingly, the term-entropy2 score does not fare as well when large, general-purpose collections are sampled. When the full WT2g collection and the WT2g-half collection were sampled, the resultant term-entropy2 scores fell in the middle of the ranges exhibited by the focused collections. This creates problem for using term-entropy2 in a practical setting. An effective DIR algorithm must be able to distinguish between topically-focused collections and general-purpose collections. The potential problems with focused collections make it desirable to query at least one general-purpose for every information need. Because of this, the prototype DIR algorithm will need to make use of both the term-entropy2 and the retrieved-focus scores.

7.3 Result Merging

Merging result sets from multiple search systems should be straightforward, but no result merging algorithm has yet been recognized as having a clear advantage over other algorithms in all situations. Voorhees has suggested several merging algorithms (Voorhees, Gupta, & Johnson-Laird

1995; Voorhees 1997), but all require training data on documents known to be relevant and non-relevant from each collection being used. Callan (2000) describes several merging algorithms that were used with the INQUERY system, but all of these are shown to have flaws. The best-performing algorithm requires cooperation of the target search engines. Other merging strategies (like that described in Manmatha & Sever 2002) require only minimal cooperation from the individual search engines. This typically requires reporting of internal similarity scores, which are not always available, and can be misleading.

Two recent comparisons of merging algorithms (Tsikrika & Lalmas 2001; Lu *et al.* 2005) came to the same conclusion: merging algorithms perform best when each result document's title and summary are used to compute its position in the final ordering, rather than using only the document's rank ordering alone. However, these results were obtained based on studies of systems that merge results from multiple general-purpose search engines, not the more focused search engines we are concerned with. While document titles are almost always available from topical search engines, in practice document summaries are often not available. Many topical search engines provide very brief summaries, or no summaries at all, with their result lists. In initial tests, it was determined that the use of titles alone to merge documents was not adequate (all P@20 scores were at or near 0).

The unavailability of summary information and the poor performance of title information alone forces us to use the rank-based merging algorithm described in (Yuwono & Lee 1997). This algorithm assumes that the first result from each search engine has an equal chance of being relevant, and then spreads apart subsequent documents from each search engine in a manner inversely proportional to the engine's goodness score. For result document d from selected collection c, the ranking score is:

$$\begin{aligned}
\text{score}(d,c) &= 1-(\text{rank}(d,c)-1)\cdot\text{spread}(c) && (7.3)\\
\text{rank}(d,c) &= \text{position of document } d \text{ in collection } c\text{'s result list} && (7.4)\\
\text{spread}(c) &= \frac{\text{minimum goodness score of any collection for this query}}{\text{documents being retrieved}\cdot\text{goodness of collection } c} && (7.5)
\end{aligned}$$

Duplicate documents must be removed from the result list. Often, this can be done by comparing URLs, but since some documents can be referenced by multiple URLs, it is often best to compare the document summaries. In our case, it is possible to unambiguously identify documents by their WT2g ID number. When duplicate documents are located, their scores are added, under the presumption that documents returned by multiple search engines are more likely to be relevant than documents returned by a single search engine.

7.4 Testing the Combined System

As a practical test for the combined system, a set of collections was gathered. This set contains a general-purpose collection (full WT2g) and several more focused collections. Four highly-focused collections were included: the three highly-focused t1 collections, as well as a highly-focused collection based on topic 413 (this collection had previously been used as an "off-topic" t4 collection). Six moderately focused collections included the "water" and "germany" collections from Chapter 5. These were supplemented with four new collections, built using a single prominent term from each of topics 402, 403, 404, and 405: "genetics", "osteoporosis", "ireland", and "cosmic". Each of these four terms matched less than 4000 documents in the WT2g collection, so no artificial restrictions were placed on the size of these collections. The full set consists of one general-purpose collection and 10 topic-specific collections: 4 highly focused on one of the target topics, 5 moderately focused on a given term from a target topic, and one moderately focused on a general term ("water").

An alternative set of tests was run, replacing the full WT2g collection with the WT2g-half collection, in which half of the documents have been uniformly removed. This simulates a situation common in actual Web search: the centralized collection has some documents relevant to the query, but the topical collections contain many relevant documents that are not available to the centralized collection.

The DIR Algorithm

Distributed retrieval was performed according to the following algorithm:

1. Each collection was sampled using query-based sampling to create a collection representative that contained 500 documents or less.
2. The term-entropy2 score (from Section 5.2) and retrieved-focus score (from Section 5.2) for each collection representative is pre-computed.
3. For each query received,
   (a) Calculate each collection's base "goodness score" using the sampled collection representative, according to the formula:

$$\text{goodness}(q, c) = \alpha \cdot \text{similarity}_{CORI}(q, c) + \beta \cdot \text{term-entropy2}(c) \tag{7.6}$$

   (b) If retrieved-focus$(c) < \delta$, increase the goodness score by γ.

(c) If goodness$(q, c) > \gamma$, submit the query to the actual search engine and collect the result documents. Otherwise, ignore the engine.

(d) When results from all matching search engines have been collected, merge them in order according to the scores from Equation 7.3 above.

In the initial experiments, α was set to 0.8967, and β was set to 0.1509, based on the results of the multiple linear regression shown at the end of Chapter 5. However, this proved to give too much weight to collection focus, and allowed many off-topic collections to match the queries. The value of β was halved, to 0.0755, which provided much better results.

Collections with extremely low focus are considered to be general-purpose collections, and are consulted for every query. The threshold for determining these collections (δ) was set to 0.05, based on manual examination of the retrieved-focus scores for sampled collections. The threshold goodness score for an engine to be selected (γ) was set to 1.32, based on manual examination of the goodness scores that resulted from this algorithm.

Results

Figure 7.4 compares the performance of retrieval from centralized collections (WT2g and WT2g-half) with retrieval from a distributed system that contains the 10 focused collections and one of the centralized collections. Selector-full indicates the distributed system that contains the full WT2g collection, while selector-half indicates the distributed system with WT2g-half.

It is clear that WT2g-half does not perform as well as the full WT2g collection, and this is to be expected, because it only contains half of the relevant documents for each topic. It is surprising, though, that the selector-full and selector-half scores track so closely along their associated WT2g scores. Using a distributed system does not greatly help retrieval performance. Neither does it

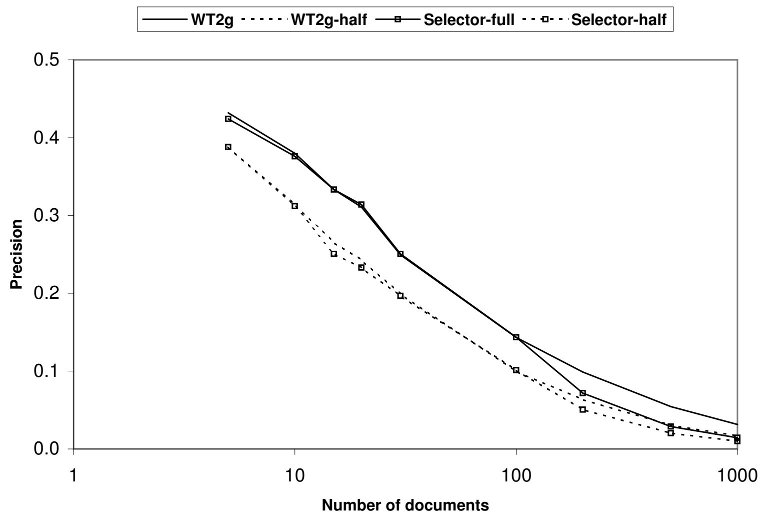

Figure 7.4: Comparison of the source selector vs. the centralized collections over all 50 topics.

greatly harm the retrieval performance, which is important for topics that do not have any matching focused search engines. By setting the goodness threshold high enough, we were able to avoid unnecessary selection of “off-topic” collections that would have contributed poor documents to the final result set.

Even though the topical collections on their own can outperform the WT2g-half collection (see Section 3.8), using the distributed system to combine documents from the topical collections and WT2g-half does not work as well, presumably because of the difficulties in choosing the best document ordering in the merging process. Even for the four topics that were explicitly covered by focused collections, the selector-half system did no better than the WT2g-half collection alone.

Admittedly, the number of collections in this sample is small, but it gives a general idea of the performance we can expect of a DIR system using this approach. It is clear that much more work needs to be done before a DIR system can provide substantial benefits over a large, centralized collection.

8

Conclusions and Future Work

This dissertation has explored the process of distributed information retrieval (DIR) from the standpoint of a system that must automatically select search engines on the Web. The most important result from these explorations is a better understanding of the factors in a search engine's underlying collection that affect retrieval performance, and how these factors may be measured.

The experiments in this dissertation have focused on TREC topics 401 through 450 and the WT2g collection, using simple keyword-based representations of queries and collections, and using vector-space matching algorithms. All results described below should be interpreted in that context. While it is expected that the results will apply to other vector-space retrieval systems, they are unlikely to apply to systems based on different retrieval algorithms. Additional experiments on a broader range of datasets would be desirable to further examine the hypotheses arising from this dissertation.

8.1 Summary of Results

The experiments presented in this dissertation support a number of hypotheses. When a collection is topically focused, the best results will be obtained when there is a high degree of similarity between the central focus of the collection and the user's need. Regardless of a collection's topical focus, collections that contain more documents relevant to the query are more likely to perform well than collections that contain few documents relevant to the query. This was shown in the experiments with the "best case" and "worse case" collections, and was further supported when the topically focused collections produced better results than the WT2g-half collection.

As collections become larger (and more general), there is a greater chance that they will perform well. Small collections only perform well when they are centered exactly on the topic of the query. If a collection is small, the similarity score must be high to produce good results. Larger collections can perform well despite lower similarity scores. Very large (e.g., centralized) collections always perform well. Therefore, if an engine is known to be very large (with respect to the universe of documents), there is little point in finding its "central" topic.

Collection focus does not appear to affect retrieval performance as much as central topic, but collection focus does have an affect on performance. Highly focused collections do not perform as well as more general collections. A collection's central topic and focus can be accurately estimated by using a simple sampling technique to retrieve a subset of the full collection. While there is no simple method for estimating the size of a collection, knowledge of collection size is not necessary for accurate selection. When detailed information about a user's search preferences, search process, or current information need is available, retrieval performance may be improved by adding a small amount of contextual information to the query when available, but contextual information is unlikely to aid in the process of selecting topically focused search engines.

Retrieval from a distributed system is much more difficult than retrieval from a centralized system, and the overall performance may not be better. Distributed information retrieval is a complex process, and each step in the process is a place where noise may be introduced. If a topical collection matching the target query does not exist in the current set of collections, the system can do no better than sending the query to one or more general-purpose search engines. If a topical collection does exist, that collection will only provide a benefit if it contains relevant documents that the available general-purpose engines do not contain. Even then, the additional documents may get "drowned out" by documents from general-purpose engines or other (less relevant) topically focused sources.

Although the current results indicate that centralized systems can provide more reliable results, it is still possible for a centralized system to provide links to more focused collections when it detects that the focused collections will provide good results. Smaller, more focused collections are still useful, since they can provide novel results. Perhaps the best approach is to start off using a general-purpose search engine, and if results are not adequate, get suggestions of successively more focused engines, knowing that the very small engines are a big risk. This approach mirrors the "ramping interface" approach suggested by Rhodes (2000) as a means of assisting the user in maintaining the best cost/benefit ratio for the search process.

8.2 Future Directions

The obvious next step for this work is to confirm the results with "real" search engines on the Web. We have previously performed some initial work with real search engines (Leake & Scherle 2001), but not to the extent required to verify the results presented here. To fully verify these results would require gathering a set of topic-specific search engines, writing wrappers to communicate with the engines, and recruiting human judges to evaluate the relevance of hundreds of documents. Care would need to be taken to ensure the target topics had enough overlap with the topical search engines being used. We are only aware of one previous experiment that has attempted to use distributed retrieval techniques with topically-focused Web search engines (Ipeirotis & Gravano 2002). In this experiment, the resultant precision scores were much lower than expected. This is likely due to minimal overlap between the topics covered by the 50 topic-specific engines in the test set and the 50 TREC topics used for testing.

Throughout this dissertation, we have seen that some critical pieces of DIR have still not been the object of adequate research. It would be useful to flesh these out. The largest holes are methods for automatic discovery of search engines, methods for communicating with search engines, and methods for estimating the size of search engines. Many of these problems could be solved by wider adoption of standard search protocols, but there will always be some search engines that choose not to follow these protocols. Having better automated methods to handle these systems would greatly increase the number of search engines that are available in a distributed system, providing better coverage of the topics users may search for.

It would be useful to look more closely at tagging and other methods of “folk cataloging”. Since contextual information can improve retrieval performance, can user commentary about documents also improve performance? A wealth of user-contributed data is being cataloged at sites like Delicious[1] and the Open Directory Project[2]. In addition to expanding the number of keywords associated with a document, tags may be used to describe features that are orthogonal to topic (e.g., dates or generic descriptors like “useful”). This user-submitted information has the potential to greatly improve retrieval performance, because the human-generated descriptions of documents are likely to be more precise than simply examining all keywords in each document. However, tags

[1]http://del.icio.us
[2]http://dmoz.org/

and user-generated categories can also contain a great deal of noise, which may be difficult for a retrieval system to filter out.

In addition to informal "folk cataloging", a DIR system may be able to take advantage of the implicit information contained by a document's popularity on the Web. This popularity may be measured directly (Zhu & Gauch 2000), or more indirectly, using a link-citation algorithm like Google's PageRank (Brin & Page 1998). Is there a good way to determine the most popular search engine within a particular topic? Is there a way to determine whether a search engine makes use of linking or popularity calculations within its internal document ranking? While popular pages will not always be the best results, it is worthwhile to examine the effects of popularity on the ranking process.

Popularity is one factor that affects the "difficulty" of a topic. Topic difficulty clearly affects the usefulness of results. More work needs to be done to determine whether it is possible to predict the difficulty of a query, and how to best handle this knowledge in a distributed system. Our results initially point towards avoiding distributed retrieval for difficult topics, but there may be types of topics for which general-purpose search engines will perform poorly, indicating that topical search engines would be a better first choice.

Although the majority of human searchers rely on general-purpose search engines (Graham & Metaxas 2003), searchers sometimes use topically-focused search engines. As we have seen, selection of topically-focused search engines that will perform well for a given query is a difficult task. An interesting question for future research is whether human searchers who use topic-specific search engines (manually performing distributed retrieval) produce better results than could be produced by performing the same search in a general-purpose search engine. If the human searchers do produce better results, it would be worthwhile spend more research effort on accurately modeling the decision processes that expert human searchers use and determining whether such a model would improve automatic distributed retrieval systems.

More work needs to be done in determining when contextual information can usefully be applied to the retrieval process. Methods need to be developed that can distinguish between a context related to the query and a completely unrelated context. More sophisticated methods for parsing contextual information need to be developed, to boost the weights of terms that are important for topic discrimination, and remove terms that are likely to cause spurious matches. One possible source of improvement in this area is to use more sophisticated methods for tracking contextual information. Instead of storing context as a single set of keywords, contextual information could be divided into sets of keywords (e.g., positive and negative terms) that could be handled differently. Some sets of keywords would be added to the query as in the experiments described here, but other sets of keywords would be modified before being added to the query. For example, if the search engine's language supports negation, negative terms could used to exclude documents from the search results. Richer contextual information may also aid the the process of merging results in a distributed retrieval system.

A

Appendix: Topics and Queries

This appendix details the topics and queries that were used to generate the topically-focused collections. Each of the three target topics is listed below, along with the actual queries that were used to generate collections at various similarity levels for the topic. When the queries are listed, an "x" is used as a placeholder. For example, the query labeled t441-x1 was used to select 4000 documents for the t441-t1 collection, while the same query was used to select 40,000 documents for the t441-l1 collection.

Easy topic - 441

Query difficulty can be affected by two major factors. A query can be "easy" because the information is widely available, or because there is a unique word in the query that would only occur in relevant documents. This topic was relatively easy due to the rare word "Lyme". However, since the desired result set was a subset of the information available on Lyme disease (prevention and treatment only), this topic provided medium difficult when using title-only queries.

```
<top>

<num> Number: 441

<title> Lyme disease
```

```
<desc> Description:
How do you prevent and treat Lyme disease?
<narr> Narrative:
Documents that discuss current prevention and treatment
techniques for Lyme disease are relevant.  Reports of
research on new treatments of the disease are also relevant.
</top>
```

Actual queries used in generating the collections were:

- t441-x1: “prevent treat Lyme disease”
- t441-x2: “prevent treat tick illness”
- t441-x3: “prevent treat illness”
- t441-x4: “methods producing steel”

Medium difficulty topic - 401

This topic was of medium difficulty. Again, since the title did not fully specify the intended documents, the title-only queries proved more difficult than the description-based queries. Interestingly, for this query, the t2 set contained more relevant documents than the t1 set. The only query terms repeated were “integration” and “germany”. When these terms are used as a query by themselves, 44 of the 45 relevant documents are included in the (4000 size) result set. This had no obvious effects on the results.

```
<top>
<num> Number: 401
<title> foreign minorities, Germany
```

```
<desc> Description:
What language and cultural differences impede the integration of foreign
minorities in Germany?
<narr> Narrative:
A relevant document will focus on the causes of the lack of integration
in a significant way; that is, the mere mention of immigration difficulties
is not relevant.  Documents that discuss immigration problems unrelated to
Germany are also not relevant.
</top>
```

Actual queries used in generating the collections were:

- t401-x1: "what language cultural differences impede integration foreign minorities germany"
- t401-x2: "what dialect conflicts slow alien integration germany"
- t401-x3: "conflicts slow integration"
- t401-x4: "methods producing steel"

Difficult topic - 437

Difficult queries can be difficult either because they are inherently ambiguous (a poorly-specified query), the terms of the query are ambiguous ("to be or not to be"), or because little information on the topic is available. This topic appears to fall in the latter category.

```
<top>
<num> Number: 437
<title> deregulation, gas, electric
<desc> Description:
```

```
What has been the experience of residential utility customers
following deregulation of gas and electric?
<narr> Narrative:
Documents that discuss privatization of government-owned utilities
 alone are not relevant.  Also, not relevant are documents that
discuss the deregulation of utilities for commercial customers.
</top>
```

Actual queries used in generating the collections were:

- t437-x1: "experience residential utility customers following deregulation gas electric"
- t437-x2: "experience residential people following deregulation utilities"
- t437-x3: "experience people less regulation"
- t437-x4: "methods producing steel"

B

Appendix: Results of Topic and Size Testing

This appendix contains the similarity and precision scores that serve as a basis for the discussions in Chapters 3 and 4. Collections in the following tables are designated by the name of the topic from which they were derived, with the form tXXX-YN. In the place of XXX is the number of the target topic. Y is replaced the letter by t, s, or l to indicate small size, typical size, or large size. N is replaced by a number from 1 to 4 that indicates how similar the collection is to the target topic (see Appendix A for details). A suffix of 'rel' indicates that the collection has been modified to include all documents relevant to the target topic. A suffix of 'wst' indicates that the collection has been modified to include none of the documents relevant to the target topic.

For example, the following designations are given to the typical size collections based on topic 401:

- t401-t1 = documents retrieved using topic 401's description as a query
- t401-t2 = documents retrieved by modifying the topic description slightly
- t401-t3 = documents retrieved by modifying the topic description greatly
- t401-t4 = documents retrieved using the description of a completely different topic

In all cases, statistics for the full WT2g collection are included to aid in comparison, although WT2g should not be considered a “small” or “typical” size collection.

Typical size, topic 441 (easy)

	CORI Similarity	Df Similarity	Centroid Similarity	Number Relevant	P@20	Average Precision
wt2g	0.9279	0.0202	0.0165	29	0.6500	0.5055
t441-t1rel	0.9174	0.1286	0.1102	29	0.6500	0.6643
t441-t2rel	0.9035	0.0968	0.0709	29	0.8000	0.7447
t441-t3rel	0.9031	0.0995	0.0714	29	0.9000	0.8119
t441-t4rel	0.8378	0.0244	0.0103	29	0.8500	0.8800
t441-t1	0.9173	0.1290	0.1104	29	0.6500	0.6645
t441-t2	0.9013	0.0967	0.0702	16	0.6500	0.4438
t441-t3	0.8946	0.0990	0.0709	10	0.4500	0.3045
t441-t4	0.7807	0.0235	0.0092	3	0.1500	0.0897
t441-t1wst	0.8837	0.1280	0.1083	0	0.0000	0.0000
t441-t2wst	0.8557	0.0961	0.0689	0	0.0000	0.0000
t441-t3wst	0.8505	0.0988	0.0694	0	0.0000	0.0000
t441-t4wst	0.7726	0.0235	0.0090	0	0.0000	0.0000

Typical size, topic 401 (medium)

	CORI	Df	Centroid	Number	P@20	Average
	Similarity	Similarity	Similarity	Relevant		Precision
wt2g	0.9167	0.0301	0.0289	45	0.4000	0.2094
t401-t1rel	0.8967	0.0941	0.0945	45	0.6500	0.6194
t401-t2rel	0.8461	0.0707	0.0743	45	0.3000	0.2089
t401-t3rel	0.8389	0.0651	0.0659	45	0.5000	0.3938
t401-t4rel	0.7803	0.0298	0.0196	45	0.9000	0.6547
t401-t1	0.8961	0.0947	0.0913	42	0.4500	0.2267
t401-t2	0.8462	0.0707	0.0742	43	0.3000	0.2057
t401-t3	0.8374	0.0651	0.0617	22	0.4500	0.2070
t401-t4	0.7800	0.0294	0.0155	1	0.0500	0.0019
t401-t1wst	0.8953	0.0952	0.0872	0	0.0000	0.0000
t401-t2wst	0.8440	0.0707	0.0678	0	0.0000	0.0000
t401-t3wst	0.8330	0.0647	0.0564	0	0.0000	0.0000
t401-t4wst	0.7794	0.0294	0.0155	0	0.0000	0.0000

Typical size, topic 437 (difficult)

	CORI Similarity	Df Similarity	Centroid Similarity	Number Relevant	P@20	Average Precision
wt2g	0.9909	0.0486	0.0447	14	0.3000	0.1878
t437-t1rel	0.9777	0.1374	0.1435	14	0.2500	0.1590
t437-t2rel	0.9778	0.1052	0.1154	14	0.3000	0.1900
t437-t3rel	0.9332	0.0664	0.0600	14	0.4500	0.4875
t437-t4rel	0.9169	0.0530	0.0529	14	0.6000	0.8022
t437-t1	0.9778	0.1373	0.1433	14	0.2500	0.1590
t437-t2	0.9779	0.1054	0.1151	12	0.3000	0.1857
t437-t3	0.9230	0.0661	0.0547	3	0.1500	0.0637
t437-t4	0.8929	0.0525	0.0490	0	0.0000	0.0000
t437-t1wst	0.9769	0.1375	0.1402	0	0.0000	0.0000
t437-t2wst	0.9767	0.1052	0.1110	0	0.0000	0.0000
t437-t3wst	0.9221	0.0660	0.0526	0	0.0000	0.0000
t437-t4wst	0.8925	0.0525	0.0490	0	0.0000	0.0000

Small size, topic 441 (easy)

	CORI Similarity	Df Similarity	Centroid Similarity	Number Relevant	P@20	Average Precision
wt2g		0.0202	0.0165	29	0.6500	0.5055
t441-s1rel	0.8486	0.1868	0.1956	29	0.6500	0.5734
t441-s2rel	0.8072	0.1422	0.1130	29	0.7000	0.6629
t441-s3rel	0.8247	0.1458	0.1247	29	0.8000	0.7864
t441-s4rel	0.6447	0.0306	0.0261	29	1.0000	0.9881
t441-s1	0.8501	0.1881	0.2014	26	0.7000	0.5640
t441-s2	0.7963	0.1442	0.1106	9	0.4500	0.2745
t441-s3	0.803	0.1457	0.1169	2	0.1000	0.0690
t441-s4	0.5245	0.0189	0.0102	0	0.0000	0.0000
t441-s1wst	0.8104	0.1868	0.1946	0	0.0000	0.0000
t441-s2wst	0.749	0.1392	0.0982	0	0.0000	0.0000
t441-s3wst	0.7643	0.1452	0.1123	0	0.0000	0.0000
t441-s4wst	0.5384	0.0193	0.0109	0	0.0000	0.0000

Small size, topic 401 (medium)

	CORI Similarity	Df Similarity	Centroid Similarity	Number Relevant	P@20	Average Precision
wt2g		0.0301	0.0289	45	0.4000	0.2094
t401-s1rel	0.828	0.1030	0.1528	45	0.4500	0.2713
t401-s2rel	0.7287	0.0790	0.0998	45	0.4500	0.3951
t401-s3rel	0.7134	0.0731	0.0938	45	0.8000	0.6182
t401-s4rel	0.6497	0.0406	0.0867	45	1.0000	0.9397
t401-s1	0.8332	0.1036	0.1536	32	0.4500	0.2301
t401-s2	0.7356	0.0794	0.1043	35	0.4000	0.3046
t401-s3	0.6752	0.0770	0.0613	4	0.1000	0.0486
t401-s4	0.5321	0.0267	0.0225	0	0.0000	0.0000
t401-s1wst	0.8263	0.1087	0.1392	0	0.0000	0.0000
t401-s2wst	0.7049	0.0819	0.0707	0	0.0000	0.0000
t401-s3wst	0.6638	0.0739	0.0512	0	0.0000	0.0000
t401-s4wst	0.5333	0.0255	0.0208	0	0.0000	0.0000

Small size, topic 437 (difficult)

	CORI Similarity	Df Similarity	Centroid Similarity	Number Relevant	P@20	Average Precision
wt2g		0.0486	0.0447	14	0.3000	0.1878
t437-s1rel	0.9032	0.1749	0.2636	14	0.2000	0.1630
t437-s2rel	0.9219	0.1352	0.1873	14	0.2500	0.1788
t437-s3rel	0.8034	0.0764	0.1087	14	0.7000	0.9614
t437-s4rel	0.7784	0.0637	0.0971	14	0.6500	0.9635
t437-s1	0.904	0.1737	0.2610	12	0.1500	0.1569
t437-s2	0.9226	0.1331	0.1838	10	0.2500	0.1488
t437-s3	0.6479	0.0727	0.0499	0	0.0000	0.0000
t437-s4	0.6391	0.0579	0.0420	0	0.0000	0.0000
t437-s1wst	0.8967	0.1810	0.2582	0	0.0000	0.0000
t437-s2wst	0.9161	0.1361	0.1687	0	0.0000	0.0000
t437-s3wst	0.644	0.0719	0.0496	0	0.0000	0.0000
t437-s4wst	0.6459	0.0578	0.0426	0	0.0000	0.0000

Large size, topic 441 (easy)

	CORI Similarity	Df Similarity	Centroid Similarity	Number Relevant	Size	P@20	Average Precision
wt2g		0.0202	0.0165	29	247,491	0.6500	0.5055
t441-l1rel				29	26,454		
t441-l2rel				29	22,434		
t441-l3rel				29	22,100		
t441-l4rel				29	24,199		
t441-l1	0.9279	0.0714	0.0391	29	26,454	0.7000	0.6623
t441-l2	0.9186	0.0638	0.0341	18	22,423	0.6500	0.4713
t441-l3	0.916	0.0642	0.0343	16	22,087	0.7000	0.4623
t441-l4	0.8971	0.0246	0.0179	7	24,177	0.3500	0.2288
t441-l1wst				0	26,425		
t441-l2wst				0	22,405		
t441-l3wst				0	22,071		
t441-l4wst				0	24,170		

Large size, topic 401 (medium)

	CORI Similarity	Df Similarity	Centroid Similarity	Number Relevant	Size	P@20	Average Precision
wt2g		0.0301	0.0289	45	247,491	0.4000	0.2094
t401-l1rel		0.0780	0.0611	45	40,000	0.3500	0.1896
t401-l2rel		0.0529	0.0402	45	23,903	0.2000	0.1342
t401-l3rel				45	15,981		
t401-l4rel				45	24,204		
t401-l1	0.9164	0.0781	0.0612	45	40,000	0.3500	0.1897
t401-l2	0.902	0.0529	0.0402	45	23,903	0.2000	0.1342
t401-l3	0.8968	0.0451	0.0366	36	15,972	0.3500	0.2159
t401-l4	0.8982	0.0334	0.0304	18	24,177	0.2000	0.1063
t401-l1wst				0	40,000		
t401-l2wst				0	23,858		
t401-l3wst				0	15,936		
t401-l4wst				0	24,159		

Large size, topic 437 (difficult)

	CORI Similarity	Df Similarity	Centroid Similarity	Number Relevant	Size	P@20	Average Precision
wt2g		0.0486	0.0447	14	247,491	0.3000	0.1878
t437-l1rel				14	40,000		
t437-l2rel				14	40,000		
t437-l3rel				14	40,000		
t437-l4rel				14	24,180		
t437-l1	0.9905	0.0982	0.0791	14	40,000	0.2500	0.1853
t437-l2	0.9892	0.0760	0.0616	14	40,000	0.3000	0.2056
t437-l3	0.9821	0.0633	0.0506	12	40,000	0.3500	0.2286
t437-l4	0.9789	0.0523	0.0530	11	24,177	0.3500	0.2937
t437-l1wst				0	40,000		
t437-l2wst				0	40,000		
t437-l3wst				0	40,000		
t437-l4wst				0	24,166		

C

Appendix: Results of Focus Testing

The following tables display the scores received by various collections on the four focus measures, as described in Chapter 5. For ease of comparison, collections are grouped by size.

Varying focus

	term-entropy	term-entropy2	retrieved-focus	average-similarity
WT2g	1167	10.0756	0.0657	0.7793
all440	0	8.0637	0.6048	0.9089
all440-435	1076	8.0633	0.4699	0.8454
all440-400	1107	8.0743	0.5013	0.8789
all10	2390	8.9497	0.2149	0.8034
germany	1311	9.7498	0.0836	0.6289
water	930	9.4238	0.0878	0.6182

Typical size

	term-entropy	term-entropy2	retrieved-focus	average-similarity
t441-t1	2056	9.7201	0.1691	0.6859
t441-t2	1996	9.6875	0.1533	0.6804
t441-t3	2000	9.6708	0.1556	0.6825
t441-t4	2174	9.6080	0.1419	0.7061
t401-t1	3005	9.8518	0.2059	0.7297
t401-t2	2491	9.9003	0.1623	0.6952
t401-t3	1911	9.7440	0.1395	0.6817
t401-t4	2174	9.6080	0.1419	0.7061
t437-t1	2341	9.7228	0.1910	0.7231
t437-t2	2354	9.6796	0.1761	0.7127
t437-t3	2111	9.5764	0.1823	0.7012
t437-t4	2174	9.6080	0.1419	0.7061

Small size

	term-entropy	term-entropy2	retrieved-focus	average-similarity
t441-s1	1536	9.2984	0.1691	0.5242
t441-s2	1225	9.1414	0.1450	0.5080
t441-s3	1364	9.1246	0.1868	0.5241
t441-s4	1858	9.4835	0.1374	0.5420
t401-s1	3047	9.5457	0.2444	0.5979
t401-s2	2566	9.5309	0.2135	0.5684
t401-s3	1723	9.2254	0.1841	0.5392
t401-s4	1858	9.4835	0.1374	0.5420
t437-s1	1971	9.2211	0.2622	0.5875
t437-s2	2296	9.2841	0.2372	0.5891
t437-s3	1637	9.2105	0.1722	0.5482
t437-s4	1858	9.4835	0.1374	0.5420

Large size

	term-entropy	term-entropy2	retrieved-focus	average-similarity
t441-l1	2366	9.8765	0.1423	0.7827
t441-l2	2457	9.8497	0.1455	0.7807
t441-l3	2460	9.8444	0.1470	0.7809
t441-l4	2424	9.9159	0.1325	0.7861
t401-l1	2004	9.9487	0.1309	0.7815
t401-l2	2666	10.0932	0.1301	0.7797
t401-l3	2696	9.9404	0.1440	0.7753
t401-l4	2424	9.9159	0.1325	0.7861
t437-l1	1923	9.8490	0.1203	0.7835
t437-l2	2073	9.8021	0.1328	0.7856
t437-l3	1842	9.7671	0.1272	0.7759
t437-l4	2424	9.9159	0.1325	0.7861

Although not reported in the main text, the best- and worst- case collections for the t401 typical sized collections were scored on the focus measures, in the hopes that they would shed more light on the trends being observed. Results are shown below:

Best & worst cases

	term-entropy	term-entropy2	retrieved-focus	average-similarity
t401-t1rel	3035	9.8491	0.2073	0.7305
t401-t2rel	2494	9.9058	0.1651	0.6950
t401-t3rel	1924	9.7465	0.1423	0.6820
t401-t4rel	2377	9.5710	0.1752	0.6963
t401-t1wst	2980	9.8527	0.2041	0.7287
t401-t2wst	2462	9.9074	0.1618	0.6935
t401-t3wst	1880	9.7408	0.1367	0.6800
t401-t4wst	2175	9.6110	0.1406	0.7056

D

Appendix: Results of Context Testing

This appendix details the similarity and performance scores obtained by varying the amount of context available in a query, as described in Chapter 6.

CORI Similarities with queries at various context levels

	title	desc	narr	narr (mod)	title-desc
t441-t1	0.8616	0.9173	0.9240	0.9314	0.9173
t441-t2	0.8303	0.9013	0.9206	0.9193	0.9013
t441-t3	0.8160	0.8946	0.9180	0.9154	0.8946
t441-t4	0.6860	0.7807	0.8940	0.8671	0.7807
t401-t1	0.9801	0.8961	0.8356	0.8349	0.8961
t401-t2	0.9392	0.8462	0.8151	0.8190	0.8462
t401-t3	0.9266	0.8374	0.7994	0.8018	0.8374
t401-t4	0.8343	0.7800	0.7702	0.7542	0.7800
t437-t1	0.9735	0.9778	0.9474	0.9685	0.9778
t437-t2	0.9724	0.9779	0.9506	0.9674	0.9779
t437-t3	0.9123	0.9230	0.9241	0.9341	0.9230
t437-t4	0.8550	0.8929	0.8625	0.8472	0.8929

DF Similarities with queries at various context levels

	title	desc	narr	narr (mod)	title-desc
t441-t1	0.0766	0.1290	0.1294	0.1283	0.1061
t441-t2	0.0345	0.0967	0.0966	0.0862	0.0650
t441-t3	0.0345	0.0990	0.0968	0.0865	0.0661
t441-t4	0.0072	0.0235	0.0689	0.0460	0.0151
t401-t1	0.0633	0.0947	0.0604	0.0503	0.0871
t401-t2	0.0587	0.0707	0.0657	0.0605	0.0723
t401-t3	0.0302	0.0651	0.0657	0.0577	0.0522
t401-t4	0.0194	0.0294	0.0451	0.0376	0.0271
t437-t1	0.0867	0.1373	0.0727	0.0828	0.1217
t437-t2	0.0423	0.1054	0.0730	0.0793	0.0786
t437-t3	0.0161	0.0661	0.0480	0.0448	0.0429
t437-t4	0.0289	0.0525	0.0379	0.0311	0.0440

Differences between the t1 and t4 collection, using CORI similarities

	title	desc	narr	narr (mod)	title-desc
Easy	0.1756	0.1366	0.0300	0.0643	0.1366
Med	0.1458	0.1161	0.0654	0.0807	0.1161
Diff	0.1185	0.0849	0.0849	0.1213	0.0849

Differences between the t1 and t4 collection, using DF similarities

	title	desc	narr	narr (mod)	title-desc
Easy	0.0694	0.1054	0.0605	0.0823	0.0910
Med	0.0439	0.0653	0.0153	0.0127	0.0600
Diff	0.0578	0.0848	0.0348	0.0517	0.0777

P20 With Queries at Varying Levels of Context

The following table was used as the basis for the evaluations of how changing the amount of context in a query affects retrieval from a topically-focused collection.

	title	desc	narr	narr (mod)	title-desc
t441-t1	0.6000	0.6500	0.0500	0.4000	0.4500
t441-t2	0.6000	0.6500	0.1000	0.4500	0.6000
t441-t3	0.4000	0.4500	0.1000	0.3500	0.4500
t441-t4	0.1500	0.1500	0.1500	0.1500	0.1500
t401-t1	0.2000	0.4500	0.5500	0.5500	0.4500
t401-t2	0.2500	0.3000	0.5500	0.5500	0.4500
t401-t3	0.3500	0.4500	0.7500	0.6000	0.4500
t401-t4	0.0000	0.0500	0.0500	0.0500	0.0500
t437-t1	0.1000	0.2500	0.1000	0.1000	0.2000
t437-t2	0.0500	0.3000	0.1000	0.1000	0.3000
t437-t3	0.1500	0.1500	0.1000	0.1000	0.1500
t437-t4	0.0000	0.0000	0.0000	0.0000	0.0000

References

Adelberg, B., and Denny, M. 1999. Building robust wrappers for text sources. Technical report, Northwestern University.

Akavipat, R.; Wu, L.-S.; Menczer, F.; and Maguitman, A. 2006. Emerging semantic communities in peer web search. In *Workshop on Information Retrieval in Peer-to-Peer Networks (P2PIR2006)*.

Arens, Y.; Chee, C. Y.; nan Hsu, C.; and Knoblock, C. A. 1993. Retrieving and integrating data from multiple information sources. *Journal on Intelligent and Cooperative Information Systems* 2(2):127–158.

Ashish, N., and Knoblock, C. 1997. Wrapper generation for semi-structured internet sources. In *Workshop on Management of Semistructured Data*.

Baeza-Yates, R., and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. Addison Wesley.

Balabanović, M. 1997. An adaptive web page recommendation service. In *Proceedings of the First International Conference on Autonomous Agents*.

Bergman, M. K. 2000. The deep web: Surfacing hidden value.

Berry, M. W.; Dumais, S. T.; and Letsche, T. A. 1995. Computational methods for intelligent information access. In *Proceedings of Supercomputing 95*.

Billsus, D., and Pazzani, M. 1999. A personal news agent that talks, learns and explains. In *Proceedings of the Third International Conference on Autonomous Agents*.

Bollacker, K. D.; Lawrence, S.; and Giles, C. L. 1999. A system for personalized recommendation of new scientific literature. In *Proceedings of the AAAI-99 Workshop on Intelligent Information Systems*. AAAI Press, Menlo Park, CA.

Bowman, C. M.; Danzig, P. B.; Hardy, D. R.; Manber, U.; Schwartz, M. F.; and Wessels, D. P. 1995.

Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado.

Brandt, D. S., and Uden, L. 2003. Insight into mental models of novice internet searchers. *Communications of the ACM* 46(7):133–136.

Brin, S., and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference (WWW7)*.

Buckley, C., and Walz, J. 1999. The TREC-8 query track. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*.

Budzik, J., and Hammond, K. 2000. User interactions with everyday applications as context for just-in-time information access. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces (IUI2000)*, 44–51.

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

Callan, J.; Connell, M.; and Du, A. 1999. Automatic discovery of language models for text databases. In *ACM SIGMOD Conference*, 479–490.

Callan, J. P.; Lu, Z.; and Croft, W. B. 1995. Searching distributed collections with inference networks. In *Proceedings of the 18th International ACM SIGIR Conference*, 21–28.

Callan, J. 2000. Distributed information retrieval. In Croft, W. B., ed., *Advances in Information Retrieval*, 127–150. Kluwer Academic Publishers.

Chakrabarti, S.; van den Berg, M.; and Dom, B. 1999. Focused crawling: A new approach to topic-specific web resource discovery. In *Proceedings of the 8th International World Wide Web Conference (WWW8)*.

Chen, L., and Sycara, K. 1998. Webmate : A personal agent for browsing and searching. In *Proceedings of the Second International Conference on Autonomous Agents*.

Cheng, I., and Wilensky, R. 1997. An experiment in enhancing information access by natural language processing. Technical Report CSD–97–963, Berkeley.

Craswell, N.; Bailey, P.; and Hawking, D. 2000. Server selection on the world wide web. Technical report, CSIRO – Mathematical and Information Sciences.

CronenTownsend, S.; Zhou, Y.; and Croft, W. B. 2002. Predicting query performance. In *Proceedings of the 25th International ACM SIGIR Conference*.

Davis, J. R. 1995. Creating a networked computer science technical report library. *D-Lib Magazine*.

Dreilinger, D., and Howe, A. E. 1997. Experiences with selecting search engines using meta-search. *ACM Transactions on Information Systems* 15(3).

Engelbart, D. 1962. Augmenting human intellect: A conceptual framework summary report. Technical Report AF 49(638)-1024, Stanford Research Institute.

Fan, Y., and Gauch, S. 1999. Adaptive agents for intelligent information gathering from multiple, distributed information sources. In *AAAI Symposium on Intelligent Agents in Cyberspace*.

French, J. C., and Powell, A. L. 2000. Metrics for evaluating database selection techniques. *World Wide Web* 3(3):153–163.

French, J. C.; Powell, A. L.; Callan, J.; et al. 1999. Comparing the performance of database selection algorithms. In *Proceedings of the 22nd International ACM SIGIR Conference*, 238–245.

French, J. C.; Powell, A. L.; Gey, F.; and Perelmanz, N. 2002. Exploiting manual indexing to improve collection selection and retrieval effectiveness. *Information Retrieval* 5(4):323–351.

Fu, Y.; Ke, W.; and Mostafa, J. 2005. Automated text classification using a multi-agent framework.

In *Proceedings of the Fifth Joint Conference on Digital Libraries (JCDL2005)*, 157–158. Denver, CO: IEEE Computer Society.

Gauch, S.; Wang, G.; and Gomez, M. 1996. Profusion: Intelligent funsion from multiple, distributed search engines. *Journal of Universal Computing* 2(9).

Glover, E. 2001. *Using Extra-Topical User Preferences to Improve Web-Based Metasearch*. Ph.D. Dissertation, University of Michigan.

Golder, S. A., and Huberman, B. A. 2006. The structure of collaborative tagging systems. *Journal of Information Science* 32(2).

Graham, L., and Metaxas, P. T. 2003. "Of course it's true; I saw it on the Internet!". *Communications of the ACM* 46(5):71–75.

Graham-Rowe, D. 2004. Eavesdropping call centre computers cut talk time. *New Scientist*.

Gravano, L., and García-Molina, H. 1995. Generalizing GlOSS to vector-space databases and broker hierarchies. In *Proc. of the 21st International Conference on Very Large Data Bases (VLDB'95)*.

Gravano, L.; Chang, K.; García-Molina, H.; and Paepcke, A. 1997. Starts stanford proposal for internet meta-searching. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*.

Gravano, L.; García-Molina, H.; and Tomasic, A. 1994. Precision and recall of GlOSS estimators for database discovery. In *Proceedings of the third international Conference on Parllel and Distributed Information Systems (PDIS '94)*.

Hagedorn, K. 2005. Looking for pearls of information. *Research Information* 16.

Harman, D. 1992. Overview of the first text retrieval conference (trec-1). In *Proceedings of the Text REtrieval Conference (TREC-1)*.

Hawking, D., and Thistlewaite, P. 1999. Methods for information server selection. *ACM Transactions on Information Systems* 17(1):40–76.

Hawking, D.; Voorhees, E.; Craswell, N.; and Bailey, P. 1999. Overview of the trec-8 web track. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*.

Horvitz, E.; Breese, J.; Heckerman, D.; Hovel, D.; and Rommelse, K. 1998. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 256–265. San Francisco, CA: Morgan Kaufmann.

Hotchkiss, G. 2003. Inside the mind of the searcher. Technical report, Enquiro.

Hutchins, E. 1995. *Cognition in the Wild*. MIT Press.

Ipeirotis, P. G., and Gravano, L. 2002. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th VLDB Conference*.

Kaplan, C.; Fenwick, J.; and Chen, J. 1993. Adaptive hypertext navigation based on user goals and context. *User Modeling and User-Adapted Interaction* 3(3):193–220.

Kleinberg, J. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM* 46.

Kraft, R.; Chang, C. C.; Maghoul, F.; and Kumar, R. 2006. Searching with context. In *Proceedings of the 15th International World Wide Web Conference (WWW2006)*.

Kullback, S.; Keegel, J. C.; and Kullback, J. H. 1987. *Topics in Statistical Information Theory*. Springer Verlag.

Kulyukin, V. A. 1999. Application-embedded retrieval from distributed free-text collections. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, 447–452.

Kushmerick, N.; Weld, D. S.; and Doorenbos, R. 1997. Wrapper inducation for information extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-97)*.

Lagoze, C., and Van de Sompel, H. 2001. The open archives initiative: Building a low-barrier interoperability framework. In *Proceedings of the First Joint Conference on Digital Libraries (JCDL2001)*, 54–62.

Lagoze, C.; Krafft, D.; Cornwell, T.; Dushay, N.; Eckstrom, D.; and Saylor, J. 2006. Metadata aggregation and "automated digital libraries": A retrospective on the nsdl experience. In *Proceedings of the Fourth Joint Conference on Digital Libraries (JCDL2006)*. Chapel Hill, NC: IEEE Computer Society.

Larkey, L. S.; Connell, M. E.; and Callan, J. 2000. Collection selection and results merging with topically organized U.S. patents and TREC data. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM 2000)*.

Lawrence, S. 2000. Context in web search. *IEEE Data Engineering Bulletin* 23(3):25–32.

Leake, D. B., and Scherle, R. 2001. Towards context-based search engine selection. In *Proceedings of the 2001 International Conference on Intelligent User Interfaces (IUI2001)*. ACM Press.

Leake, D., and Sooriamurthi, R. 2004. Case dispatching versus case-base merging: When MCBR matters. *International Journal of Artificial Intelligence Tools* 13(1):237–254.

Leake, D. B.; Bauer, T.; Maguitman, A.; and Wilson, D. C. 2000. Capture, storage and reuse of lessons about information resources: Supporting task-based information search. In *Proceedings of the AAAI-2000 Workshop on Intelligent Lessons Learned Systems*.

Leake, D. B.; Maguitman, A.; and Reichherzer, T. 2005. Exploiting rich context: An incremental approach to context-based web search. In *Modeling and Using Context: 5th International and Interdisciplinary Conference, CONTEXT 2005*, 254–267. Berlin: Springer Verlag.

Lesk, M. 1997. *Practical Digital Libraries: Books, Bytes, and Bucks*. Morgan Kaufman.

Levy, A. Y.; Rajaraman, A.; and Ordille, J. J. 1996. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd Conference on Very Large Databases*.

Liu, K.-L.; Yu, C.; and Meng, W. 2001. Discovering the representative of a search engine. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM 2001)*, 577–579.

Lu, J., and Callan, J. 2005. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *Proceedings of the 27th European Conference on Information Retrieval (ECIR)*.

Lu, Y.; Meng, W.; Shu, L.; Yu, C.; and Liu, K.-L. 2005. Evaluation of result merging strategies for metasearch engines. In *6th International Conference on Web Information Systems Engineering (WISE05)*, 53–66.

Lucene: Frequently Asked Questions Web site. 2006.
http://lucene.sourceforge.net/cgi-bin/faq/faqmanager.cgi.

Manmatha, R., and Sever, H. 2002. A formal approach to score normalization for metasearch. In *Proceedings of the 2002 Human Language Technology Conference*, 88–93.

Marcus, R. S. 1983. An experimental comparison of the effectiveness of computers and humans as search intermediaries. *Journal of the American Society for Information Science* 34(6):381–404.

Mathé, N., and Chen, J. R. 1996. User-centered indexing for adaptive information access. *User Modeling and User-Adapted Interaction* 6(2–3):225–261.

Meng, W.; Liu, K.-L.; Yu, C. T.; Wu, W.; and Rishe, N. 1999. Estimating the usefulness of search engines. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, 146–153. IEEE Computer Society.

Miller, R. 1968. Response time in man-computer conversational transactions. In *AFIPS Conference Proceedings of the Fall Joint Computer Conference*, volume 33, 267–277.

Minsky, M. 1975. A framework for representing knowledge. In Winston, P. H., ed., *The Psychology of Computer Vision*. New York, NY: McGraw-Hill.

Montbriand, J. 1999. Technote 1141: Extending and controlling sherlock. Technical Report 1141, Apple Computer. Available at http://devworld.apple.com/technotes/tn/tn1141.html.

Mostafa, J.; Mukhopadhyay, S.; Lam, W.; and Palakal, M. 1997. A multilevel approach to intelligent information filtering: Model, system, and evaluation. *ACM Transactions on Information Systems* 15(4):368–399.

Nichols, D. 1997. At the event: SIGIR '97. *Ariadne* 11.

Norman, D. A. 1988. *The Psychology of Everyday Things*. Basic Books.

Ogilvie, P., and Callan, J. 2001. The effectiveness of query expansion for distributed information retrieval. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM 2001)*.

Pazzani, M. J., and Billsus, D. 1997. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning* 27:313–331.

Powell, A. L.; French, J. C.; Callan, J.; Connell, M.; and Viles, C. C. 2000. The impact of database selection on distributed searching. In *Proceedings of the 23rd International ACM SIGIR Conference*, 232–239.

Pretschner, A., and Gauch, S. 1999. Ontology based personalized search. In *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, 391–398.

Qin, J.; Zhou, Y.; and Chau, M. 2004. Building domain-specific web collections for scientific digital libraries: A meta-search enhanced focused crawling method. In *Proceedings of the Fourth Joint Conference on Digital Libraries (JCDL2004)*. Tucson, AZ: IEEE Computer Society.

Rhodes, B., and Starner, T. 1996. The remembrance agent: A continuously running automated information retrieval system. In *The Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96)*, 487–495.

Rhodes, B. J. 2000. *Just-In-Time Information Retrieval*. Ph.D. Dissertation, Massachusettes Institute of Technology.

Rhodes, B. J. 2002. Margin Notes: Building a contextually aware associative memory. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces (IUI2002)*, 219–224. ACM Press.

Salton, G.; Wong, A.; and Yang, C. 1975. A vector space model for automatic indexing. *Communications of the ACM* 18:613–620.

Schank, R. C., and Abelson, R. P. 1977. *Scripts, plans, goals, and understanding*. Hillsdale, NJ: Lawrence Erlbaum.

Selberg, E., and Etzioni, O. 1995. Multi-service search and comparison using the metacrawler. In *Proceedings of the 1995 World Wide Web Conference*.

Shannon, C. E. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27:379–423, 623–656.

Spink, A., and Jansen, B. J. 2004. A study of web search trends. *Webology* 1(2).

Sugiura, A., and Etzioni, O. 2000. Query routing for web search engines: Architechture and experiments. In *Proceedings of the 9th International World Wide Web Conference (WWW9)*.

Svenonius, E. 2000. *The Intellectual Foundation of Information Organization*. Cambridge, MA: MIT Press.

Tenenbaum, J. M. 2005. Ai meets web 2.0: Building the web of tomorrow today. Keynote Talk, 22th National Conference on Artificial Intelligence (AAAI-2005).

Tsikrika, T., and Lalmas, M. 2001. Merging techniques for performing data fusion on the web. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM 2001)*, 127–134.

Voorhees, E. M.; Gupta, N. K.; and Johnson-Laird, B. 1995. Learning collection-fusion strategies. In *Proceedings of the 18th International ACM SIGIR Conference*, 172–179.

Voorhees, E. M. 1997. Database merging strategies for searching public and private collections. In *ACM-SIGIR97 Workshop on Networked Information Retrieval*.

Xu, J., and Callan, J. 1998. Effective retrieval with distributed collections. In *Proceedings of the 21st ACM SIGIR Conference*, 112–120. Melbourne, Australia: ACM.

Xu, J., and Croft, W. B. 1999. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd International ACM SIGIR Conference*, 254–261.

Ye, Y., and Fischer, G. 2002. Information delivery in support of learning reusable software components on demand. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces (IUI2002)*, 159–166. ACM Press.

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

Zhao, H.; Meng, W.; Wu, Z.; Raghavan, V.; and Yu, C. 2005. Fully automatic wrapper generation for search engines. In *Proceedings of the 14th International World Wide Web Conference (WWW14)*, 66–75.

Zhu, X., and Gauch, S. 2000. Incorporating quality metrics in centralized/distributed information retrieval on the world wide web. In *Submitted to SIGIR2000*.

Zhu, X.; Gauch, S.; Gerhard, L.; Kral, N.; and Pretschner, A. 1999. Ontology based web site mapping for information exploration. In *Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM'99)*, 188–194.

Curriculum Vitae

Ryan E. Scherle

EDUCATION

Ph.D. Computer Science and Cognitive Science, Indiana University, 2006
Advisors: Dr. David B. Leake and Dr. Michael Gasser

M.S. Computer Science, Indiana University, 1998

B.S. Computer Science, summa cum laude, Rose-Hulman Institute of Technology, 1996
Concentration certificate: Imaging systems

TECHNICAL EXPERIENCE

Software Analyst/Programmer: Digital Library Infrastructure, Digital Library Program, Indiana University, Bloomington, IN, 2005-present. Design and develop a repository for digital collections. Coordinate the design and development of tools to manage collections and deliver them to users, including cataloging tools, presentation tools, and tools for archival storage management.

Software Analyst/Programmer: Digital Music Library, Digital Library Program, Indiana University, Bloomington, IN, 2002-2005. Designed and developed software for storing, searching, and interacting with musical information. Maintain server software, improving its speed, functionality, and stability. Design and integrate changes to the data model. Coordinate development group's documentation and automated quality-control system.

Independent Consultant, Scherle Consulting, Bloomington, IN, 1995-2002. Provided computer support and service for small businesses and organizations. Projects included conversion of data from legacy systems, creation of custom database systems, conducting surveys, and selection of business software.

Web Developer, Jasper Communications, Jasper, IN, 1996-1997. Designed and implemented commercial websites, including retail sales sites and a real-time weather tracking system.

NOVELL Manager, Waters Computing Center, Terre Haute, IN, 1993-1996. Installed network operating systems and applications. Maintained public computing labs and faculty computers. Trained three network administrators and over 60 help desk operators.

TEACHING EXPERIENCE

Instructor and Course Developer, Introduction to Computing, 1996-1997, 2000-present. Taught 90 students computer applications in hands-on labs. Taught over 600 students from a wide variety of backgrounds via distance learning.

Instructor, Introduction to Programming, 2002. Taught 130 students basic programming techniques in Java. Supervised seven associate instructors.

Departmental Tutor, various courses, 1998-2001. Provided individual tutoring to Computer Science students. Assisted students with physical disabilities and learning disabilities.

Instructor, Discrete Structures for Computer Science, 1998. Taught 40 students mathematical techniques used in computer science. Supervised one associate instructor.

Associate Instructor, Introduction to Computer Science, 1997-1998. Assisted in teaching and grading for a course on basic programming techniques.

PUBLICATIONS

Looking for a Haystack: Selecting Data Sources in a Distributed Retrieval System. Ryan Scherle. Ph.D. Dissertation, Indiana University. 2006.

Variations2: Retrieving and Using Music in an Academic Setting. Jon W. Dunn, Donald Byrd, Mark Notess, Jenn Riley, and Ryan Scherle. Communications of the ACM. 49 (8). 2006.

The Anatomy of a Bibliographic Search System for Music. Ryan Scherle and Donald Byrd. In *Proceedings of the 2004 International Conference on Music Information Retrieval (ISMIR 2004)*.

V2V: A Second Variation on Query-by-Humming. William P. Birmingham, Kevin O'Malley, Jon W Dunn, and Ryan Scherle. In *Proceedings of the 2003 Joint Conference on Digital Libraries (JCDL2003)*.

Introduction to Computing, course learning guide. Rod Stark and Ryan Scherle. Indiana University Independent Study Program. 2002.

Towards Context-Based Search Engine Selection. David B. Leake and Ryan Scherle. In *Proceedings of the 2001 Conference on Intelligent User Interfaces (IUI2001)*.

Selecting Task-Relevant Sources for Just-in-Time Retrieval. David Leake, Ryan Scherle, Jay Budzik, and Kristian Hammond. In *Proceedings of the AAAI-99 Workshop on Intelligent Information Systems*, 1999.

PRESENTATIONS

Searching Digital Collections via SRU. Ryan Scherle and Randall Floyd. Indiana University Digital Library Brown Bag Series. October, 2006.

A Fedora Architecture to Support Diverse Collections. Ryan Scherle and Jon W. Dunn. Fedora Users Conference 2006.

Reexamining Digital Library Infrastructure at IU. Jon W. Dunn, Ryan Scherle, and Eric Peters. Indiana University Digital Library Brown Bag Series. November, 2005.

Flipping the Switch: Lessons Learned from a Major Digital Library Migration Project. Jon W. Dunn, Ryan Scherle, and Mark Notess. Digital Library Federation Fall Forum, 2005.

PROFESSIONAL MEMBERSHIPS

American Association for Artificial Intelligence, 1997-present
Association for Computing Machinery, 1992-present

PROFESSIONAL SERVICE

Reviewer, *AI Magazine*, 2002, 2006
Reviewer, International Conference on Intelligent Information Technology, 2002
Reviewer, Conference on Intelligent User Interfaces, 2002
Reviewer, International Conference on Case-Based Reasoning, 2001
Associate Editor, *Van Nostrand's Scientific Encyclopedia*, 2000
Reviewer, *Journal of the Learning Sciences*, 2000
Volunteer, National Conference of the American Association for Artificial Intelligence, 1998, 1999
Volunteer, Second International Conference on Case-Based Reasoning, 1997

SELECTED HONORS AND AWARDS

Fellowship, US Department of Education, Graduate Assistance in Areas of National Need (GAANN), 1998-2002
Scholarship, American Association for Artificial Intelligence (AAAI), 1998, 1999
Honorable Mention, National Science Foundation Graduate Fellowship Program, 1997
Scholarship, International Conference on Case-Based Reasoning (ICCBR), 1997
Scholarship, Cognitive Science Department, Indiana University, 1996, 1997
Addison-Wesley Book Award, Rose-Hulman Institute of Technology, 1996
Microsoft Senior Award, Rose-Hulman Institute of Technology, 1996
Presidential Scholarship, Rose-Hulman Institute of Technology, 1992-1996
Member, Upsilon Pi Epsilon (Computer Science honorary)
Member, Pi Mu Epsilon (Math honorary)
Member, Iota Nu Phi (Informatics honorary)

DEPARTMENTAL SERVICE

President, Computer Science Graduate Student Association, 1999-2000
Secretary, Computer Science Graduate Student Association, 1996-1997
Member, Departmental Graduate Admissions Committee, 2001
Member, Departmental Undergraduate Curriculum Committee, 1999-2000
Graduate Student Mentor, 1997-1999
President, Student Chapter of the Association for Computing Machinery, 1995
Coordinator, Midwest Invitational Programming Contest, 1995

COMMUNITY SERVICE

Volunteer Coordinator, Computer Tutoring Project, Area 10 Agency on Aging, 1997-2000. Taught computing techniques to senior citizens. Managed a team of 15 volunteer tutors.

Member, Secretary, President, Tau Lambda chapter of Alpha Phi Omega, National Service Fraternity, 1993-1996. Presided over a chapter with 45 members. Wrote and presented a bid to host Regional conference. Participated in a variety of service projects.

TECHNICAL SKILLS

Languages: Java, C/C++, SQL, Visual Basic, Perl, PHP, Scheme, Ada, Common Lisp, Pascal, Various Assembly Languages, XML
Operating Systems: Windows, Linux, Unix, OS X, AIX, NOVELL, MS Dos, NeXTStep, VMS
Databases: DB2, Microsoft Access, PGSQL, SQL Server