Server characterisation and selection for personal metasearch

Paul Thomas

A thesis submitted for the degree of
Doctor of Philosophy of
The Australian National University

May 2008

Except where otherwise indicated, this thesis is my own original work.

Paul Thomas
27 May 2008

My PhD programme has resulted in several publications:

- **Thomas, P.**, and Hawking, D. Evaluating sampling methods for uncooperative collections. In *Proc. ACM SIGIR* (2007). This paper describes a number of methods for generating random samples and evaluates each on a number of collections. It also introduces a new sampling technique, "multiple queries", which produces samples of similar quality to the best current techniques but with significantly reduced cost. This work forms the basis of Chapter 4.

- **Thomas, P.**, and Rowlands, T. Estimating the value of automatic disambiguation. In *Proc. ACM SIGIR* (2007). Poster. A common motivation for personalised search systems is the ability to automatically disambiguate queries. Following analysis of log files from three search providers, this paper suggests that automatic disambiguation is of use in only limited situations. Some of this work is included in Section 3.2.

- **Thomas, P.**, and Hawking, D. Evaluation by comparing result sets in context. In *Proc. CIKM* (2006). This paper considers established evaluation techniques for personal metasearch, personalised search, or enterprise search and introduces a new technique which offers real judgements in context without significant overhead. Experiments demonstrate the effectiveness of the method. This work forms the basis of Chapter 8.

- Hawking, D., and **Thomas, P.** Server selection methods in hybrid portal search. In *Proc. ACM SIGIR* (2005). Exhaustive metasearch methods are not appropriate in some real-world Web systems. This paper argues instead for a hybrid metasearch system, and evaluates server selection methods for such a situation. Two new methods, HARP and AWSUM, are introduced and perform well compared with established methods. This work is continued, with more methods and a different evaluation framework, in Chapter 7. Hybrid metasearch models are discussed in Section 2.1.5.

- Wu, M., **Thomas, P.**, and Hawking, D. TREC 14 enterprise track at CSIRO and ANU. In *Proc. TREC* (2005). This paper reports experiments carried out in the TREC enterprise track by the CSIRO/ANU group in 2005. Email structure was used in known-item and discussion search tasks. Although email search remains an important open area, this thesis concentrates on a metasearch framework and these experiments are not reported here.

The experiments reported in Chapters 4 and 8 were run in collaboration with David Hawking. In each case the experimental work, including software and analysis, were original with me. The new sampling technique of Chapter 4 was also original.

Acknowledgements

I am most grateful for the advice, support, and encouragement of my supervisor, David Hawking. His broad knowledge, technical expertise, and continued good humour has made this work possible. Thanks too to Peter Christen and Tom Gedeon, also on my panel, for their support and input.

Colleagues at the CSIRO, the ANU, and elsewhere have provided sound advice and useful conversations; I would especially like to thank Tom Rowlands, Peter Bailey, Ross Wilkinson, Tim Tang, Tim′ Jones, Alex Krumpholz, Mark Baillie, Leif Azzopardi, and Milad Shokhoui for their input. Thanks also to the participants in the "IR and friends" series for their enthusiasm.

The libraries of the Australian National University and the CSIRO, and the Statistical Consulting Unit at the Australian National University, have provided a good deal of support.

I am grateful for the help of the approximately 50 people who participated in the survey of Section 3.3, and those who distributed it; the approximately 100 people who participated as test users in experiments of Chapter 8; and the users who participated in the experiments of Chapter 9.

Additional financial and in kind support during this project was provided by the CSIRO.

Abstract

A single search interface to all a person's digital resources, such as email archives, corporate databases, websites, and subscription services, is appealing but a central index of all private, corporate, subscription and web data is impractical. A metasearch approach can instead integrate any number of existing search services over a variety of data.

This thesis advocates and examines *personal metasearch*, or metasearch over a user's entire set of digital resources. Metasearch has been well studied in other environments, but has not before been considered with this range of resources; therefore several aspects are re-examined in this new application. Experiments in document sampling, collection size estimation, language modelling, and server selection, all important subproblems in metasearch, demonstrate that established techniques which work well in traditional settings do not necessarily operate well over the wide range of resources in personal applications.

Many techniques for sampling documents from a collection are biased, especially towards longer documents; other metasearch subproblems often rely on unbiased samples and their performance is adversely affected. A new technique for generating samples is therefore proposed and evaluated, and results indicate improvements in sample quality.

Techniques for collection size estimation, language modelling, and server selection are also investigated in a personal metasearch framework. Several techniques prove inappropriate or have been over-fitted in earlier work, but some appear useful. In each case, performance is improved with better-quality samples of documents as input.

Finally, standard evaluation techniques are a poor fit to the personal metasearch environment, and this thesis proposes a new method based on a functioning search tool inserted into the natural retrieval process. This allows study of real information needs, works with dynamic and/or private collections, and records judgements in their full context. It has been validated in a number of experiments and used with a working personal metasearch tool to compare methods for server selection.

Contributions of this thesis include the first analysis of personal metasearch, from a theoretical basis and from studies of potential users; a new algorithm for document sampling which is better able to operate over the wide variety of data sources found in this application; an evaluation of a number of metasearch algorithms and an analysis of common failures; an evaluation technique suited to personal and dynamic collections; and a platform for further research.

Contents

Acknowledgements **vii**

Abstract **ix**

1 Introduction **1**

1.1 Thesis outline 2

2 Metasearch **5**

2.1 Models for search 5

2.2 Implementation 11

2.3 Operational systems 14

2.4 Summary 16

3 Personal metasearch **19**

3.1 Alternatives 19

3.2 Personalisation 21

3.3 A survey of information sources 22

3.4 PIS, a personal information searcher 26

3.5 Research problems in personal metasearch 27

3.6 Summary 28

4 Server characterisation: sampling documents **29**

4.1 Notation 30

4.2 Related work 30

4.3 Multiple queries 37

4.4 Sampling cost 38

4.5 Sampling experiments 39

4.6 Results 43

4.7 Conclusions 49

5 Server characterisation: size **51**

5.1 Related work 52

5.2 Size estimation experiments 60

5.3 Results 61

5.4 Conclusions 72

6 Server characterisation: subject matter and language 73
6.1 Related work . . . 73
6.2 Language modelling experiments . . . 78
6.3 Results . . . 83
6.4 Conclusions . . . 89

7 Server selection 91
7.1 Related work . . . 92
7.2 Selection experiments . . . 112
7.3 Results . . . 117
7.4 Conclusions . . . 129

8 Evaluating personal metasearch 131
8.1 Established approaches . . . 131
8.2 Embedded comparisons . . . 142
8.3 Validating the design . . . 146
8.4 Case study . . . 153
8.5 Applications . . . 154
8.6 Conclusions . . . 154

9 Evaluating server selection 157
9.1 Experiment . . . 157
9.2 Results . . . 159
9.3 Observations . . . 160
9.4 Conclusions . . . 161

10 Summary and conclusions 163
10.1 Summary of findings . . . 163
10.2 Building a personal metasearch tool . . . 165
10.3 Future work . . . 166
10.4 Overall conclusion . . . 167

A Notation and terminology 169
A.1 Notation . . . 169
A.2 Terminology . . . 171

B Instruments for searcher survey 173

C Instruments for evaluation experiments 177
C.1 Demographic questions . . . 178
C.2 First experiment . . . 179
C.3 Later experiments . . . 180

D Instruments for server selection evaluation 181
D.1 Sorting with SM-TSS . . . 183

E Software 185
E.1 Interface 185
E.2 Modules 185

Bibliography 191

List of figures

2.1 Models for search 6
2.2 A metasearch process 12

3.1 Information sources available to an individual 20
3.2 Two approaches to personalising search tools 21
3.3 PIS, a *p*ersonal *i*nformation *s*earcher. 27

4.1 Calculations for pool-based sampling 34
4.2 Email samples for test "S" 46

5.1 The Schumacher-Eschmeyer (capture history) technique 55
5.2 Errors in size estimates, given random samples 63
5.3 Errors in size estimates, with and without the correction of Shokouhi et al. 66
5.4 Improvement in size estimates with more interactions 67

6.1 Quality of models 84
6.2 Improvement in quality measures with more documents 86
6.3 Improvement in quality measures with more documents from multiple queries samples 87
6.4 Improvement in quality measures with more documents from query-based samples 88

7.1 Similarity under the high correlation scenario 95
7.2 Example calculations for ReDDE 105
7.3 Performance of server selection algorithms 119
7.4 Correlation of collection rankings with size-based ranking 120
7.5 Queries for which .GOV is the best collection 122
7.6 Queries for which calendar is the best collection 123
7.7 Performance of server selection algorithms, given models from best samples 125
7.8 Performance of server selection algorithms, given models from typical samples 126
7.9 Correlation between measures of model quality and selection performance 128

8.1 Sample two-panel interfaces 143
8.2 Extra feedback from the two-panel interface 144

9.1 Server selection for each panel in PIS 158

D.1 Logged data from PIS 183
D.2 Reviewing logged data in PIS 183
D.3 The SM-TSS algorithm 184

E.1 The PIS software 186
E.2 PIS with two panels 186
E.3 Extra feedback from PIS 187

List of tables

4.1 Characteristics of sampling methods 31
4.2 Collections used in the experiments 40
4.3 Metasearch testbeds in the literature 41
4.4 Estimated cost for each sampler 43
4.5 Email samples for test "T" 44
4.6 χ^2 results for test "T" 45
4.7 χ^2 results for test "S" 45
4.8 Summary of sampling methods 50

5.1 Characteristics of size estimation methods 52
5.2 Size estimates compared, using random samples 64
5.3 Size estimates compared, using the multiple queries sampler 69
5.4 Size estimates compared, using the QBS sampler 70

6.1 Term frequencies extracted from the email collection 80
6.2 Document frequencies extracted from the email collection 81

7.1 Characteristics of server selection methods 93

8.1 User demographics for first evaluation experiment 147
8.2 User demographics for second evaluation experiment 149
8.3 User demographics for third evaluation experiment 151
8.4 User demographics for fourth evaluation experiment 152
8.5 User demographics for evaluation case study 153
8.6 Summary of evaluation validation experiments 155

9.1 Inferred preferences for server selection 160
9.2 Inferred preferences with alternative indicators 161

Chapter 1

Introduction

Many people routinely access information from several different computer-based resources, such as email archives, corporate databases, local and public websites, and subscription services. Typically, each resource has its own search tool, each with a distinct interface and set of capabilities. This multiplication of tools means that a person who wants to find some information which could be in any one of these sources — or which may be scattered amongst several — faces increased work and is exposed to certain types of error.

This thesis considers one solution, a *personal metasearch* system which integrates arbitrary combinations of existing search tools. In metasearch, a single tool provides a unified interface to a number of otherwise independent search servers, forwarding a user's query to each of them and collating result sets. Each server can operate normally, with its own index and performing any local optimisations as needed, and copies of documents are not necessarily needed by the metasearch tool.

An effective metasearch system raises interesting research questions in several areas including source discovery, characterisation, selection, query translation, result merging, and presentation. This thesis considers the subproblems of *server characterisation*, or inferring parameters such as the size and subject matter of each resource, and *server selection*, the problem of identifying the resources most likely to be of use for a given query. While both problems have been well studied in other environments, they have not before been considered with the range of resources likely to be used in personal metasearch. This thesis presents the first work in this setting.

Experiments reported here consider document sampling, collection size estimation, language modelling, and server selection and demonstrate that established techniques which work well in traditional settings do not necessarily operate well over the range of resources in personal applications. These experiments also confirm the interdependence of each subproblem; poor-quality document samples, for example, provide poor-quality size estimates and then poor language models and server selection in turn. Despite this, several techniques show promise and it has been possible to build and test a personal metasearch tool.

Evaluating personal metasearch tools or algorithms also presents a new set of challenges, and this thesis suggests and validates a new evaluation technique which is appropriate for applications such as personal metasearch which include private or personal documents, fast-changing collections, and in which personal context plays an important role. Experiments with a working personal metasearch tool and this novel evaluation technique demonstrate it can be used to compare metasearch algorithms *in situ* and that the results of these comparisons broadly, but not entirely, agree with those from test collections.

1.1 Thesis outline

The remainder of this thesis is arranged as follows. Since there is little overlap between the literature on, for example, server characterisation and system evaluation, related work is considered in each chapter as appropriate.

Metasearch, in contrast to other problems in information retrieval, assumes retrieval tasks are carried out over multiple independent collections. Chapter 2 dicusses this and introduces five models od the retrieval process. Consideration of these suggests a hybrid metasearch model, with direct access to some collections and only indirect access to others, is most feasible for operational tools. Chapter 2 also outlines a process for metasearch, including the important steps of server characterisation and selection.

Previous applications of metasearch have operated over sets of public collections, and over documents which are more or less homogeneous. Chapter 3 suggests an alternative application, *personal metasearch*, which is the focus of this thesis. This chapter reports on a survey of computer users which confirms that people make use of a great many information sources, with a wide variety of characteristics, and indicates that a personal metasearch tool could be of real benefit.

For a tool to be of most use, however, several outstanding questions need to be considered. This thesis considers two: server characterisation and server selection. Chapters 4 to 6 consider server characterisation, the process of determining attributes such as the number of documents in each collection and the range of subjects covered; and Chapter 7 considers server selection, the process of choosing the server or servers most likely to be useful in answering a given query.

An essential first step in most characterisations is collating one or more samples of documents from each collection. These samples are generally assumed to be randomly selected, but random selection is a challenge without the cooperation of servers. Chapter 4 considers a range of techniques for generating samples, without assuming server cooperation, and for the first time evaluates each using two tests for bias. This chapter also introduces a new testbed, more appropriate for personal metasearch techniques than those used to date. Although all sampling techniques appear biased, there is a good deal of variation and a technique proposed here, "multiple queries sampling", seems promising for the range of collections likely in personal metasearch.

Chapter 5 discusses techniques for estimating the size of collections. Size is an important characteristic in its own right, as a proxy for coverage and completeness, and is also used in language modelling and server selection. Several algorithms have been proposed for estimating collection size in a metasearch environment, and five of these are tested. No single size estimation technique works well in all cases, but broad patterns suggest some might be useful for a working tool. The quality of size estimates is also shown to depend upon the quality of the document samples used as input; in particular, those algorithms which produce more biased samples lead in turn to poorer size estimates. Estimates using samples from the multiple queries algorithm, however, are largely indistinguishable from those using perfectly random samples.

Chapter 6 concludes work on characterisation by considering techniques for inferring the subject matter and language of collections. Two main approaches are considered: classifications into topic hierarchies and simple language modelling. Language models are examined further in a series of experiments, where the quality of an inferred model is again shown to depend on the quality of the document samples given as input. Again, models based on the multiple queries sampler are largely indistinguishable from those based on true random samples. Three measures of model quality are also investigated.

Using the samples, size estimates, and models from previous chapters, Chapter 7 discusses the problem of server selection: choosing the collection, or collections, most likely to be useful for a given query. A large number of techniques have been proposed for this task, and again a number of these are tested in a personal metasearch context. Many methods are shown to be biased toward larger collections, regardless of utility, but a method based on Kullback-Leibler divergence performs well in a variety of circumstances. Once again, however, the performance of selection techniques depends crucially on the quality of the samples, size estimates, and models used as input.

With techniques for server characterisation and selection it is possible to build a personal metasarch tool. Having built such a tool, however, questions remain: how can it be evaluated? How can one choice of algorithm or parameter be compared with another? Experiments in Chapters 4 to 7 make use of a fixed testbed and invented queries, which are assumed to be representative of real use. This is a standard technique, but in Chapter 8 a survey of this and other options suggests none are a good match for the personal metasearch environment. An alternative technique, which embeds evaluation in a working search tool, is proposed instead and tested in a series of experiments. These experiments demonstrate that the technique is able to distinguish high- from low-quality result sets, under a variety of circumstances, and that it can do so with minimal overhead for users or experimenters while still allowing for a user's full context, real information needs, and private or fast-changing collections. This technique is used in Chapter 9 to evaluate two of the server selection methods considered earlier, and this case study confirms that the embedded evaluation technique is practical and can produce useful comparisons.

Finally, Chapter 10 summarises the work of previous chapters and suggests directions for future research.

Appendix A summarises the notation and terminology used in this thesis, and Appendices B to D have details of the instruments used in survey and evaluation experiments. The software used in Chapters 8 and 9 is described in Appendix E.

Chapter 2

Metasearch

Computer users typically have access to a variety of data sources, covering different subjects and with different methods for access [Barreau and Nardi 1995; Teevan et al. 2004]. This chapter considers techniques for information retrieval in these multiple--source situations.

Section 2.1 describes five models for search over multiple collections. Two traditional designs, the "multiple services" and "central index" models, are commonly implemented but are, respectively, prone to error and not generalisable. Metasearch models, which provide a single interface to a number of search tools, are more promising, and Section 2.1.5 argues for a "hybrid metasearch" model for many applications. Section 2.2 discusses implementation issues for metasearch tools, including a summary of each step in the process, and Section 2.3 considers existing metasearch tools and related work.

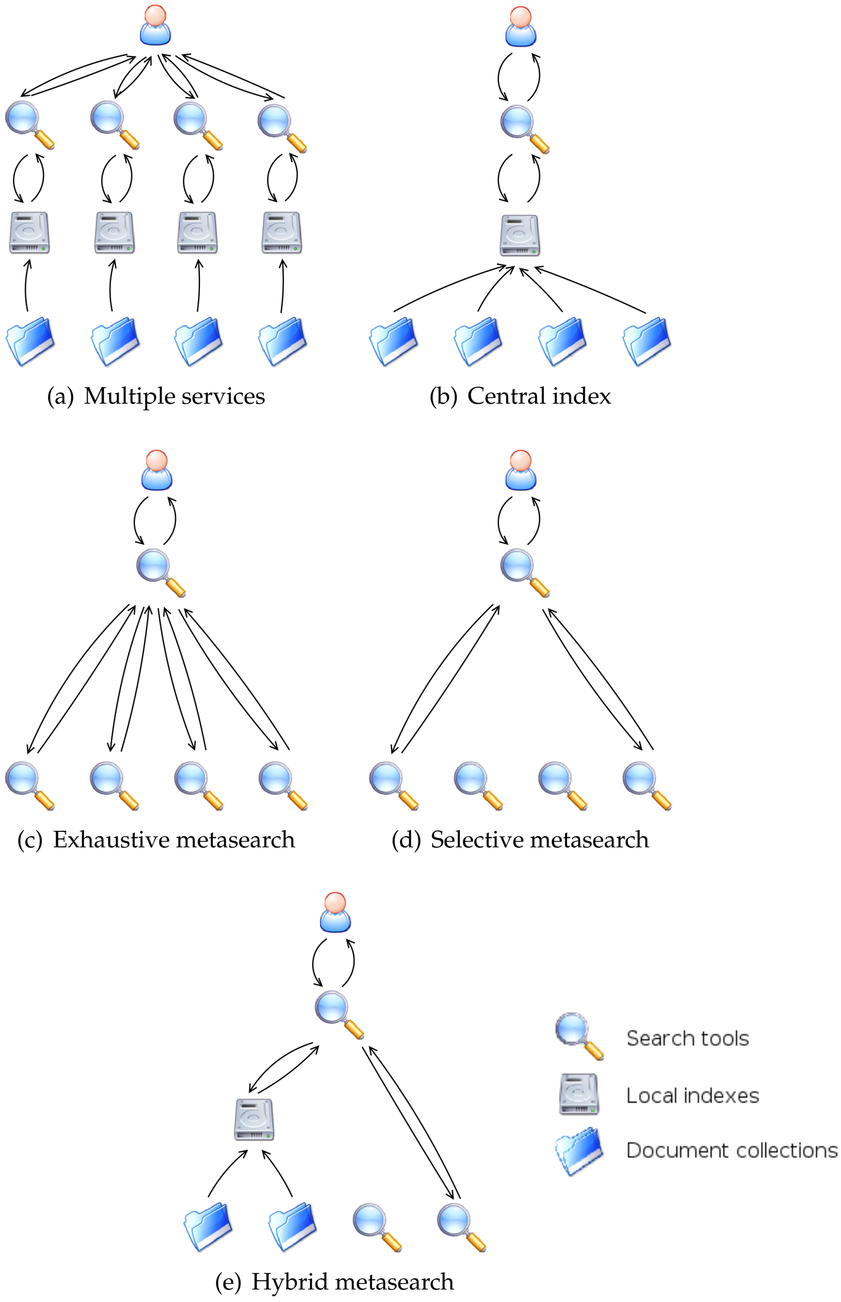
In this discussion, a *document* is considered the basic unit of retrieval. This may be, for example, a piece of email; a posting to an online forum; a web page;[1] or a single record from a database. A *collection* is a set of documents held at the same repository, such as all archived email or a complete database. *Servers* are applications, local or on the network, which provide search capabilities for a collection: for example, an email application's "search" function or a database front-end. Section A.2 summarises this and other terminology used in this thesis.

2.1 Models for search

There are several feasible models for search services which span more than one data source. Extreme cases are multiple services, a central index, and exhaustive metasearch; other models are selective metasearch or a hybrid approach (Figure 2.1).[2] In this section, each model is described and implications considered for quality, speed, and cost of search.

[1] This thesis follows the convention of using "web" for any set of documents mainly using HTTP [Fielding et al. 1999] and HTML [Raggett et al. 1999], including local or private webs, and capitalised "Web" for the publicly-available World-Wide Web.

[2] The terminology in this section is based on that of Craswell et al. [2004].


(a) Multiple services

(b) Central index

(c) Exhaustive metasearch

(d) Selective metasearch

(e) Hybrid metasearch

Figure 2.1: Models for search. Arrows down the page represent queries; arrows up the page represent sets of documents.

2.1.1 Multiple services

The "multiple services" model (Figure 2.1(a)) is the simplest of the five. In this model, there is a one-to-one correspondence between collections and search tools. Each collection is indexed separately, and each index has its own search tool with its own interface. This represents the status quo for most users and collections: for example, on a typical PC there may be individual tools (and indexes) for email, web search, file search, and any databases. This is further discussed in Section 3.3 following.

Although simple to implement and well-understood, this model imposes a significant burden on users and allows certain types of errors:

Cognitive burden If each collection presents its own search interface, a user with access to more than one collection must choose amongst different tools to find information they need. For each search task, they must decide where the required information is likely to be held; launch the appropriate tool for that collection; translate their information need into an appropriate query, or sequence of interface actions, for the tool; and scan the returned result set for the information they need. If the need is not satisfied after an initial query and the user does not give up, he or she must either reformulate the query to search the same collection or, worse, repeat the entire process with a different collection.

The load imposed by this model is significant: as well as the burden of remembering how to operate a separate tool for each collection, and the capabilities and features of each, a user must decide which collection is the most appropriate for each information need.

Errors The requirement that the user choose a collection within which to search makes certain types of error possible.

The first may occur when a collection is chosen and searched, and no relevant information is returned; a user may give up at this point unaware that there is relevant information in one or more other collections and that a second search could return it. The second type of error made possible by the multiple services model occurs when a user chooses a collection, searches it, and finds apparently relevant information; however, unbeknownst to the user, more correct or more up-to-date information may be available from another collection.

These errors are mitigated by a model that subsumes several collections into a single search tool. Four such models are discussed below.

2.1.2 Central index

An obvious way to search multiple collections is to make a local copy of each collection ahead of time, building a local index, and use this index to respond to queries as illustrated in Figure 2.1(b). This is the model employed by some existing systems, such as the FirstGov portal,[3] which searches several independent US government web

[3] `http://www.first.gov/`

sites, and the Australia.Gov portal,[4] which does the same for the Australian goverment. This can provide a single search tool, avoiding the errors and load of multiple services; in addition, any duplicate documents can be eliminated. However, there are often constraints on copying ("crawling", in web systems). Further, the cost is significant, local optimisations may be lost, and the time taken to copy may result in a slow response to changes in a collection. If copying is carried out frequently to maintain freshness, significant load may be imposed on the constituent document servers and networks.

Constraints on copying Most significantly, it will not be possible to copy a number of important collections. Access to some collections is charged for every document retrieved; others will be crippled by copyright or restrictions on usage. A further case, common to many collections, is that there is no way to enumerate all documents: the only access is through a search tool without browsing capability. Many online databases work on this model, including for example the popular CiteSeer[5] and PubMed[6] collections. Evidently, if a collection cannot be reliably copied then a central index will be impractical.

Cost of copying A major problem faced by a copying search tool is the network traffic generated by large-scale copying. Since some overhead is incurred in copying (network protocol headers, duplicate documents, etc.), the traffic generated will exceed the total size of real data. Craswell et al. [2004] estimate as much as 70% overhead in a web crawl, and it is reasonable to expect similar figures for other network protocols.

Incremental copying, where only small parts of a collection are copied at a time, can reduce instantaneous network load considerably but the traffic generated for an entire copy is unchanged. A number of other methods are available for reducing network traffic or copying time [Baeza-Yates and Castillo 2007; Castillo et al. 2004; Craswell et al. 2004; Edwards et al. 2001]; however even with these techniques the scale of some document collections (such as the Web) means copying would require considerable resources and would certainly be beyond the reach of either a personal or an enterprise tool.

Staleness *Staleness* is introduced when a local index is out-of-date with respect to a collection. Result sets returned from a stale index may miss recent additions to a collection, may not make best use of the available information, or may include documents no longer in the collection and therefore inaccessible; the chance of staleness will increase as the collection is updated more frequently relative to the index. Systems which maintain an index may choose to update the index more frequently, incurring more cost and reducing staleness, or less frequently, saving some cost at the

[4] `http://govsearch.australia.gov.au/search/search.cgi?collection=gov_combined&form=au`

[5] `http://citeseer.ist.psu.edu/`

[6] `http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=pubmed`

expense of staleness; they may also use document characteristics to estimate an appropriate update frequency [Cho and Garcia-Molina 2000; Cho and Garcia-Molina 2003; Craswell et al. 2004; Fetterly et al. 2003], but some staleness is inevitable.

Local optimisations Since individual servers can be expected to know something about the collections they index and the users they serve, they can offer appropriate local optimisations. For example, they may translate terms ("law" to "act" for a legislation search, or "exhaust" to "emissions" for a search of environmental information) or rank results according to frequency of use [Agichtein et al. 2006; Freyne et al. 2004; Joachims 2002b]. This specialised knowledge will be lost if the collection is copied.

2.1.3 Exhaustive metasearch

The problems with multiple tools and a central index suggest a *metasearch* model.[7] In metasearch, a single tool (sometimes called a "broker" or "receptionist") provides a unified interface to a number of otherwise independent search servers, forwarding a user's query to each of them and collating result sets. Each server can operate normally, with its own index and performing any local optimisations as needed, and copies of documents are not necessarily needed by the metasearch tool.

In the "exhaustive" metasearch model of Figure 2.1(c), each user query is translated and submitted to each server, and result sets from all servers are merged before being presented to the user. Since no copying is needed and each server can operate normally, this avoids the problems of copying and of local optimisations described above. Significant difficulties, however, exist with scale and availability.

Network bandwidth, availability, and response time An obvious problem is the cost, in network traffic, of forwarding a user's query to each server and receiving the replies. Following Craswell et al. [2004], the network traffic generated is $(|q| + |r|)(|\mathcal{S}| + 1)$, where $|q|$ and $|r|$ are the size of a query and response and $\mathcal{S}$ is the set of servers interrogated. (The extra query and response is the traffic from and to the user.) Using figures of 1kB for $|q|$ and 20kB for $|r|$, with even five servers being searched 126kB of network traffic is required to present a result list to the user, and this grows with the number of servers known to the system. The traffic can be reduced by limiting the number of results retrieved from each server [Lempel and Moran 2002], but the approach rapidly becomes infeasible for even the best-connected hosts.

Final results can be returned to the user only after every search engine has returned its result list (or been timed out, with consequent loss of effectiveness). This means the user is likely to experience significant delays, and the system as a whole will often be faced with unavailability of one or more servers.

Interface wrappers Since we cannot in general assume that each server provides an identical search interface, we will need to provide a wrapper for each interface

[7] Metasearch is also referred to as "distributed information retrieval" — for example by Baumgarten [1999] and Callan et al. [1995] — and "federated search", for example by Avrahami et al. [2006].

— potentially each server — which converts the user's query, feeds it to the search interface, and extracts search results. These wrappers can be generated automatically in some circumstances [Perkowitz et al. 1997; Raghavan and Garcia-Molina 2001], but generating and maintaining wrappers for a large number of servers represents a good deal of work [Avrahami et al. 2006].

Availability of search interfaces Not all collections are associated with a search server; evidently, if a search interface is not available for a collection then that collection cannot be included in an exhaustive metasearch tool. Experiments by the author suggest that in a large set of Web sites, only 31% of collections had a search interface [Hawking and Thomas 2005]. The survey of search users reported in Chapter 3 also provides evidence that search functions, while widespread, are not universal.

2.1.4 Selective metasearch

A well-studied, although little-used, alternative to exhaustive metasearch is the more selective model of Figure 2.1(d). In this model, a metasearch tool forwards each user query to a subset of the available servers, chosen for their likely utility. The metasearch tool must maintain adequate data to inform the selection, but in exchange can dramatically reduce the time and traffic needed to respond to a query. Selection methods are discussed in Section 2.2.2 below and in Chapter 7.

There is some evidence [Abbaci et al. 2002; Hawking and Thistlewaite 1999; Powell et al. 2000; Xu and Croft 1999] that a selective metasearch model can provide better search quality than a central index, given a "very good" server selection algorithm, although the "relevance-based ranking" used in these studies and by French et al. [1999] uses prior knowledge of which documents are relevant.

The selective metasearch model, with a strict one-to-one correspondence between servers and collections, is generally assumed in studies of server selection [French et al. 1998; French et al. 1999; Powell and French 2003; Rasolofo et al. 2001]. However, as it stands it is unlikely to be appropriate for searching all of a user's collections: to work as required, every collection must have an associated server and a wrapper must be maintained for each. Selective metasearch can be incorporated into a hybrid model to accommodate this.

2.1.5 Hybrid metasearch

Both exhaustive and selective metasearch seem to be impractical for real-world tools. An alternative model suggested by Craswell et al. [2004], pictured in Figure 2.1(e), accommodates constraints on server availability. Some or all servers are considered candidates for metasearch, and hence for server selection, and other collections are copied and indexed locally. To answer a user's query, a number of servers are selected (as in selective metasearch) along with one or more collections from the local index. Since larger collections are more likely to provide a search interface [Hawking and

Thomas 2005], the volume of data copied can be substantially reduced and staleness is less of a concern.

Hybrid search can be considered a special case of selective metasearch, except for the consideration of copying collections to local storage; in this special case the metasearcher has full information for one or more collections, and almost zero cost of lookup and retrieval. In this thesis therefore the selective model is assumed. For simplicity, this thesis also assumes a one-to-one correspondence between collections and servers; for those collections in the hybrid model which are copied locally, a "server" may just be a part of the metasearcher itself.[8]

2.2 Implementation

Any working metasearch implementation must make some assumptions about the environment it works in, and in particular about the search servers included. Any implementation will also follow the same general process: the description below draws on and expands that in Cope et al. [2003].

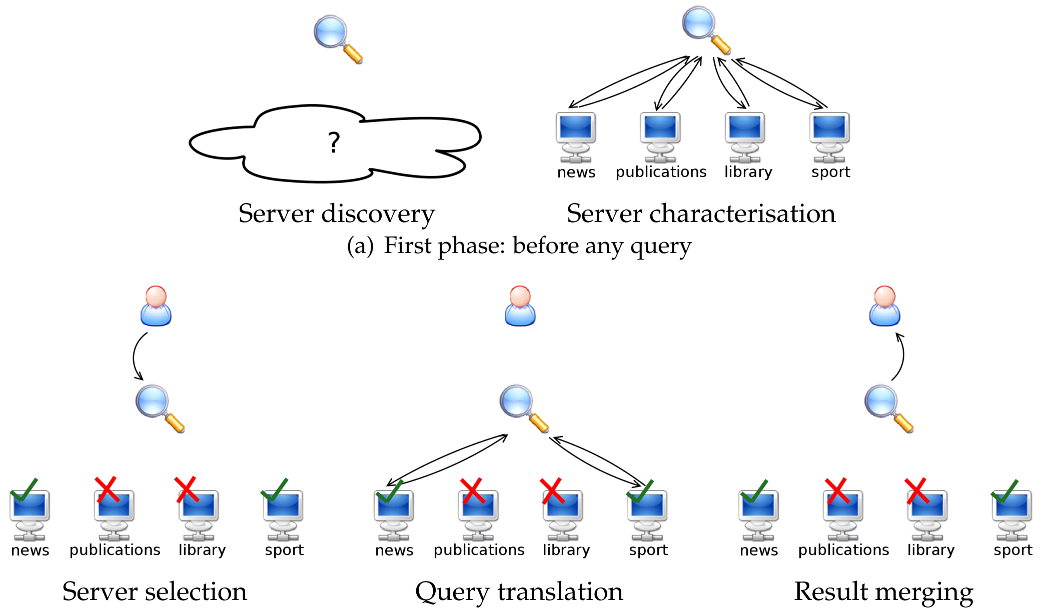
2.2.1 Assumptions

In building a working metasearch tool, some minimum assumptions are necessary. In particular, such a tool requires that each server has a well-defined and stable interface, which accepts a query and returns a set of more or less useful documents. More expressive client-server protocols supporting metasearch have been suggested, but are not well-used. For example, Z39.50 [ANSI/NISO 2003] is supported in the main only by bibliographic databases, and STARTS [Gravano et al. 1997a; Gravano et al. 1997b] and Pharos [Dolin et al. 1996] have not been widely implemented. Smaller, personal or corporate collections such as intranets, email archives, calendars, and the like do not support any such protocols and are unlikely to in the future.

In the general case a useful metasearch tool should not make any assumptions about the way a server generates a result set, or the quality of the search algorithm employed. Further, although result sets are typically ordered and very often include extra information, such as document titles or summary text, for generality this thesis assumes that only an unordered set of document identifiers is available.

Some techniques make further demands of server capabilities. Many sampling techniques, most algorithms for characterising term distributions, and some server selection algorithms require that the text of documents is available to the metasearcher. Document ranking and merging techniques variously make use of document text, server-assigned quality scores or ranks, or document titles and summaries. In the following chapters these requirements are noted, should they exceed the minimum outlined above. These additional requirements may affect the techniques' applicability in some cases.

[8] PIS, the personal metasearch tool described in Section 3.4, is implemented in this way.



(a) First phase: before any query

(b) Second phase: for each query

Figure 2.2: A metasearch process

2.2.2 A metasearch process

Figure 2.2 suggests a process for metasearch. In this view there are two phases: a first phase, carried out before any query is issued, discovers and characterises available servers. A second, query-time phase selects a set of servers for a user's query; translates the query as appropriate, issues the translated queries, and gathers result sets; and merges the result sets before presenting them to the user. Each step is discussed further below.

Server discovery Before a metasearch tool can be used, it must determine which data sources are available. This can be automated to some extent — for example, Cope et al. [2003] give a decision tree which can find search interfaces in a Web crawl and Barbosa and Freire [2005] introduce a "form-focused crawler" — but is generally considered a manual process.

Server characterisation Having determined which servers are available to a user, the offline phase continues by determining relevant characteristics of each. (This step is elided by some metasearch engines, which assume all servers cover the same collection and operate in similar fashion [Selberg and Etzioni 1995], or which hard-code characteristics of servers used [Glover et al. 1999].) Relevant characteristics for metasearch tools may include collection size [Liu et al. 2001; Shokouhi et al. 2006; Si and Callan 2003b], subject matter [Callan et al. 1999; Callan and Connell 2001; Gravano and Ipeirotis 2003; Ipeirotis et al. 2001], effectiveness [Craswell et al. 2000], and overlap between collections [Bharat and Broder 1998; Hernandez and Kambhampati 2005].

A tool with knowledge of a user's preferences may also attempt to discover characteristics such as language or languages used, reading level, update frequency, or geographic coverage.

Server characterisation can rely on manual descriptions [Dolin et al. 1996; Levy et al. 1996], classification against a predetermined taxonomy [Gravano and Ipeirotis 2003], or can be entirely automated.

With these two steps completed in advance of any query, the metasearch tool has the information needed to weight, or select, servers on a per-user or per-query basis.

Server selection For reasons outlined in Sections 2.1.3 and 2.1.4, it is not generally desirable or even feasible to forward every query to every server. *Server selection* is the process of choosing the server or servers most useful for answering a particular query.

Several algorithms have been developed for performing server selection, each of which uses information about the query as well as characteristics of each server to predict which are likely to provide the best match. Examples include CORI [Callan et al. 1995], CVV [Yuwono and Lee 1997], GlOSS [Gravano et al. 1999], Kullback-Leibler divergence [Si et al. 2002], and ReDDE [Si and Callan 2003b].

This step has also been called "collection selection" (for example, Callan et al. [1995]), "subcollection selection" [Baumgarten 1999], "text database discovery" [Gravano et al. 1994], "text-source discovery" [Gravano et al. 1999], and "database selection" [French et al. 1998]. This thesis uses the term "server selection" as the selective metasearch model already assumes a one-to-one correspondence between servers and collections; "database" is ambiguous in the case where some server is in fact a front-end to a business database; and "text-source discovery" is likely to be confused with "server discovery".

Query translation In some cases, the query as entered by the user will need to be translated from the metasearcher's syntax to that of each selected server. The translated queries can then be forwarded to the selected servers, and a result set gathered from each. This portion of the process is likely to be driven by rules or grammars defined when servers are discovered.

Query translation is only possible to the extent that each query language has matching semantics [Chang et al. 1996]. In some cases, more sophisticated semantics (such as fielded search, proximity, or different notions of "aboutness") may be lost; the extent of this problem depends on the servers used and is not presently understood.

Result merging Given a number of result sets, from those servers selected which have returned one or more documents, the remaining task is to generate a single useful result set. This step is generally called "result merging", although other terms in use include "database merging" [Voorhees and Tong 1997], "collection fusion" [Voorhees et al. 1994], and "subcollection fusion" [Baumgarten 1999]. In the case where collections overlap in one or more documents, it is possible that two or more result sets may include the same document and a deduplication step will be necessary [Bernstein et al. 2006; Conrad et al. 2003]. Further, in most cases a metasearch tool will return a ranked list, and not an unordered set, to the user and therefore a ranking step will be needed as part of the result merging process [Callan et al. 1995; Rasolofo et al. 2003; Si et al. 2002]. This ranking step may make use of overlap in result sets, if any, as well as document features and any knowledge about the servers.

2.3 Operational systems

A number of metasearch systems have been demonstrated, searching library catalogues, news sources, and especially the Web. They have adopted either an exhaustive or a selective model.

2.3.1 Exhaustive metasearch systems

A number of exhaustive metasearch systems use public search engines to search the Web. MetaCrawler [Selberg and Etzioni 1995] was developed to increase coverage of Web search since "no single search service [was] sufficient" and at the time the growth of the Web made it seem unlikely that a single search service would previde enough coverage. MetaCrawler submitted users' queries to six search engines in parallel (at the time of writing, the current version uses eight or more[9]), collated the results, re-sorted them and removed duplicates. Optionally, MetaCrawler could make a local copy of each result to remove dead links and improve sorting.

Users' clicks on results were logged and treated as an indication that the clicked-on result was useful. Using this metric, Selberg and Etzioni calculated that the best contemporary search engine could return only 45% of the total pool of useful results and concluded that metasearch was significantly improving performance of Web search, although later experiments showed no improvement [Hawking et al. 2001].

As of August 2007 similar exhaustive systems for Web metasearch, working on the same model, include Dogpile, Ixquick (which also offers manual server selection), and Mamma.[10]

2.3.2 Selective metasearch systems

Selective metasearch systems have seen significantly more research effort, but have been used only on a small scale. As well as whole-of-Web search services, they have been used for more specialised domains.

The ProFusion system of Gauch et al. [1996] used six Web search engines to increase coverage of the public Web. It offered automatic server selection as well as manual selection; the automatic selection used a list of known terms to determine

[9] http://www.metacrawler.com/

[10] Dogpile: http://www.dogpile.com/. IxQuick: http://www.ixquick.com/. Mamma: http://www.mamma.com/.

which of 13 categories a query belonged to (such as "food" or "society, law, and government") and then chose a server based on prior manual relevance judgements for queries in this category.

Documents were deduplicated and re-sorted according to server-assigned scores and per-server modifiers based on recent performance and the assigned query category. Evaluation with twelve queries suggested that precision and recall were substantially improved over any single Web search engine and over other metasearch systems of the time.

Like MetaCrawler and ProFusion, the SavvySearch system [Dreilinger and Howe 1997; Howe and Dreilinger 1997] was motivated by the apparent impossibility of a single search engine covering the public Web. SavvySearch used a pool of eleven servers, and performed server selection to minimise network load and processing effort at each server. Two to six servers were selected for each query, depending upon network and CPU load at SavvySearch itself.

Server selection in SavvySearch used a combination of recent server performance (as measured by the number of results returned and the response time) and learned effectiveness on similar queries in the past (as measured by the number of user clicks on results from each server). Informal evaluation from several months of public use suggested it was a well-liked and useful service, and that precision improved as query terms were repeated.

Inquirus 2 [Glover et al. 1999] also provided whole-of-Web metasearch, using eight servers (six general and two specific to news and current events). Users were invited to select one of seven categories, such as "research papers" or "individual homepages", when submitting a query, and these categories were used to select servers based on manual rules. (A similar feature was mooted for SavvySearch.) Choice of category also affected query translation; for example, Inquirus 2 added the terms "abstract" and "references" to some translated queries in the "research papers" category.

Results were downloaded by the metasearcher and reordered according to rules which again varied by category and which were based on features such as topical relevance, reading level, and the presence or absence of particular keywords. In informal evaluations, this reranking in particular appeared responsible for promoting many relevant documents which otherwise would have been far down the combined result list.

Four other projects have examined metasearch in more specialised environments: MetaSEEk for image search, PENG in the context of journalism, AllInOneNews for news search, and FedLemur in a government portal.

The MetaSEEk tool of Beigi et al. [1998], which offered metasearch for images, was motivated by the difficulties of "knowing where search engines are, what they are designed to retrieve, and how to use them" — a motivation similar to the "cognitive load" observations on p. 7. Users were able to search by colour or texture, giving the system an example image, or by keyword. MetaSEEk's server selection algorithm used the type of search and past performance on similar queries to select up to ten "search options", each a particular mode of a particular underlying server. Performance was measured by clickthrough logging, in the same manner as SavvySearch, and explicit “like” and “don’t like” options attached to each retrieved image.

Beigi et al. [1998] compared their server selection with random selection over 250 sessions, measuring the number of queries per session and the number of images marked “like” and “don’t like”. An informal analysis suggested that sessions using automatic server selection were shorter, which may have been a result of users finding a usable image faster. No difference was observed in users’ feedback, but since feedback was not compulsory this may not be surprising. In the same vein, later experiments [Benitez et al. 1998] suggested MetaSEEk users could find a single target image with fewer interactions.

PENG [Baillie et al. 2006] provides personalised tools for journalists. As well as a “push” component, which filters incoming news feeds to display new material, a “pull” component provides a metasearch service. A variant of query-based sampling [Azzopardi et al. 2006; Callan et al. 1999] is used to build a description of each server, personalised for each user; at query time the CORI algorithm [Callan et al. 1995] plus a user-provided “trust” score are used to select servers and merge results.

AllInOneNews [Liu et al. 2007] is a news metasearcher which is intended to be more up-to-date and more scalable than alternatives with a central index. Descriptions of each news server are built by crawling, and AllInOneNews uses these descriptions for server selection and result merging.

FedLemur [Avrahami et al. 2006] is a metasearch system built on the Lemur toolkit[11] which searches official statistics from 20 US government agencies by using existing Web-based search functions. As with PENG, FedLemur uses query-based sampling to build a model of each collection, and CORI or SSL [Si and Callan 2003c] to merge results. Tests reported by Avrahami et al., using Cranfield-style queries and judgements (see Section 8.1.1), suggest selecting 30 documents from each of three servers for each query; however no comparisons have yet been made with a central index.

2.4 Summary

This chapter has considered five models for searching multiple collections. The multiple services model, which is common at present, is inconvenient and prone to error; the other common alternative, the central index model, is infeasible in most circumstances and may lose local optimisations. The remaining alternatives are metasearch models, which present a single interface to the user but delegate the work of searching to existing specialised tools. Of these, the hybrid model is the most feasible in real-world situations and is the model considered in the remainder of this thesis.

Given some minimal assumptions about the participating servers, metasearch implementations typically use a two-phase process involving some processing prior to any query (server discovery and characterisation) and some for each query (server selection, query translation, and result merging).

[11] http://www.lemurproject.org/

A number of metasearch systems have been built, either on an exhaustive or a selective model. Many have been motivated by the apparent impossibility of indexing the entire Web; current Web search engines however provide an excellent service without resort to metasearch for coverage. A small number of other tools operate in more specialised domains, but none suggest a compelling use of metasearch techniques. One application which would be useful, which could not be built but for metasearch techniques, and which has not been seriously considered, is *personal metasearch*. This is described in the next chapter.

Chapter 3

Personal metasearch

The metasearch systems of Section 2.3 operate over a set of public collections, in most cases, and over collections which are more or less homogenous. This thesis advocates an alternative application, *personal metasearch*, which instead operates over all the heterogeneous and possibly private collections which a user has access to.

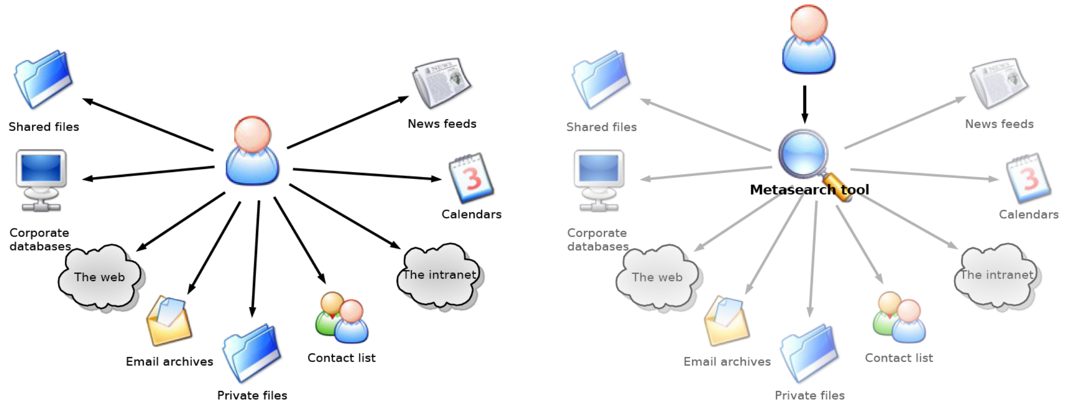
Most computer users have access to a vast amount of information in electronic form: personal files, calendars, public and private web sites, corporate databases, email, and so forth (Figure 3.1). At present, each information source typically offers its own search tool or tools, each with its own interface and each with different capabilities and restrictions: this is the multiple services model of Section 2.1.1. Given that a needed piece of information may come from any of these sources, a user needs to: (1) decide where it's likely to be found; (2) start or switch to the appropriate tool; (3) translate their information need into some appropriate syntax or sequence of actions; and (4) scan the resulting set of documents for the information they need. If the need isn't satisfied, the user will have to either reformulate the query (and repeat steps 3–4) or, worse, start a different tool to look elsewhere (and repeat all of steps 1–4).

A single metasearch tool operating over all collections would reduce cognitive load by eliminating step 1 and the need to guess which source is needed; by providing a single interface, it would also save users having to learn how to use several tools. A single tool would also save the time needed to start seperate applications or open separate URLs. Further, by searching all sources at once a metasearch tool would mitigate the risk of missing information by choosing the wrong source, or of accepting information while a more correct or up-to-date version can be found elsewhere. Such a tool compares well to other solutions (Section 3.1), and offers an alternative view of personalisation (Section 3.2). A survey of searchers reported in Section 3.3 suggests a personal metasearch tool would be of real benefit.

3.1 Alternatives

There are apparent alternatives to a personal metasearch tool, each of which provides a unified interface to at least some data types.

A number of free and commercial "desktop search" programs provide a single search interface to many, or most, of the documents available on a single PC. Ex-



(a) Distinct tools (b) Unified tool

Figure 3.1: An example of the range of information sources available to an individual. A personal metasearch system aims to provide a unified search interface to all of them.

amples include Beagle,[1] Copernic,[2] Google Desktop,[3] Phlat [Cutrell et al. 2006], Spotlight,[4] Stuff I've Seen [Dumais et al. 2003], Yahoo! Desktop Search,[5] and many others. Typically, they are able to index a large number of common data types, and will crawl a filesystem to index all local or networked files.

Personal information management (PIM) systems offer another alternative by presenting a single front-end for managing a variety of data types. These systems can offer search capability as part of their single-tool approach; examples include Haystack [Adar et al. 1999] and TaskVista [Bellotti and Thornton 2006].

Both approaches provide a single interface to a variety of document types, but both adopt the central index model. This has several implications, as outlined in Section 2.1.2. The most important are a limit to the data which can be indexed, and a loss of optimisations.

For the most part, these programs are only able to consider local files. Some also offer search of the public Web, either through affiliation with a Web search engine or in a very limited sense through a search of visited or cached Web pages. Other collections are not searchable by these programs even in principle: even if there were parsers or adapters for every collection, the size of the collection or the rules surrounding document access would make local indexing impossible in many cases.[6] As many users make use of data on local intranets or subscription services (Section 3.3.2), this is an important omission.

Further, any optimisations of more specialised tools may be lost. For example, a

[1] `http://www.beagle-project.org/`

[2] `http://www.copernic.com/`

[3] `http://desktop.google.com/`

[4] `http://www.apple.com/macosx/features/spotlight/`

[5] `http://desktop.yahoo.com/`

[6] For example, in May 2007 Dialog, a commercial search tool for publications, claimed to include 20,000,000,000 documents.

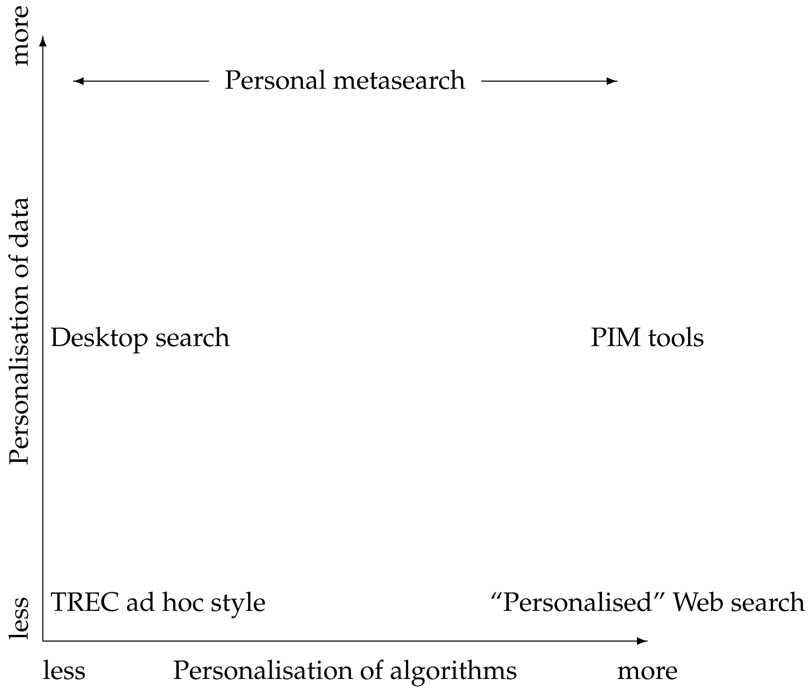


Figure 3.2: Two orthogonal approaches to personalising search tools: personalisation of algorithms or of data. See Section 8.1.1 for comments on "TREC ad hoc".

calendar application may display upcoming appointments in preference to past ones in response to a query; this sort of local knowledge would be lost if the same files were indexed centrally.

These limitations can be overcome only by reverting to a multiple services model, which is undesirable, or by introducing metasearch as a personal application.

3.2 Personalisation

As a tool used by individuals, a search tool can be adapted to each user and their particular circumstances. Figure 3.2 illustrates two orthogonal approaches to personalisation: personalisation of algorithms, as exemplified by personalised Web search and some PIM tools, and personalisation of data, as exemplified by PIM tools and desktop search.

Personalisation of algorithms The more common approach is to search the same collections for each user, but use adaptive algorithms which modify each result set according to the user's preferences. Examples include UCAIR [Shen et al. 2005], which used a user's viewing patterns, as exposed by clicks on result lists, to rerank Web search results; a more comprehensive variant from Teevan et al. [2005b] also considered documents indexed with Stuff I've Seen [Dumais et al. 2003] and Web pages viewed as evidence in reranking. Both studies showed improved performance over Web searches for a number of users. A similar approach was suggested by McGowan et al. [2002], and algorithms for personalisation have been further considered by Glover et al. [1999], Limbu et al. [2006], and Pitkow et al. [2002] amongst others.

A common motivation for personalising a tool in this way is to automatically disambiguate queries which might otherwise perform poorly [Koutrika and Ioannidis 2005; Limbu et al. 2006; Shen et al. 2005; Sugiyama et al. 2004; Teevan et al. 2005a]. Experiments across a range of search tools, however, have found little likely benefit from automatic disambiguation in domains where a user has some knowledge [Thomas and Rowlands 2007]. The benefit to using these techniques in personal metasearch therefore seems small.

Personalisation of data An alternative approach, which can be investigated independently, is to adapt search tools by using different data sources for each user, including data sources which (alone or in combination) may be unique to an individual. This approach has been explored to some extent by desktop and PIM tools, but the central index model adopted by these tools limits the possible data sources and hence the personalisation. A personal metasearch tool should be capable of much greater personalisation in this sense. This alternative requires re-examination of metasearch techniques, and has not yet been well studied.

3.3 A survey of information sources

In order to understand the variety of collections in day-to-day use and understand the potential benefit of personal metasearch, a number of knowledge workers were invited to participate in a survey. Participants, all of whom were frequent users of a variety of search services, were asked about tasks they perform with computer-based sources; what those sources were; and characteristics of each.

Information gathered by this survey includes collection characteristics such as scale, contents, and search interfaces, which can inform system design. Information on scale and contents can also inform evaluation of personal metasearch sytems, for example by suggesting appropriate test collections. Other information, such as cost and success rates, provides realistic constraints on the interactions a metasearch tool may have with servers. After describing one or more sources, participants were also invited to comment on metasearch tools and the search process in general.

Survey instruments, including a full list of questions, are included as Appendix B.

3.3.1 Respondents and tasks

Respondents were recruited from amongst people who were heavy users of computerised information sources. While this convenience sample is not representative of computer users more generally, it does include people who make heavy use of search, who use many sources, and who have a good understanding of the sources they use. It is reasonable to expect that early adopters of new search tools would share these characteristics.

The survey ran for five months and 52 responses were recorded. Respondents included librarians, undergraduate and graduate students, taxonomy developers, researchers, lecturers, journalists and news analysts, as well as a publisher, a translator, a computer system operator, an adminstrative officer, and an editor. They reported using computerised sources for a wide variety of tasks including research for colleagues and customers, their own research, developing teaching material, literature search, developing taxonomies and thesauri, finding material for publication, checking copyright information, consulting archives, editing, translating, budgeting, carrying out technical support, and entertainment.

Many respondents indicated that these tasks involve searching multiple collections, using multiple interfaces; for example, a librarian described research for clients "involving general internet searching, individual website searches, library databases and subscription services". Other responses described "searching databases and [the] web" and "[searching] main sources of information".

A follow-up question asked when participants were most likely to give up searching. Responses here also suggest that search over several collections is commonplace: a librarian described giving up "when the information cannot be tracked down in both hardcopy and electronic resources" and a reference librarian was likely to give up "when I exhausted all the library's stock and all my networks".

Most respondents indicated they considered computerised resources a first port of call, and only gave up a search after spending some effort and possibly checking in a number of collections. This suggests that anything which improves search for these heavy users will have an immediate impact. Other responses suggest that in many cases a personal metasearch tool would provide such an improvement.

3.3.2 Data sources

The major part of the survey attempted to understand characteristics of the data sources used. The vast majority of respondents indicated that they made use of local files, as well as files shared amongst their section or their organisation, the public Web, and email archives. Many also use a local intranet.

Other sources mentioned were varied and included:

- library catalogues;
- bibliographic databases, scholarly and general, including WorldCat,[7] JSTOR,[8] and Project MUSE;[9]
- online journals and e-books;

[7] http://www.worldcat.org/

[8] http://www.jstor.org/

[9] http://muse.jhu.edu/

- news sources including Factiva,[10] Dialog,[11] LexisNexis,[12] and online newspapers;
- image databases including ARTstor;[13]
- archives, including census data and digitised manuscripts;
- reference material including encyclopedias, Wikipedia,[14] glossaries, monolingual and bilingual dictionaries;
- legal and financial databases such as LawPoint;[15]
- online forums including bulletin boards, blogs, and Usenet;
- specialised databases such as patent, medical, legal, or taxonomic databases;
- local resources including CD-ROMs, backups, wikis, Lotus Notes, brochures, and manuals;
- and many unspecified commercial databases.

The majority of collections had some sort of search facility: 81 replies to the question "does this source offer a search facility of any kind?" were affirmative and only 7 negative. Although many collections required ongoing access fees or the purchase of software, no responses suggested any form of per-query fee.

Respondents identified some overlap between different collections, but it appears limited. Describing local files, one noted "some of the information can be found in other work sources, but not the complete information". Comments on specialised databases, available over the Web or otherwise, included "sometimes [data can be found elsewhere] but usually there is a time delay in access"; "yes can be found elsewhere"; and "... some of my data is from the IPUMS [Integrated Public Use Microdata Series] ... Other material is available at ICPSR [Interuniversity Consortium for Political and Social Research]".

There are clearly a great variety of collections in use, comprised of a variety of data types. In some cases, sets of collections are covered by existing search interfaces: Factiva for example searches the archives of many news and business publications, and Project MUSE searches many non-profit academic publishers. However, none of these cover more than one data type and overlap is limited. Further, since every respondent indicated they use at least one private collection, interfaces such as these could not cover the complete set of collections likely to be used at any one time.

[10] http://www.factiva.com/

[11] http://www.dialog.com/

[12] http://www.lexisnexis.com/

[13] http://www.artstor.org/

[14] http://www.wikipedia.org/

[15] http://www.lawpoint.com.au/

3.3.3 Implications for metasearch

In many cases, the collections identified above cannot be indexed centrally for reasons of access, timeliness, or scale. Each collection is used in day-to-day tasks, however, so it would not be appropriate to arbitarily exclude any from a unified interface. This argues for a metasearch interface, which could include all potentially useful collections without the cost of maintaining a central index.

Other considerations suggest a particular model of metasearch. Since some collections (for example, Project MUSE and Dialog) charge according to the number of accesses or time connected, and since some tasks described involve very particular information needs, an exhaustive model is not appropriate; further, since some collections do not offer their own search service, the selective model is infeasible. This suggests the hybrid approach described in Section 2.1.5.

The number of collections each person has access to, and the small overlap from person to person, suggests that techniques for server characterisation must demand little manual intervention. This thesis therefore considers fully automatic techniques for characterisation, not manual or semi-manual techniques such as those of Glover et al. [1999] or Dolin et al. [1996].

Comments on metasearch

At the end of the survey, participants were asked whether metasearch tools would be of use ("do you think it would be useful to be able to search all or some of the sources you've identified via a single search interface?"). Responses varied, but were broadly supportive:

- "yes, I see great value in federated searching, but also to have the option to search individual sources if required" (a librarian);
- "yes, like Gigablast or Rollyo" (a librarian);[16]
- "would be appropriate for similar resources" (a systems librarian);
- "that would be easier" (a graduate student);
- "searching Wikipedia and getting relevant sources from WorldCat could be useful" (a graduate student);
- "yes ... I have not come across any centralized interface for archive holdings ... what I might find HIGHLY useful is a centralized repository describing the online presence of varied archives" (a student);
- "if my internal documents [were] linked to the larger software, I could use both more efficiently" (an administrator);

[16] Gigablast: `http://www.gigablast.com/`. Rollyo, a Web metasearcher with manual server selection: `http://www.rollyo.com/`.

- "yes, especially when people can't remember whether the fact they're seeking was in [a specialist collection] or an email, or saved Word document" (a taxonomy developer);
- "yes ... to simultaneously search local documents/files as well as the WWW" (an IT consultant);
- "at times a federated search would be good. Especially a search of the subscription databases" (a librarian);
- "many of my customers would love to have one search interface, they often don't have the skills to identify the best starting point" (a librarian).

Negative comments included:

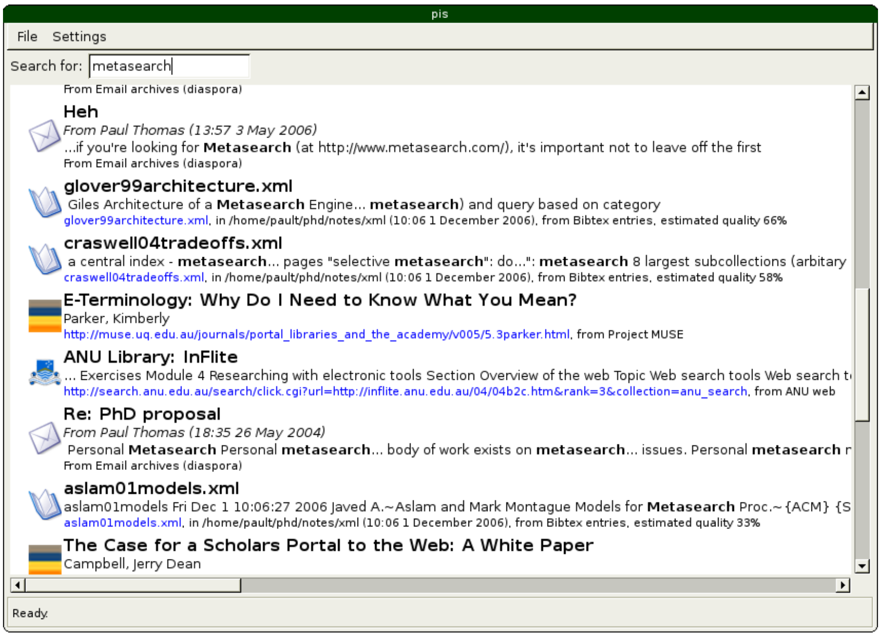
- "no, I wouldn't trust it" (a reference librarian);
- "often when I use source-specific interfaces that is OK because selecting the source is part of the search process" (a graduate student in history);
- "I have mentally organized these different search/information areas" (a graduate student in the history of medicine);
- "absolutely not. The databases that I use are specialized and well designed" (a PhD student);
- "...stuff I get from people I know, which means I actually have to talk to them" (a university lecturer).

It appears that people who make heavy use of search services, and who use several different collections day-to-day, largely consider metasearch to be useful. Those who disagree have concerns about losing the specialised nature of collections, or the specialised tools available; a good metasearch tool should therefore at least offer explicit server selection. It may also expose the advanced features of any underlying servers, although these features are rarely used in traditional search [Silverstein et al. 1999] and may not be of pressing importance even in specialist retrieval for research [Cox 2006].

Together with the observations above and in Section 2.1, the responses to this survey suggest that many people commonly use more than one collection, and that a personal metasearch tool could be of real use in this situation. In the next two sections, a working metasearch tool is described and some research problems are outlined.

3.4 PIS, a personal information searcher

A *p*ersonal *i*nformation *s*earcher, PIS, has been implemented. This prototype personal metasearch tool has been used as a testbed for metasearch algorithms, to gain experience in practical aspects of personal metasearch, and in the user experiments of Chapter 9. Figure 3.3 shows PIS searching eleven collections for documents on metasearch, and presenting a single merged result list.



Search for: metasearch

From Email archives (diaspora)

Heh
From Paul Thomas (13:57 3 May 2006)
...if you're looking for **Metasearch** (at http://www.metasearch.com/), it's important not to leave off the first
From Email archives (diaspora)

glover99architecture.xml
Giles Architecture of a **Metasearch** Engine... **metasearch**) and query based on category
glover99architecture.xml, in /home/pault/phd/notes/xml (10:06 1 December 2006), from Bibtex entries, estimated quality 66%

craswell04tradeoffs.xml
a central index - **metasearch**... pages "selective **metasearch**": do...": **metasearch** 8 largest subcollections (arbitary
craswell04tradeoffs.xml, in /home/pault/phd/notes/xml (10:06 1 December 2006), from Bibtex entries, estimated quality 58%

E-Terminology: Why Do I Need to Know What You Mean?
Parker, Kimberly
http://muse.uq.edu.au/journals/portal_libraries_and_the_academy/v005/5.3parker.html, from Project MUSE

ANU Library: InFlite
... Exercises Module 4 Researching with electronic tools Section Overview of the web Topic Web search tools Web search t
http://search.anu.edu.au/search/click.cgi?url=http://inflite.anu.edu.au/04/04b2c.htm&rank=3&collection=anu_search, from ANU web

Re: PhD proposal
From Paul Thomas (18:35 26 May 2004)
Personal **Metasearch** Personal **metasearch**... body of work exists on **metasearch**... issues. Personal **metasearch** r
From Email archives (diaspora)

aslam01models.xml
aslam01models Fri Dec 1 10:06:27 2006 Javed A.~Aslam and Mark Montague Models for **Metasearch** Proc.~{ACM} {S
aslam01models.xml, in /home/pault/phd/notes/xml (10:06 1 December 2006), from Bibtex entries, estimated quality 33%

The Case for a Scholars Portal to the Web: A White Paper
Campbell, Jerry Dean

Figure 3.3: PIS, a *p*ersonal *i*nformation *s*earcher.

PIS implements effective algorithms for server characterisation (sampling and collection size estimation) as well as several alternatives for server selection (four methods) and result presentation (six methods). As a hybrid metasearch tool, it does its own indexing of email, calendars, contact lists, and local files; it uses adapters to existing tools for LDAP directories, the public Web, web front-ends to other servers, and SQL databases.

PIS has been tested on Unix and Windows computers. An overview of the design and information on capabilities, including modules for document sampling, file parsing, server characterisation and selection, searching, and result merging are in Appendix E on page 185.

3.5 Research problems in personal metasearch

The process outlined in Section 2.2.2 suggests several areas of research interest in personal metasearch.

Although a small body of work has considered the problem of server discovery, as discussed in Section 2.2.2, it has generally been assumed that servers are known to the metasearch tool ahead of time. Given the wide range of possible servers in a personal application, and users' knowledge of their own work tasks, it is reasonable to assume that servers will be identified by users (or, for example, system administrators) in advance.

Further, although the range of document types considered here is wider than in other work, there is no reason to believe the result merging problem should be significantly different to that already studied. Informal evaluation using the PIS tool appears to confirm this intuition (Section 9.3). Despite the importance of a good result merging technique, then, it is not considered in any detail here.

Two major questions for building a working personal metasearch tool remain: how can we characterise servers? Having done so, how can we select between them? This thesis concentrates on these problems. They have been considered in earlier work, but in very different environments, and evidence from the present research suggests that conclusions from earlier studies do not always hold in this new application. In what follows, Chapters 4 to 6 consider techniques for server characterisation (sampling, size estimation, and subject matter and language); Chapter 7 considers server selection.

Finally, having built a personal metasearch tool the remaining questions are: how can we evaluate it? How can we compare one choice of algorithm or parameter with another? Chapters 8, 9 and 10 consider these questions.

3.6 Summary

This thesis advocates *personal metasearch*, or metasearch over all the collections a computer user has access to including private or restricted collections such as email or subscription services. A personal metasearch tool would reduce cognitive load for users and mitigate the chance of certain types of error.

Such a tool can be distinguished from desktop search software, which search a wide variety of sources but are limited by their central index model; and from personal information management software, which provide a single management interface and similarly do not provide metasearch functions. It can also be distinguished from "personalised" Web search by considering personalisation of *algorithms* separately from that of *data*.

This analysis is supported by a survey of frequent searchers. The survey suggests that users have access to many collections, with very different data, and typically each with its own interface; that many search tasks make use of multiple collections; and hence that a metasearch tool is practical and would be of benefit.

A working prototype has been constructed, but the metasearch process of Section 2.2.2 suggests research will be needed in a number of areas and in particular in server characterisation, server selection, and system evaluation. The following chapters consider this research.

Chapter 4

Server characterisation: sampling documents

An important first step in the metasearch process is to characterise each server; that is, to determine ahead of time properties such as the number of documents in each collection, the subject matter, or what language or languages are used. This information is useful at query time to inform the processes of server selection, query translation, and result merging.[1]

In the most general case, we assume that search engines do not cooperate with any metasearch framework and provide only the most minimal interface: they simply accept a query and produce a set of document identifiers as a result. In particular, they do not make available statistics such as the number of documents indexed, term frequencies, or weights; nor do they expose any information on their internal workings. Techniques are therefore needed to estimate these characteristics, and the vast majority of these estimation techniques are improved by or explicitly rely upon random (unbiased) samples of documents. Overlap estimates [Bharat and Broder 1998] explicitly rely on random samples as input; so too do the standard capture-recapture [Liu et al. 2001], sample-resample [Si and Callan 2003b], multiple sample-resample and capture history [Shokouhi et al. 2006], and random document [Broder et al. 2006] techniques for estimating collection sizes. As demonstrated in Chapter 5, biased samples lead to systematic underestimates of collection size.

As will be shown in Chapters 6 and 7 biased samples, and hence biased term statistics and size estimates, also have a negative impact on the accuracy of language models and on the performance of server selection algorithms.

Obtaining a random sample from an uncooperative search engine is however a non-trivial task. With a limited interface, it is not possible to enumerate documents in the collection or to retrieve them according to some identifier; therefore a metasearcher cannot simply take a uniform sample. Further, characteristics of the search engine which improve performance for typical uses are likely to make sampling less convenient. For example, certain documents may be more likely to be returned since

[1] Some of this chapter has been published as Thomas and Hawking [2007]. The experimental work was original with the author, as was other material reproduced here including the multiple queries sampler and the "T" and "S" tests.

they are considered in some sense more important, or more useful, and this introduces strong bias. Similarly, result lists may be arbitrarily truncated, or near-duplicates may be removed, to save work at the server or to present a more useful list to a client. Any random sampling technique will have to work in this environment.

Ideally, for metasearch we would like a sampling technique which produces samples with as little bias as possible; which requires as little run-time or pre-processing resources as possible; and which works over a wide range of collections and servers with as little prior knowledge as possible.

In this chapter techniques are evaluated against these criteria. A number of candidate algorithms have been developed for sampling from the Web, some of which are more generally applicable, and the runtime cost and performance of these are considered. A new method, "multiple queries", is also introduced: this provides samples of similar quality to existing methods with much reduced cost. Finally, this chapter examines the effects of the parameters available in each technique.

4.1 Notation

In the discussion which follows, notation follows that of Bar-Yossef and Gurevich [2006]: $\mathcal{D}$ is the set of all documents available through a server, d an individual document from $\mathcal{D}$, and $N = |\mathcal{D}|$ the number of documents a server provides access to. $\overline{x}$ represents the mean of some value x.

For each query q sent to a server, $\textsc{res}(q)$ denotes the results returned. This result set may be constrained by a limit k, imposed either by the server itself or by a sampler; if $|\textsc{res}(q)| \geq k$ we say that q "overflows", and if $|\textsc{res}(q)| = 0$ we say q "underflows". Note that Bar-Yossef and Gurevich [2006] refer to overflow if $|\textsc{res}(q)| > k$. This thesis prefers the definition $|\textsc{res}(q)| \geq k$ as, without extra information from a server, it is not generally possible to tell whether a result set is bounded by k or by the number of matches. The two definitions are interchangable by incrementing or decrementing k.

This and other notation is summarised in Appendix A.

4.2 Related work

The problem of sampling from an uncooperative server is very similar to that of sampling from the Web, and several algorithms have been introduced for the latter. These are summarised in Table 4.1.

"General" methods do not rely on particular document characteristics, and are applicable across a wide variety of document types.

4.2.1 Single queries

Bharat and Broder [1998] introduced a simple technique for sampling random pages from Web search engines. The technique does not however rely on any particular

	Queries needed	Docs/ run	Hyperlink graph	Doc text
General methods				
Single queries	Stream	1	—	—
Query-based	—	Any	—	Needed
Pool-based	Pool	1	—	Needed
Random walk on MATCH$_{\mathcal{P}+}$	—	1	—	Needed
Multiple queries	Stream	Any	—	—
Hyperlink methods				
PAGERANK-SAMPLE	—	Any	Needed	—
WebWalker	—	1+	Needed	Needed
(UN)DIRECTED-SAMPLE	—	1+	Needed	—

Table 4.1: Characteristics of sampling methods. "General" methods do not require a hyperlink graph and are examined in the experiments of this thesis.

characteristic of the Web or of Web pages and is applicable to a variety of collections. The algorithm is extremely simple: a single query is constructed and sent to a search engine, and a sample document is chosen at random from the set of matches returned. To build a sample of more than one document, the algorithm can be repeated.

The algorithm was orginally applied to Web search engines, which typically return a small set of results regardless of the number of possible matches. As a work-around, Bharat and Broder generated queries which they expected to return between one and 100 documents, although if a query matched more than 100 documents they used only the top 100 returned. Query terms came from a lexicon built during an earlier crawl, and queries were either four-term disjuncts or two-term conjuncts with terms chosen for their frequency.

In a later paper, Gulli and Signorini [2005] used this method with slight modifications to estimate the size of public search engines and hence the public Web. Query terms again came from an earlier crawl, but Gulli and Signorini used single-term queries in more than 75 languages.

Although working with an uncontrolled, dynamic collection meant Bharat and Broder were not able to investigate the quality of the sampler, they identified six sources of bias. Two of these are relevant to the metasearch case. "Query bias" is the bias towards longer, content-rich documents which are more likely to match the queries used. "Ranking bias" is the result of search engines ranking documents and a sampler not seeing those past rank k. By choosing queries with a smaller number of results, they note that it is possible to reduce ranking bias at the expense of increasing query bias. Query bias has been noted in other work [Bar-Yossef et al. 2000; Bar-Yossef and Gurevich 2006] and is confirmed by the results in Section 4.6 below.

In the implementation of the single queries sampler used here, ranking bias is eliminated by choosing and issuing queries until one is found which neither underflows nor overflows.

4.2.2 Query-based sampling

The popular query-based sampling method of Callan et al. [1999] uses terms seen in document text as a source of queries. Although originally introduced as a means of obtaining "sufficiently random" samples for learning language models, it has since been used as a general sampling technique by a number of researchers [Callan and Connell 2001; Hawking and Thomas 2005; Shokouhi et al. 2006; Si and Callan 2003b]. Besides the multiple queries method below, this is the only technique originally aimed at metasearch rather than Web applications.

Starting with an initial one-term query, the server is interrogated and the text of the top r ranked documents is retrieved. Each retrieved document is added to the sample and then used to update a language model m — typically just a set of (word, frequency) pairs. If the stopping criteria are not yet met, a term is chosen from the learned model and the server is interrogated again. Since the algorithm requires access to the text of documents to update the model, it may not be appropriate to all collections.

There are four major choices implementing this algorithm: the source of initial queries, the result cut-off r, the way terms are chosen from m, and the stopping criteria. Callan et al. [1999] suggest that the initial query makes little difference, and their experiments used the top four documents for each query and stopped after retrieving 300 documents or issuing 150 queries. Terms were chosen uniformly from m. Experiments considered the quality of the learned model, not the distribution of the documents sampled, since a generally-applicable random sample was not the original intent [Callan et al. 1999; Callan and Connell 2001].

Callan et al. suggested four methods for choosing query terms from m: uniform selection and selection based on document frequency, total term frequency, or mean term frequency in m. Later experiments by Baillie et al. [2006a] considered these alternatives. Using Kullback-Leibler divergence and predictive likelihood, they compared the quality of learned models for each method across a pair of TREC testbeds. Uniform selection performed well, as did selection based on document frequency, but the samples were not evaluated for bias towards any subset of the collection.

Baillie et al. [2006a] and Shokouhi et al. [2006] also considered alternative stopping criteria, in both cases based upon the rate of change of m. Both groups were able to demonstrate improved performance in a metasearch system evaluation with their stopping criteria as compared with a fixed threshold. In the experiments in this chapter, however, the consideration is not the quality of a learned model but the bias in a sample and so a fixed number of documents are sampled with each run.

In the implementation used here, the first query was chosen at random from the SMART stopword list [Salton and McGill 1983], a list of common English words, and subsequent terms were chosen randomly and uniformly from the terms in m. No stemming or stopping was used in processing the queries.

4.2.3 Pool-based sampling

Rejection sampling is a Monte Carlo method which can be used to simulate sampling according to one distribution π (for example, the uniform distribution) when it is only feasible to draw samples with some other distribution p (such as that resulting from query or ranking biases). Bar-Yossef and Gurevich [2006] introduced an application of this technique to the problem of sampling Web pages. The algorithm consists of a first round of rejection sampling, which chooses a query according to the size of its result set; and a second round, which chooses a document from the result set of this query.

Pool-based sampling requires as input a "query pool" $\mathcal{P}$, a set of queries drawn from all possible queries a server accepts. Ideally queries in $\mathcal{P}$ have high recall (meaning that taken together, they cover a large proportion of the documents in $\mathcal{D}$), and simultaneously a low probability of underflow or overflow. $\mathcal{P}+$ denotes the subset of $\mathcal{P}$ which neither underflows nor overflows, so $\mathcal{P}+ = \{q \in \mathcal{P} : 0 < |\textsc{res}(q)| < k\}$.

$\textsc{match}_{\mathcal{P}}(d)$, the "match set" of d for some $d \in \mathcal{D}$, is the set of queries which a document d matches from amongst a pool $\mathcal{P}$. $\textsc{match}_{\mathcal{P}}(d)$ can be calculated from the text of a document — for example, if $\mathcal{P}$ is the set of all 3-gram queries, it suffices to extract text from d and enumerate all 3-grams. The pool-based sampler does therefore rely on the text of a document being available to the metasearch system; in some cases this will not be true and the pool-based sampler will not be practical. Further, should the sampler and the underlying search engine consider the "matching" operation differently, due for example to differences in stemming and stopping, this difference in $\textsc{match}_{\mathcal{P}}(d)$ will introduce error.

The first part of the pool-based algorithm, summarised in Figure 4.1, uses rejection sampling to choose a query q from $\mathcal{P}+$ with a distribution based on $|\textsc{res}(q)|$. A query is chosen uniformly from the pool and forwarded to the server; if it neither underflows nor overflows we accept it with probability $|\textsc{res}(q)|/k$ and continue.[2]

In the second part of the algorithm, a candidate document is chosen uniformly from $\textsc{res}(q)$. At this point the probability of choosing a document d as a candidate is proportional to the probability that d matches the query; in other words to $|\textsc{match}_{\mathcal{P}+}(d)|$. A second round of rejection sampling therefore returns d as a sample with probability $1/|\textsc{match}_{\mathcal{P}}(d)|$, and otherwise iterates selecting another query and document. Note that although documents are presented as candidates with a distribution based on $|\textsc{match}_{\mathcal{P}+}(d)|$, they are selected as samples based on $1/|\textsc{match}_{\mathcal{P}}(d)|$. This introduces an error the size of which depends on the difference between $\mathcal{P}$ and $\mathcal{P}+$. This difference is hard to determine without detailed knowledge of how queries and documents are processed at the search engine.

Experiments to characterise possible pools $\mathcal{P}$ used text from a crawl of web pages in the Open Directory Project (ODP).[3] Bar-Yossef and Gurevich used a crawl of a subset of the ODP as a source of terms and considered single terms, 3-, 5-, and 7-term

[2] The original description of the algorithm leaves the parameters C and $\hat{\phi}(q)$, the envelope constant and the unnormalised sample distribution, unspecified. In the implementation used here, $\phi(q)$ is uniform, the sampler uses an unnormalised version $\hat{\phi}(q) = 1$ for all q, and therefore $C = k/\hat{\phi}(q) = k$.

[3] `http://dmoz.org/`

Step		Probability
1	Select a query q	$\Pr(q \text{ selected}) = 1/\lvert\mathcal{P}\rvert$
2	Issue q	
3	If q overflows, reject it	$\Pr(\neg q \text{ overflows}) = \lvert\mathcal{P}+\rvert/\lvert\mathcal{P}\rvert$
4	Accept q, possibly	$\Pr(q \text{ accepted}) = \lvert\textsc{res}(q)\rvert/k$
5	—	$\Pr(d \text{ in } \textsc{res}(q)) \propto \lvert\textsc{match}_{\mathcal{P}+}(d)\rvert$
6	Select document d from results	$\Pr(d \text{ selected}) = 1/\lvert\textsc{res}(q)\rvert$
7	Accept d, possibly	$\Pr(d \text{ accepted}) = 1/\lvert\textsc{match}_{\mathcal{P}}(d)\rvert$

$$\begin{aligned}
\Pr(d \text{ sampled}) &= \Pr(q \text{ selected})\ \Pr(\neg q \text{ overflows})\ \Pr(q \text{ accepted}) \\
&\quad \Pr(d \text{ in } \textsc{res}(q))\ \Pr(d \text{ selected})\ \Pr(d \text{ accepted}) \\
&\propto \frac{1}{\lvert\mathcal{P}\rvert}\frac{\lvert\mathcal{P}+\rvert}{\lvert\mathcal{P}\rvert}\frac{\lvert\textsc{res}(q)\rvert}{k}\lvert\textsc{match}_{\mathcal{P}+}(d)\rvert\frac{1}{\lvert\textsc{res}(q)\rvert}\frac{1}{\lvert\textsc{match}_{\mathcal{P}}(d)\rvert} \\
&\propto \underbrace{\frac{\lvert\mathcal{P}+\rvert}{\lvert\mathcal{P}\rvert^2}\frac{1}{k}}_{a}\underbrace{\frac{\lvert\textsc{match}_{\mathcal{P}+}(d)\rvert}{\lvert\textsc{match}_{\mathcal{P}}(d)\rvert}}_{b}
\end{aligned}$$

Figure 4.1: Calculations for pool-based sampling. Part a is independent of the query and the document; part b is independent of the query, and independent of the document to the extent that $\mathcal{P}+ = \mathcal{P}$.

phrases for recall, underflow, overflow, and other properties. Five-term phrases were considered the best tradeoff in this instance, and were used in the initial experiments below.

4.2.4 Random walk on $\textsc{match}_{\mathcal{P}+}$

A further method, based on sampling through random walks, was introduced at the same time as pool-based sampling [Bar-Yossef and Gurevich 2006]. This variant uses the Metropolis-Hastings algorithm, which carries out a random walk with each step chosen according to a "proposal function", and before each step employs an acceptance/ rejection procedure to determine whether or not the step will be taken. If the proposal function satisfies certain simple criteria, a walk with the Metropolis-Hastings algorithm will eventually converge on a desired distribution. The parameter B, the burn-in period, determines the maximum number of steps in the walk: as with other random walk methods, this can only be set empirically without prior knowledge of the collection.

Bar-Yossef and Gurevich adapt this algorithm to collections without a hyperlink structure by defining a graph such that two documents are joined by an edge iff there is at least one query which both match: i.e. an edge exists between two documents x and y iff $\textsc{match}_{\mathcal{P}+}(x) \cap \textsc{match}_{\mathcal{P}+}(y) \neq \emptyset$. Given a document d, the sampler then proceeds by choosing a query uniformly from $\textsc{match}_{\mathcal{P}+}(d)$ (which determines a subset of edges from the current document); choosing a document d' uniformly from the results of this query (which determines an individual edge); and then choosing to follow the edge and set d to d' with probability $|\textsc{match}_{\mathcal{P}+}(d)|/|\textsc{match}_{\mathcal{P}+}(d')|$ (which adjusts for the odds of seeing d'). After B iterations, the current document is returned as the sample.[4]

The walk does not require that the entire graph be computed ahead of time, which would require knowledge of every document in the collection; instead, edges can be computed from the text of a document and adjacent nodes can be enumerated, given an edge, from a match set. $\textsc{match}_{\mathcal{P}}(d)$ can also be computed as needed, and queries that under- or overflow can be removed as they are discovered.

Evaluation experiments compared this random walk with the pool based and single query samplers on a testbed of 2.4 million documents from the ODP. With both the pool based and random walk samplers, "no or little" bias was seen due to document size, and no "significant bias" due to the static document rank used by their search sytem [Bar-Yossef and Gurevich 2006]. As with other methods, it seems no quantitative tests for bias have been performed.

Unlike other variants on random walks, a random walk on $\textsc{match}_{\mathcal{P}+}$ does not require an explicit hyperlink structure and is appropriate to a wide range of collections. It is therefore included in the experiments below.

A further set of sampling methods assume documents are linked in a graph, such as a web graph; a random walk can then provide a uniform sample. These techniques are not generally applicable in the metasearch case, since not all constituent collections will have such a structure (databases and catalogues for example may not). They are therefore not examined in these experiments.

4.2.5 PAGERANK-SAMPLE

The PAGERANK-SAMPLE algorithm of Henzinger et al. [2000] samples pages from a web based on a random walk of the web graph.[5] It follows from the observation that, given a web crawl, for any document d

$$\begin{aligned}\Pr(d \text{ is sampled}) &= \Pr(d \text{ is crawled}) \times \\ &\quad \Pr(d \text{ is sampled} \mid d \text{ is crawled})\end{aligned}$$

so a uniform sample can be obtained by sampling from crawled pages such that $\Pr(d \text{ is sampled} \mid d \text{ is crawled})$ is inversely proportional to $\Pr(d \text{ is crawled})$.

Henzinger et al. note that for a walk of length L, where L is "long enough", $\Pr(d$ is crawled$) \approx L \times \textsc{pagerank}(d)$, where $\textsc{pagerank}(d)$ is the classic PageRank score

[4] The general form of the algorithm is to accept each move with probability $P(d')/P(d)\, Q(d|d')/Q(d'|d)$, where P is the desired distribution and Q the sampling distribution. Since P is uniform, the first term can be ignored. $Q(d'|d)$, the chance of seeing d' if we are at d, is proportional to the number of queries both match: $Q(d'|d) = |\textsc{match}_{\mathcal{P}+}(d) \cap \textsc{match}_{\mathcal{P}+}(d')|/|\textsc{match}_{\mathcal{P}+}(d)|$. $Q(d|d')$ is similar, and the second term simplifies to $|\textsc{match}_{\mathcal{P}+}(d)|/|\textsc{match}_{\mathcal{P}+}(d')|$.

[5] Henzinger and colleagues did not name this algorithm. The name PAGERANK-SAMPLE follows Rusmevichientong et al. [2001].

[Brin and Page 1998]. Further, it is possible to estimate PAGERANK(d) by doing a walk on the web graph and using the number of times d is visited, normalised by the walk length. This in turn allows the estimation of $\Pr(d \text{ is crawled})$ and hence the probability of including d in a sample.

There are limitations to this technique. The first is in the choice of L: for the approximation to hold L must be approximately $\sqrt{N}$, but N is unlikely to be known in advance — indeed, as will be discussed in Chapter 5, one major use for samples is to estimate N. Further, the estimation of PAGERANK(d) from the number of visits to d is made approximate by a difference in definition: while PAGERANK is described by a random walk which includes jumps to every node in the web graph, in this case the entire graph will not be known in advance and the sampler is limited to jumping to nodes already crawled. This also limits it to sampling from pages which are connected to the start point.

Experiments by Henzinger et al. investigated several likely sources of error: bias due to the choice of start node, dependence between nodes in the walk, and errors in estimating PAGERANK. Results on an artificial web graph did seem to show a bias towards pages with high in-degree and high PAGERANK, but less than with a more naïve scheme. No quantitative tests were performed.

4.2.6 WebWalker

Bar-Yossef et al.'s WebWalker algorithm [2000], again for sampling web pages, is derived from the fact that a random walk on a regular, undirected graph provides a close to uniform sample of nodes. While web graphs are neither regular nor undirected, and while it is not feasible to know the graph in its entirety before sampling, Bar-Yossef et al. suggest a technique for adapting a web crawler to provide a suitable subgraph.

At each step in a random walk, a search engine is interrogated to find inlinks to the current node. These links, plus outgoing links from parsed HTML, are made undirected and added to the graph along with sufficient self edges such that the node is of some constant, high, degree. The Web graph is therefore transformed into a regular, undirected graph, and the walk will converge on a uniform distribution.

The length required before a walk can produce uniform samples depends on the transition matrix and so on the exact edges in the induced graph. From earlier experiments Bar-Yossef et al. estimate a walk length of 3,000,000 steps, although since the vast majority (>99%) of edges are self edges the actual number of document downloads will be far smaller. With further information about the distribution of links, it is also possible to construct a walk which is longer but allows an arbitrary number of documents to be sampled in one run.

Experiments on documents from a large Web crawl seemed to show a definite bias towards pages with high indegree, but did not seem to be biased towards pages near to (or far from) the seed. Again, no quantitative tests have been reported.

4.2.7 DIRECTED- and UNDIRECTED-SAMPLE

Two further sampling methods based on random walks of the web graph were described by Rusmevichientong et al. [2001]. The intuition is similar to that behind PAGERANK-SAMPLE: on a random walk of the web graph (modified so that there are no "dead ends" and so that every document has a self link), a uniform sample can be obtained by including each document d with probability $\Pr(d \text{ is sampled}) = 1/\Pr(d \text{ is crawled})$. The two methods differ in how they estimate the probability of d being included in a crawl.

DIRECTED-SAMPLE starts with a random walk on the web graph; during the last part of this walk, each distinct document is recorded. These documents are candidates for inclusion. To estimate $\Pr(d \text{ is crawled})$ for each of these candidates, DIRECTED-SAMPLE carries out a second crawl, starting from the candidate and noting the proportion of steps in this crawl which return to d.

UNDIRECTED-SAMPLE reduces the walk length by collecting inbound links for each document visited, in the manner of WebWalker, and constructing an undirected graph. The algorithm then uses the degree of each document d as an estimator for $\Pr(d \text{ is crawled})$ directly, eliminating the second crawl of DIRECTED-SAMPLE.

Rusvevichientong et al. provide a proof that given a long enough walk, sample probabilities with either algorithm approach the uniform distribution. As with PAGERANK-SAMPLER and WebWalker, choosing the walk length L is important but difficult a priori.

A set of experiments on an artificial web graph of 100,000 nodes did not appear to show any bias by either sampler towards documents of high degree, although an implementation of PAGERANK-SAMPLE on the same testbed did appear biased.

4.3 Multiple queries

The multiple queries sampler is a straightforward extension of the single queries sampler. To reduce query bias, the sampler simply runs several queries with a large cut-off k and then chooses any number of documents from the union of all result sets. Note that if a document is returned in reponse to more than one query, the sampler records it only once. This has a similar effect to adjusting for visit frequency in PAGERANK-SAMPLE. Queries are chosen from a pool defined independently of the collection, with as high a recall as possible, and as with other samplers any which under- or overflow are ignored.

The high values of k (10,000 in initial experiments) suggest a large amount of network traffic; however since there is no need to download the text of documents, and since the sampler can return many documents from a single run, this traffic does not seem excessive. Further, since document text is not required the sampler is applicable to a wide variety of servers. Experiments varying k and queries used are described in Section 4.6.2 below.

4.4 Sampling cost

A desideratum for a working metasearch system is that samplers should require as few resources as possible. The most significant resource is communication with the servers being sampled, and this section considers the cost for each document sampled both in the number of queries issued and in the number of documents downloaded and parsed. As noted, the PAGERANK-SAMPLE, WebWalker, and (UN)DIRECTED-SAMPLE algorithms rely on explicit hyperlinks between documents and are not applicable to most collections likely to be used in metasearch; they are therefore not further considered.

For each of the following analyses a first step is to determine how many queries successfully complete (i.e. neither underflow nor overflow). Following Bar-Yossef and Gurevich [2006] the *validity density* of a document, denoted $\text{VDENSITY}(d)$, is just $|\text{MATCH}_{\mathcal{P}+}(d)|/|\text{MATCH}_{\mathcal{P}}(d)|$, or the proportion of those queries d matches which return $1 \ldots k-1$ results. Similarly, $\text{VDENSITY}(\mathcal{P})$ represents the proportion of queries in a pool $\mathcal{P}$ which neither underflow nor overflow: $\text{VDENSITY}(\mathcal{P}) = |\mathcal{P}+|/|\mathcal{P}|$.

4.4.1 Single queries

In its original form, the cost of the single queries sampler is a single query per document sampled for any collection and any source of queries. In the implementation used here, it first must find a query which neither underflows nor overflows; the number of queries needed for this is geometrically distributed with $p_1 = \text{VDENSITY}(\mathcal{P})$, so the sampler can be expected to issue $1/\text{VDENSITY}(\mathcal{P})$ queries per document sampled.

In both implementations, there are no documents downloaded or parsed.

4.4.2 Query-based sampling

The cost of the query-based sampler depends upon characteristics both of the collection (for example, the overlap in queries covering each document) and of the server (for example, the tendency to promote certain documents). It is possible however to determine a lower and an upper bound on the cost.

For n documents sampled in a single run with results cut off at r documents, the minimum number of queries is $\lceil n/r \rceil$, or $1/r$ queries per document sampled. An upper bound is set by the stopping criteria: for example, as originally defined the sampler terminates after issuing a maximum $\lfloor n/2 \rfloor$ queries. By construction, the sampler also downloads and parses each sampled document.

4.4.3 Pool-based sampling

We consider the first round of rejection sampling (to find a query) and the second round (to find a document) seperately.

In the first round, the number of iterations needed to find and select a valid query is geometrically distributed with $p_1 = (\overline{|\text{RES}(q)|}/k)\,\text{VDENSITY}(\mathcal{P})$; so the expected number of queries issued before one is selected is $1/p_1$ or $k/(\overline{|\text{RES}(q)|}\,\text{VDENSITY}(\mathcal{P}))$.

In the second round, having selected a query a document from the result set is downloaded and considered for the final sample. The chance of sampling a document after each iteration is $1/|\text{MATCH}_{\mathcal{P}}(d)|$, so the number of queries selected before a document is selected is geometrically distributed with expected value $\overline{|\text{MATCH}_{\mathcal{P}}(d)|}$.

The expected number of queries executed per document sampled is therefore

$$E(Q) = \frac{k}{\overline{|\text{RES}(q)|}\ \text{VDENSITY}(\mathcal{P})} \overline{|\text{MATCH}_{\mathcal{P}}(d)|}.$$

We can also expect that $\overline{|\text{MATCH}_{\mathcal{P}}(d)|}$ documents will be downloaded and parsed for each document in the final sample.

4.4.4 Random walk on MATCH$_{\mathcal{P}+}$

As with the single queries sampler, the expected number of queries the sampler must issue before finding one which neither underflows nor overflows is $1/\text{VDENSITY}(\mathcal{P})$. After each such query is found, the sampler will download and parse a document; and it repeats the entire process B times (recall that B is the burn-in time of the random walk). It will therefore need an expected $B/\text{VDENSITY}(\mathcal{P})$ queries and B downloads per document sampled, although with more knowledge of the query pool and collection it is possible to collect second and subsequent documents more cheaply by continuing the random walk.

Since the cost depends on B, a tunable parameter, it is possible to improve runtime at the expense of decreasing the walk length and hence the randomness of the collected sample.

4.4.5 Multiple queries

The cost of the multiple queries sampler is determined by the number of queries per sample s_q, the number of documents per sample n, and the pool validity density. Since the method needs s_q queries which neither underflow nor overflow, from which it will select n documents, the expected cost is $(s_q/n)(1/\text{VDENSITY}(\mathcal{P}))$ queries per document sampled. (Note that $\text{VDENSITY}(\mathcal{P})$ will likely be higher for the multiple queries sampler than elsewhere, as k is very high and overflow is less likely.) No documents are downloaded.

4.5 Sampling experiments

A number of experiments explored two questions: first, do the sampling techniques described above provide unbiased samples across a range of collections? Secondly, what effect do the available parameters have on performance?

Tests were run with PADRE [Hawking et al. 2000], versions 5.7.3.33 and 6.3.1.3, doing indexing and searching. No stemming or stopping was used.

		Size (in terms)			
Collection	No. docs	Range	Mean	Std dev	Topics
calendar	1k	1–20	4	2	Mixed
zsh-list	9k	2–59k	176	179	Narrow
procmail	24k	2–14k	207	215	Narrow
email	25k	1–26k	199	295	Mixed
WSJ	99k	9–10k	462	450	Broad
.GOV	1.2M	0–43k	6803	5720	Broad

Table 4.2: Summary statistics of collections used in the experiments.

4.5.1 Collections used

Past work has evaluated sampling techniques either on the public Web [Bar-Yossef et al. 2000; Bar-Yossef and Gurevich 2006; Bharat and Broder 1998; Gulli and Signorini 2005; Henzinger et al. 2000; Rusmevichientong et al. 2001] or on divisions of TREC ad hoc [Harman 2005] or Web Track data [Baillie et al. 2006a; Callan et al. 1999; Shokouhi et al. 2006]. However, these collections do not resemble those likely to be used in personal metasearch applications: the Web is extremely large and covers a great variety of topics, and TREC collections are all of one data type. Further, neither the Web nor TREC collections include private or semi-private information such as email; and most investigations have used only one collection, so a sampler's performance can only be evaluated in a single application.

The experiments in this and later chapters use six collections, summarised in Table 4.2. These are representative of the range of resources which are likely to be used in personal metasearch applications, as examined in Section 3.3: sizes range over three orders of magnitude, data types are varied, and topic areas range from the very focussed (development of the zsh shell[6]) to the very broad (several years' worth of archived email). Each collection is mostly English-language. None are on the scale of the largest likely collections, such as the Web or Dialog. However, rather than ask each user's tool to sample (for example) the public Web, it is likely that a personal metasearch tool would use hard-coded or pre-computed characteristics for larger collections. The collections used here span the likely size range of local, private, or workgroup collections, where pre-computed estimates are infeasible.

Table 4.3 summarises this testbed and a selection of others commonly used in metasearch experimentation — generally partitions of TREC ad hoc or Web data, often created with an eye to producing collections of approximately equal size. The availability of extensive relevance judgments for the TREC ad hoc collections has made them the basis for most metasearch testbeds. However, arbitrary partitionings of a small collection of newspaper and government documents don't seem to model likely applications of personal metasearch as suggested by the survey of Section 3.3.[7] Com-

[6] http://zsh.dotsrc.org/

[7] D'Souza et al. [2004a] note that this division is unlikely to model other metasearch applications

		Size (×1000 documents)			
Testbed	Colls	Range	Mean	Std dev	Partitions
TREC ad hoc					
TREC123-17-bysource	17	7– 226	64	56	By source & date
UBC-100†	100	1– 40	11	9	Sequential, for size
SYM-236	236	< 1– 8	3	3	By source & date
UDC-236	236	3– 3	5	< 1	Sequential, for size
TREC4-kmeans	100	< 1– 83	6	11	By topic
TREC123-10col	10	18– 263	108	—	Arbitrary
Web					
WT2g‡	951	< 1– 8	< 1	< 1	By server
WT10g‡	11590	1– 26	< 1	< 1	By server
.GOV‡	7792	< 1– 16	< 1	< 1	By server
.GOV2‡	17186	< 1– 717	1	18	By server
Other					
Personal metasearch	6	1–1200	234	454	By source

Table 4.3: A selection of metasearch testbeds in the literature. After Hawking and Thomas [2005], Powell et al. [2000], and Si and Callan [2003b]. "Colls" abbreviates "collections". TREC123-10col is not distributed, so data for this collection is incomplete. † The UBC-100 testbed is also referred to as "TREC123-100col". ‡ Figures for Web testbeds assume that documents are divided according to Web server.

pared with other testbeds, the personal metasearch testbed used here has a much larger range of collection sizes, fewer collections, and a much wider variety of document topics and sources.

The "calendar" collection, the smallest in the testbed used here, contains 1049 documents (appointments) from a calendar application, spanning about two years. Documents are typically sentence fragments, only a few terms long, and the terms used in each document have little overlap with others.

The "zsh-list" collection is the archives of a public mailing list discussing development of the zsh shell, a narrow technical topic. The archives used span 12 years and include 9138 documents (email messages). The "procmail" collection is similar, being the archives of six years of a public mailing list discussing the procmail[8] email software. This collection includes 24,482 documents.

"Email" is a third email collection, this time of 24,974 documents from a personal email archive with much broader topics. The archive spans around five years. Spam has been excluded.

"WSJ" (from TREC CD 1) collects several years of contents of the Wall Street Journal, including articles and letters. It covers a broad range of topics, and length varies widely across the 98,732 documents.

either.

[8] http://www.procmail.org/

The largest collection, ".GOV" (from the TREC Web Track), is a 1,247,753 page partial crawl of Web hosts in the `.gov` top-level domain (US government agencies).[9] As with the WSJ collection, document size, style, and topics vary considerably. Documents in the .GOV collection were converted to plain text using lynx.[10]

4.5.2 Parameter settings

In the first instance, each sampler was run with parameters set as originally described by their authors. For the single queries sampler, samples were taken with $k = 100$ and with queries from a 1% subset of terms in each collection. (If all queries were exhausted in the course of a long run, two- and finally three-term disjuncts were used.)

The query-based sampler used the top four results from each query, and stopped when the sample size n had been reached or (following Callan et al. [1999]) when $\lfloor n/2 \rfloor$ queries had been issued. Terms were chosen randomly and uniformly from the learned model.

The pool-based sampler was run with a limit $k = 5$, and a query pool of a 1% subset of 5-grams, or phrases of five terms, from each collection (for .GOV, a 0.1% subset was taken, as the collection is large). The random walk sampler used the same parameters, although it was possible to use a pool of all possible 5-grams since in this instance queries do not need to be enumerated in advance.

The multiple queries sampler used single-term queries, with terms chosen independently of the collection from the SMART stopwords [Salton and McGill 1983]. Each run used 100 queries, with a result limit $k = 10{,}000$, and twenty documents were sampled per run.

The estimated cost of each sampler with these settings is summarised in Table 4.4. The random walk sampler is significantly more expensive than others; however this cost scales with B, the burn-in time. Lower burn-in times are considered in Section 4.6.2 below. Costs for the pool-based and multiple queries sampler are similar, although the multiple queries sampler generally requires fewer interactions and never requires document text.

4.5.3 Tests for bias

Two simple tests were developed to indicate bias in samples.

"T", number of times seen If from a collection of size N we draw i independent samples, each of size n, and each sample is drawn randomly, then the probability of seeing a document t times in the i samples follows a binomial distribution: $\Pr(\text{seen } t \text{ times}) = \binom{i}{t}(\frac{n}{N})^t(\frac{N-n}{N})^{i-t}$. A χ^2 test [Sheskin 2004] compared this expected distribution with observed frequencies from large numbers of samples.

[9] `http://es.csiro.au/TRECWeb/govinfo.html`

[10] `http://lynx.isc.org/`

	Single queries		Query based		Pool based		Random walk		Multiple queries	
Collection	q	d	q	d	q	d	q	d	q	d
calendar	1	0	< 1	1	1	1	1000	1000	34	0
zsh-list	1	0	< 1	1	7	2	1123	1000	5	0
procmail	1	0	< 1	1	8	2	1137	1000	6	0
email	1	0	< 1	1	8	2	1173	1000	5	0
WSJ	1	0	< 1	1	25	4	1161	1000	8	0
.GOV	2	0	< 1	1	11	1	2140	1000	23	0

Table 4.4: Estimated cost in queries (“q”) and document downloads (“d”), per document sampled, for each sampler.

Failure of this test would most likely indicate that some individual documents are being sampled too frequently and others too infrequently — in other words that there is a bias towards some subset of the documents.

“S”, size distribution Test “S” considers one likely source of bias. Earlier work has suggested that the single query sampler, in particular, strongly favours longer documents. This could be due to two factors: a ranking bias, if search engines promote longer documents, or a query bias, since longer documents are more likely to contain some chosen search terms. Ranking bias is controlled in the implementations used here, but there may still be effects of query bias.

To investigate, each collection was divided into deciles according to document length, and the proportion of documents sampled from each decile was noted. A random sample should have documents from all deciles equally represented. As for test “T”, a χ^2 test ($df = 9$) provided comparisons. Failure of this test indicates a bias towards documents of particular lengths; in practice this tends to be towards longer documents.

4.5.4 Exploring parameter settings

Informed by the results of the initial tests, a series of experiments investigated the effects of the available parameters for each sampler. These are described in more detail in Section 4.6.2 below.

4.6 Results

Results showed samplers performed well with only some collections, and that all were subject to some degree of query bias. Altering query pools or other parameters did not provide improvements.

t	Expected	Single queries	Query based	Pool based	Random walk	Multiple queries
0	24381	24540	24431	24389	24381	24378
1	586	333	498	570	586	592
2	7	63	40	15	7	4
3		27	3			
4		4				
5		3				
6		1				
7		1				
8		2				
9			1			
p		$\underline{\leq 0.01}$	$\underline{\leq 0.01}$	$\underline{0.01}$	1.00	0.54

Table 4.5: Email samples for test "T". $i = 30$ samples of $n = 20$ documents each where possible; $i = 600$ and $n = 1$ otherwise. $N = 24,974$ documents. Significant deviations from randomness (i.e. probable non-uniform samples) are underlined. The random walk sampler produces a distribution identical to that expected.

4.6.1 Tests for bias

Test "T", number of times seen Table 4.5 summarises the number of times each document in the email collection was returned by each sampler, and the theoretical distribution described in Section 4.5.3. p is the chance of seeing a distribution this far from the expected if we had a truly uniform sample (χ^2 test, $df = 2$ with $t \geq 2$ counted as one category). The email collection is used only for illustration; similar trends were seen with the other collections tested.

In this example, the single queries sampler has returned fewer documents than expected: 434 unique documents are returned whereas we would expect 593 from a truly random sample. Further, those documents which are sampled are returned too frequently: 63 documents are seen twice (only seven would be expected from a random sample), and two documents are seen eight times each. Together these observations suggest a bias towards a subset of the collection. One form of this bias is considered below.

Table 4.6 summarises test "T" across each collection and sampler. In every case of test "T" failing, the sampler has sampled too few documents too frequently, suggesting a bias towards some part of the collection. This bias is most extreme with the calendar collection, on which every sampler failed: the random walk sampler returned one document 24 times and the pool based sampler returned seven distinct documents more than 15 times each.

The failures on the calendar collection are likely due to the unusual combination of document length and document language. Since documents are very short, the graph on MATCH$_{\mathcal{P}+}$ used by the random walk sampler is much sparser and there is

	Single queries	Query based	Pool based	Random walk	Multiple queries
calendar	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$
zsh-list	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{0.02}$	0.51	1.00
procmail	$\underline{< 0.01}$	$\underline{< 0.01}$	0.34	0.90	1.00
email	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{0.01}$	1.00	0.54
WSJ	$\underline{< 0.01}$	$\underline{< 0.01}$	0.90	0.42	0.92
.GOV	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$

Table 4.6: χ^2 results (p values) for test "T". Significant deviations from randomness (i.e. probable non-uniform samples) are underlined.

	Single queries	Query based	Pool based	Random walk	Multiple queries
calendar	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$
zsh-list	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	0.46	0.94
procmail	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	0.41	0.16
email	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{0.01}$
WSJ	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$
.GOV	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$	$\underline{< 0.01}$

Table 4.7: χ^2 results (p values) for test "S". Significant deviations are underlined.

less opportunity to move from the initial document. For other samplers, there may be strong bias towards the small fraction of documents matching the query pool since the terms used in each document vary considerably.

Test "S", size distribution Figure 4.2 illustrates the distribution of documents sampled by each algorithm from the email collection. A truly random sample would include around 10% from each decile; instead a bias is apparent in all samplers. Table 4.7 has χ^2 results ($df = 9$) for all collections and samplers.

In the vast majority of cases, a failed test is due to a bias towards longer documents. This is likely due to query bias; a longer document is more likely to match any given query, and by ignoring overflowing queries the samplers may be effectively counteracting any length normalisation at the search engine.

As may be expected, the single queries and query based samplers perform poorly on this test, failing on every collection and exhibiting a large bias towards longer documents. (The bias in the single queries sampler is particularly large.) The pool based sampler also fails in all six cases, which seems to be due to the size of the query pool: a uniform subsample of 1% of 5-grams would cover a small proportion of documents and these documents would tend to be longer. If this conjecture is correct, a larger pool (although more difficult to construct) would result in less bias. Section 4.6.2 below explores this possibility by varying query pools.

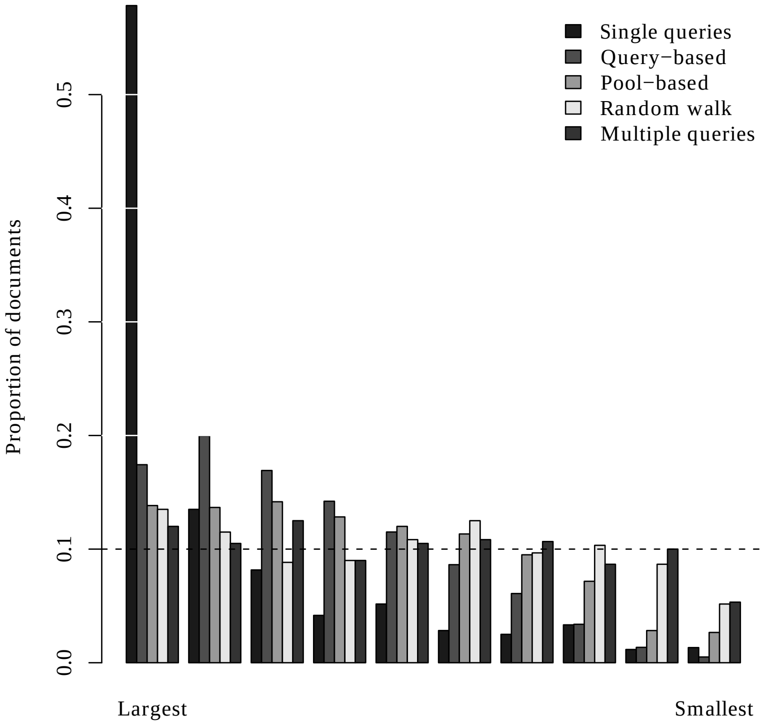


Figure 4.2: Email samples by size decile for test "S". A uniform sample would have 10% of sampled documents in each decile.

The random walk sampler fails in several cases again with a bias towards longer documents. It is not clear why this should be, since the sampler explicitly controls for the number of queries a document matches; however differences in computing $\textsc{match}_{\mathcal{P}}(d)$ at the server and at the sampler may give rise to a bias of this nature. Since the sampler must agree with an unknown server at each step of query processing, including tokenisation, stemming and stopping, and parsing queries, errors of this kind are very likely and may explain the observed bias.

The larger number of documents considered by the mutiple queries sampler provides some improvement on the single queries sampler, and for the zsh-list and procmail collections suffices to overcome query bias. In other cases a significant bias to longer documents remains, although this new sampler performs no worse than the random walk sampler and with significantly lower runtime.

4.6.2 Exploring parameter settings

The biases described above may be due in part to the particular parameters chosen for each sampler. For each sampler, a series of experiments varied the available parameters to investigate possible improvements.

Single queries

The single queries sampler proved particularly susceptible to query bias. Since the sampler ignores all queries which overflow, varying k will change the subset of queries used and may affect the quality of the resulting sample; so may varying the source of queries.

Tests "T" and "S" as above were re-run for the single queries sampler with $k = 100$, 500, and 10,000; and with queries chosen from a subsample of terms indexed, from a subsample of 5-grams, and from the collection-independent set of common English words. In each case the sampler performed poorly, again returning too few unique documents and with a strong bias towards longer documents. The single queries sampler is therefore unlikely to avoid bias for any likely combination of parameters.

Query based

Callan et al. [1999] suggest that the source of initial query term makes little difference to the query based sampler, which leaves three possible parameters: the cut-off r, which determines how many documents are downloaded with each iteration; the stopping criteria; and the method for selecting terms from m. Since in this instance the number of documents to sample, and hence the stopping criteria, is fixed by the needs of tests "T" and "S", there are two parameters remaining. Each of these was varied.

A first set of experiments modified the rank cut-off r. As r increases, there are fewer feedback loops between the learned model and future queries and the query-based sampler acts more like the single queries sampler. (At the limit, when $r \geq n$ there will only be one query issued, if the result set is large enough, and the samplers are identical.) Raising r then is unlikely to improve the sampler's performance, and with r changed from 4 to either 10 or 100 the samples still failed test "T" in almost all cases.[11]

Reducing r may be expected to improve the quality of the samples, since more queries will be issued and this may increase coverage of the collection. A smaller cutoff however will also make the query-based sampler more susceptible to query bias, and this is borne out by experiments: with $r = 1$, a lower bound, test "T" still indicates unacceptable bias.

A second set of experiments varied the strategy for selecting query terms from the learned model m. Three variants were considered (recall that the first experiments used uniform selection): selection weighted by the document frequency (df) in m, selection weighted by the total term frequency (tf), and selection weighted by the mean term frequency (tf / df). Each variant again returned a biased sample from each collection; however the samples generated with the tf-based strategies appeared still more biased. This is consistent with results reported by Baillie et al. [2006a].

Although it appears that the query-based sampler is unlikely to produce random samples given any parameter settings, this may not be a fair test since the sampler was not originally intended for this purpose. Later chapters therefore reconsider query-based samples in settings where they have been used: Chapter 5, following, considers collection size estimation and Chapters 6 and 7 consider language models and server selection.

Pool based

In the initial experiments a consistent bias was observed in the pool based sampler towards longer documents. This may have been due to the small query pool (1% of 5-grams for most collections). With a larger pool of 10% of 5-grams, results improved for both tests and most collections, which is consistent with this conjecture.

While a larger pool improves results, pools here were generated by pre-processing the collections used. It is not clear how a working system might generate pools except from document text; since any bias in the pool will result in biased samples, documents for the pool should be returned by an unbiased sampler. This leaves a difficult bootstrapping problem.

Further experiments varied k, using the original pool of 1% of all indexed 5-grams. Improvements over $k = 5$ were seen with $k = 50$, and further improvements were seen with $k = 500$ across most collections. These samplers however have significantly increased runtime, as k increases much faster than $\overline{|\textsc{res}(q)|}$.

Random walk on $\textsc{match}_{\mathcal{P}+}$

The quality of the samples generated by the random walk sampler seem to depend strongly on B, the burn-in period of the walk. The initial experiments used $B = 1000$,

[11] In contrast, Baillie [personal communication] found improved coverage with higher r across a TREC-based testbed. The samples were not however tested for bias.

as originally specified, and followup experiments used values of $B = 100$ or 10. With smaller values of B bias increased markedly; this bias was more apparent with larger collections. The choice of seed for the walk is strongly effected by query bias, so this result can be seen as a straightforward result of the sampler taking fewer steps from this initial biased choice. With some knowledge of the size of the collection and the connectivity of the $\textsc{match}_{\mathcal{P}}$ graph it would be possible to choose a minimal useful B, but it is not clear how this could be derived without prior knowledge.

Multiple queries

As for the single queries sampler, the multiple queries sampler is sensitive to the value of k. Samples were taken with k set to 10,000, 1000, and 100; in general, results were somewhat worse with $k = 1000$ and deteriorated markedly with $k = 100$. This is expected: as k decreases, only queries which are more specific will not overflow. This leads to a smaller number of documents from which to take a final sample.

This observation may also explain variation seen with different query sources. With collection-specific query pools of 1% of terms, the multiple queries sampler showed more bias. For the data here — documents of varying size and format but all in English and most with full sentences — a list of common words seems to be more convenient and also produces better samples.

4.7 Conclusions

Several key methods for metasearch rely, explicitly or implicitly, on an unbiased sample of documents from constituent collections. Techniques for generating these samples, given only a query interface and an otherwise uncooperative server, must contend with biases due to server optimisations ("ranking bias") and document content ("query bias"). It appears possible to eliminate ranking bias but query bias is persistent across a variety of samplers and collections.

This chapter has described nine samplers. Of these, five are applicable to document types without hyperlink structures and were tested across six collections representing a range of sizes and document types; the five "general" methods are summarised in Table 4.8. No sampler performed well across all collections.

The single queries and query-based samplers are very badly affected by query bias, and consistently prefer a small set of mostly longer documents across all collections tested. The pool based sampler as initially described performs poorly, but depends greatly on the choice of pool — with a larger sample of document text the samples generated are of higher quality. It is not clear, however, how a real-world system can generate such a query pool without prior knowledge of collection contents.

The random walk sampler performs better than most alternatives, but has a much higher cost as measured in interactions with the server. It also requires some knowledge of how queries are processed at each server; any errors in assumptions here are reflected in biased samples. The new multiple queries sampler can produce samples

Method	
Single queries	Badly affected by query bias; very cheap.
Query-based	Needs access to document text; badly affected by query bias; very cheap.
Pool-based	Needs access to document text; choice of pool is important.
Random walk on MATCH$_{\mathcal{P}+}$	Needs access to document text; best quality samples; extremely expensive.
Multiple queries	Needs support for large result sets; comparable quality to random walk.

Table 4.8: Summary of sampling methods.

of comparable quality, with fewer interactions and no prior knowledge of the collection being sampled, but requires servers to return large result sets.

Document samples are often used to determine collection size, one of the most important characteristics of a server for a metasearch system. Experiments in the next chapter consider these samples, and a variety of size estimation techniques; they show that the quality of a sample is of great importance in obtaining an accurate size estimate, and that for many real-world tasks the samples from the multiple queries method perform almost as well as a truly random sample. Results in Chapter 6 and 7 also demonstrate the importance of good-quality samples in language modelling and server selection.

Chapter 5

Server characterisation: size

Size — meaning the number of documents included — is an important characteristic of any collection used in metasearch. Collection size is a proxy for coverage and comprehensiveness, and is important input to many server selection methods including extended KL-divergence, three variants of CORI, and ReDDE [Callan et al. 1995; Si and Callan 2003a; Si and Callan 2003b]. Collection size is also needed to normalise term statistics for language models; these language models are used in server selection as well as result merging [Craswell et al. 1999].

In some instances, servers may report a collection size. It is likely however that a metasearch engine would rely on its own size estimates; few servers are likely to report this information, and in many instances reported figures may be inaccurate (for example, the rough numbers reported by Web search engines) or deliberately misleading.

Several algorithms have been proposed for estimating collection size in a metasearch environment with uncooperative servers, or with servers with a basic set of features. These have been tested previously either with perfectly random samples, which are hard to obtain, or with samples from query-based sampling, which was earlier demonstrated to be biased — neither is likely to give an accurate picture of their performance in a personal metasearch tool. Further, these algorithms have been tested on TREC data or on the Web, neither of which corresponds to the full range of collections in a personal context.

Section 5.1 below describes six methods for estimating collection size, as well as two variants and related work. These methods are evaluated in Sections 5.2 and 5.3, using the personal metasearch testbed of the previous chapter with samples of varying size and quality. Although no method for size estimation performs well in all conditions, broad trends suggest that a small number of methods may be useful in personal metasearch.

In the discussion following, the set of all documents in a collection is denoted $\mathcal{D}$. N is the size of this set (so $N = |\mathcal{D}|$), and is the quantity to estimate.[1] As elsewhere, methods presented in this chapter make as few assumptions as possible about each constituent server and the interface available.

[1] This and other notation is summarised in Appendix A.

	Information needed			
Method	Random sample	Model	Query pool	Doc. text
Capture-recapture (CR)	Needed	—	—	—
Multiple capture-recapture (MCR)	Needed	—	—	—
Capture history (CH)	Needed	—	—	—
Sample-resample (SR)	—	Needed	—	—
Basic estimator methods				
Random documents (BER)	Needed	—	Needed	Needed
Uncorrelated query pools (BEU)	—	—	Needed	Needed

Table 5.1: Characteristics of size estimation methods.

5.1 Related work

A number of methods for size estimation have been suggested in the literature, following two general approaches. The first approach is to take two or more random samples from the collection, and count the number of documents in common between samples; then, using a theoretical distribution, derive an estimate of the collection size. This is the intuition behind the capture-recapture, multiple capture-recapture, and capture history methods, described below.

The second general approach relies on estimates of term distribution, either from models of a collection or from repeated samples. The sample-resample, random documents, and uncorrelated query pools methods below use this approach.

In this section six methods are described; they are summarised in Table 5.1 below. Two further variants described in Section 5.2 are also used in the experiments of this chapter.

5.1.1 Capture-recapture

The most straightforward estimator has come to be called the "capture-recapture" method. It has long been used in demographics and biological science[2] and was introduced to information retrieval by Liu et al. [2001]. The method proceeds from the observation that given a random sample of n documents from a collection of size N, taken without replacement, the probability of any one document being included is n/N. Given a second, independent, sample of n documents from the same collection, the expected number of documents which are in both ("overlap") is

$$\mathrm{E}(O) = \frac{n^2}{N}. \tag{5.1}$$

(In this and the following section, the capital letter O represents a random variable and a lowercase o represents a particular value of this variable.) From a count o of the

[2] Ricker [1975] credits the first use to John Graunt in 1662, estimating mortality rates in London.

number of overlapping documents, an estimated collection size can be calculated as[3]

$$\hat{N} = \frac{n^2}{o}. \tag{5.2}$$

For example, if two 300-document samples from a collection have four documents in common, the estimated size of this collection will be $\hat{N} = n^2/o = 300^2/4 =$22,500 documents.

It is evident that the capture-recapture technique can never produce estimates of more than n^2 documents and therefore requires samples of at least size $\sqrt{N}$ documents. This is a serious drawback: unless the approximate size of a collection is known in advance, the sample size n can itself only be estimated — larger values of n would tend to reduce error but incur greater cost. The technique could, however, be extended to gradually increase n over several runs.

The capture-recapture technique was tested by Liu et al. [2001] with collections of 100,000 to 300,000 documents from TREC testbeds, and with sample size n from 1000 to 4000 documents (0.3% to 4% of the total collection). With samples chosen at random, observed errors were around 3–5%.

Two weaknesses in the evaluation were identified and addressed by Shokouhi et al. [2006], although they did not repeat the experiments in this case. First, the samples used by Liu et al. were true random samples, which as we have seen in Chapter 4 are extremely difficult to obtain in practice; Shokouhi et al. [2006] instead suggest using samples generated by query-based sampling [Callan et al. 1999]. Secondly, Liu et al. did not consider what sample sizes were needed for accurate estimates. These weaknesses are considered in the experiments in this chapter.

5.1.2 Multiple capture-recapture

Shokouhi et al. [2006] introduce two additional methods based on techniques in ecology, both of which use repeated samples to improve size estimates. The first, "multiple capture-recapture", is an elaboration on capture-recapture and considers several iterations of sampling, each time choosing n documents randomly from a collection of size N.

Recall that given two samples of size n the expected number of documents o in the overlap is $\mathrm{E}(O) = n^2/N$ (Equation 5.1). Given T independent sampling iterations, there are $T(T-1)/2$ pairs and so the expected total number of overlaps is then $T(T-1)/2 \times n^2/N$. Given a count o of overlaps between T independent samples, by rearranging it is possible to obtain an estimated collection size with

$$\hat{N} = \frac{T(T-1)n^2}{2o}.$$

If there are two samples ($T = 2$), this is identical to simple capture-recapture.

[3] Bailey [1951] notes that this is not in fact the best estimate, especially if o is small, and suggests instead $\hat{N} = (n(n+1))/(o+1)$. This estimate seems not to have been used in information retrieval experiments.

Tests of the technique used seven collections of 31,000 to 817,000 documents from TREC, and 100 to 5000 samples each of 10 documents collected with query-based sampling [Callan et al. 1999]. Multiple capture-recapture was found to consistently underestimate the true collection size, which Shokouhi et al. attribute to bias in the samples used; this is consistent with results reported in Chapter 4 earlier. With this in mind a correction was suggested for size estimates produced with the capture-recapture method, and using this correction Shokouhi et al. report errors in estimation of 10–79% given 5000 samples of 10 documents each.

5.1.3 Capture history

The Schumacher-Eschmeyer technique [Schumacher and Eschmeyer 1943], originally introduced for estimating the population of fish stocks, builds on the simple capture-recapture estimate. If each sample is the same size n, Equation 5.2 can be recast as

$$\hat{N} = \frac{Mn}{o},$$

where M is the number of documents seen ("marked") in the first sample and n and o are as before.[4] As with multiple capture-recapture, the Schumacher-Eschmeyer technique uses repeated samples to improve estimates. For each iteration i we plot o_i/n, the fraction of documents in the i-th sample which overlap previous samples, against M_i, the number of distinct documents seen in iterations $1 \ldots\ i-1$ (Figure 5.1).

Note that at the limit $M_i = N$, when the entire collection has already been seen, every document in the sample will have been seen, o_i/n must be 1, and therefore a line fitted to the plotted points but passing through the origin will have a slope around $1/N$. This allows us to estimate

$$\widehat{\left(\frac{1}{N}\right)} = \frac{\sum_i o_i M_i}{\sum_i n M_i^2} \tag{5.3}$$

Shokouhi et al. [2006] use this method, which they call "capture history", for estimating the size of a collection. Using Equation 5.3 gives

$$\hat{N} = \frac{\sum_i n M_i^2}{\sum_i o_i M_i}$$

where o_i is the number of documents in sample i which have already been seen in earlier samples, and M_i is the number of distinct documents seen in all samples up to, but excluding, the i-th.

The technique was tested on the same collections as for the multiple capture-recapture method above, using 100 to 5000 samples each of 10 documents. Again, the technique consistently underestimated the true size, and a correction was suggested for this bias. With the correction, errors of 8–78% were reported.

[4] A good summary of this and other estimation techniques can be found in Ricker [1975].

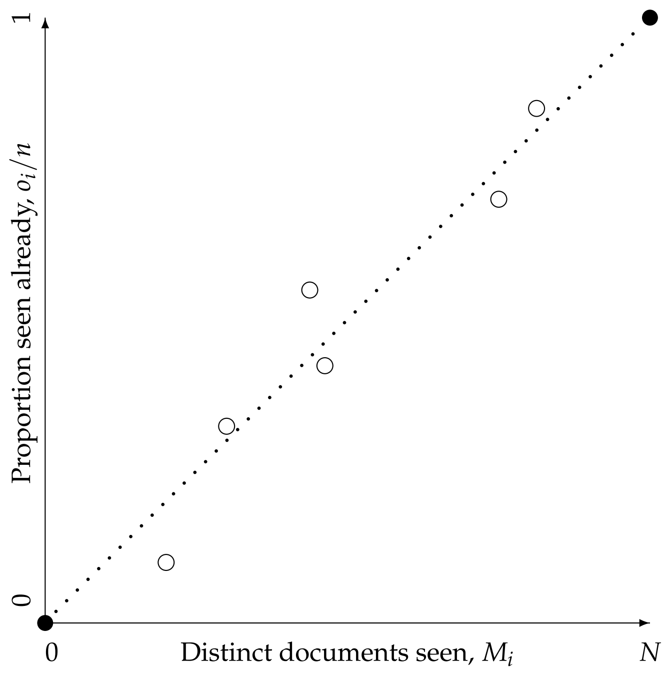


Figure 5.1: The Schumacher-Eschmeyer (capture history) technique. The i points $(M_i,\ o_i/n)$ will fall around a line with slope $1/N$.

In this case, and for the multiple capture-recapture estimates above, Shokouhi et al. report results from only one run on each collection and do not offer analysis of the distribution of errors. A Wilcoxon test [Sheskin 2004] of the published data suggests neither method is significantly better than the other ($T = 3$, $n = 7$, $p = 0.08$). This is consistent with the results reported in Section 5.3 below.

5.1.4 Sample-resample

Si and Callan [2003b] introduced the "sample-resample" method as part of the ReDDE technique of server selection (discussed further in Section 7.1.9 on page 104). This method takes the second common approach, based not on repeated random samples but estimates of term frequency.

The sample-resample algorithm starts by assuming a subset, or model, of the collection is available; this model is generated from a size s subset of the documents in the collection, and for any query q returns a count $\mathrm{df}_m(q)$ of the number of documents sampled which match q. (These simple models are discussed further in Chapter 6 following.)

To obtain an estimate of collection size, a query q is sent to the appropriate server. It is assumed that the server returns, as well as any other information, the number of matches: this is the document frequency of the query, $\mathrm{df}(q)$. Now the probability that any one document, chosen at random from the whole collection, matches q is $\mathrm{df}(q)/N$;

and the probability that any one document from the model matches q is $\mathrm{df}_m(q)/s$. If the model is a good representation of the whole collection, these probabilities are approximately equal and (after rearranging) the collection size can be estimated with

$$\hat{N} = \frac{\mathrm{df}(q)}{\mathrm{df}_m(q)} s.$$

To improve accuracy, Si and Callan [2003b] choose five different queries and use the mean of all five results as the final estimate $\hat{N}$.

Evaluation experiments by Si and Callan used two testbeds: trec123-100col, which is composed of 100 "small" collections (700–40,000 documents) from TREC CDs 1–3, and trec123-10col, which is composed of 10 "large" (17,000–263,000 document) collections from the same data (Section 4.5.1). Using models built by query-based sampling [Callan et al. 1999], the mean error ranged from 23 to 30%. The capture-recapture algorithm, evaluated over the same testbed and using a comparable number of interactions with the server, had mean errors in the range 18–94% although later experiments over a different testbed showed errors of up to 542% [Shokouhi et al. 2006].

Sample-resample is probably the most commonly used technique for collection size estimation in the literature [Avrahami et al. 2006; Hawking and Thomas 2005; Nottelmann and Fuhr 2003]. There are obstacles to using it in a real system, however. First, it requires a model of the collection, and one which can accurately predict term frequencies across the whole collection. As shown in Chapter 4, this is a non-trivial exercise.

Secondly, the technique requires some cooperation from the server in that it requires a report of $\mathrm{df}(q)$ for any query q. While this information is commonly reported, for example by Web search engines, it is not universal nor reliable [Anagnostopoulos et al. 2005]. Poor estimates, errors, or deliberate misreporting at the servers themselves will necessarily distort size estimates.

Finally, this technique is accurate only to the extent that a metasearch tool is able to process queries in the same way as each server. Differences in stemming and stopping, for example, or in interpreting queries, will violate the assumption that frequencies in the model approximate those in the collection and will result in poorer estimates. In general, it is not possible to predict exactly how a server will handle any given query.

5.1.5 Basic estimator methods

Broder et al. [2006] describe two techniques derived from a "basic estimator". Both techniques need a query pool, or set of queries, $\mathcal{P}$; this pool must be of known size, it must be possible to sample randomly, it must cover at least some of the documents in the collection, and given the text of a document d it must be possible to determine $\mathrm{MATCH}_{\mathcal{P}}(d)$, the set of queries from $\mathcal{P}$ which d matches. The set of all five-digit sequences, for example, is a candidate pool since the size is known ($|\mathcal{P}| = 10^5$), it is possible to sample with a simple random number generator, and given the text of a document it is trivial to list all five-digit sequences it contains.

Basic estimator and pool weight

The intuition behind both methods runs as follows. If we are interested in knowing the size of $\mathcal{D}_\mathcal{P}$, that subset of $\mathcal{D}$ which matches one or more queries in $\mathcal{P}$, then it is a simple matter to take a random sample of queries from $\mathcal{P}$; issue each query in turn to the server; and count the documents returned. Multiplying the number of documents seen in these sample queries by the ratio of pool size to sample size will then give an overall estimate.

Each document, however, can be covered by more than one query in the pool and if counted more than once will bias the estimate. To avoid multiple-counting these documents, Broder et al. define the *weight* of a document d as $w_d^\mathcal{P} = 1/|\text{MATCH}_\mathcal{P}(d)|$ and sum these instead. The weight of a query q is similarly defined: $w_q^\mathcal{P} = \sum_{d \in \text{RES}(q)} w_d^\mathcal{P}$, which is the sum of the weights of documents matching q. With access to the text of each document, it is possible to calculate $w_d^\mathcal{P}$ and hence $w_q^\mathcal{P}$ for any query.

Finally, the weight of a query pool $\mathcal{P}$ is defined as

$$W_{\mathcal{P},\mathcal{D}} = \mathrm{E}_{q \in \mathcal{P}}(w_q^\mathcal{P}), \tag{5.4}$$

which is the expected (or mean) weight of a query from $\mathcal{P}$. $W_{\mathcal{P},\mathcal{D}}$ will not be known without issuing every query in $\mathcal{P}$, but can be estimated by selecting a random set of queries and taking the average weight. Broder et al. show that

$$W_{\mathcal{P},\mathcal{D}} = |\mathcal{D}_\mathcal{P}|/|\mathcal{P}|, \tag{5.5}$$

which is the "basic estimator" used below.

As with the pool-based sampler of Section 4.2.3, errors will be introduced if the underlying search engine interprets queries differently to the metasearch engine — for example, due to differences in stemming or stopping. These methods also need access to the text of documents, which may make them impractical in some cases.

Random documents

The first method using the basic estimator assumes it is possible to sample documents randomly from the collection.

For any pool $\mathcal{P}$, some proportion of documents in $\mathcal{D}$ will match one or more queries. If we call this proportion $p_\mathcal{P} = |\mathcal{D}_\mathcal{P}|/|\mathcal{D}|$, it can be shown that

$$\begin{aligned}\frac{|\mathcal{P}|}{p_\mathcal{P}} W_{\mathcal{P},\mathcal{D}} &\overset{(5.5)}{=} \frac{|\mathcal{P}|}{|\mathcal{D}_\mathcal{P}|/|\mathcal{D}|}\frac{|\mathcal{D}_\mathcal{P}|}{|\mathcal{P}|} \\ &= |\mathcal{D}| \\ &\overset{\text{def}}{=} N.\end{aligned}$$

$p_{\mathcal{P}}$ is easy to estimate given a random sample of documents. Given an estimate of $W_{\mathcal{P},\mathcal{D}}$ as in Equation 5.4, therefore, the total collection size can be estimated by

$$\hat{N} = \frac{|\mathcal{P}|}{\widehat{p_{\mathcal{P}}}} \widehat{W_{\mathcal{P},\mathcal{D}}}.$$

The .GOV collection, with 1.2M documents, served as a testbed for Broder et al. [2006] to evaluate the technique. All five-digit sequences formed a pool $\mathcal{P}$. With 100 samples from $\mathcal{P}$, the relative error was 49%; with 5000 samples the relative error was reduced to 6%.

Five-digit sequences do not all have equal weight $w_q^{\mathcal{P}}$, and the variance in weight introduces further error in the final estimate. To reduce this variance, a random sample of documents (the size of which was not reported) was used to find the 25% of queries in $\mathcal{P}$ with lowest estimated weight. Using samples only from this smaller pool, errors in $\hat{N}$ were further reduced to 14% with 100 samples and less than 1% with 5000.

Using a different pool of 100,000 medium-frequency terms from .GOV provided better results, with errors from 4% with 100 samples down to 2% with 5000. No variance reduction was tested in this case.

Errors for the random documents method are small and amongst the best reported to date; further, this estimator requires many fewer queries than capture-recapture. Again, however, it relies on the availability of random samples, and as we have seen in Chapter 4 it is unlikely that such samples are available. Further, unlike most other methods (but like the uncorrelated query pools method below) it requires access to document text to calculate $\text{MATCH}_{\mathcal{P}}(d)$. In most cases, this will be possible but it is not in the general case.

The choice of query pool would also seem important to the success, or otherwise, of the random documents method. A query which overflows and which has a result set constrained by some maximum count will give poor estimates of $p_{\mathcal{P}}$, and as demonstrated by Bharat et al. the variance in weight between queries in the pool will affect the quality of the estimate. Without some knowledge of the collection it is not clear how an appropriate pool can be selected.

Uncorrelated query pools

The second method of Broder et al. [2006] does not assume a random sample is available. The "uncorrelated query pools" technique assumes instead that for any collection, there exist two query pools $\mathcal{P}$ and $\mathcal{Q}$ which are uncorrelated (so the probability of a document matching a query from $\mathcal{P}$ is not related to that of the same document matching a query from $\mathcal{Q}$). If this is the case,

$$\hat{N} = \frac{|\mathcal{D}_{\mathcal{P}}|\,|\mathcal{D}_{\mathcal{Q}}|}{|\mathcal{D}_{\mathcal{P}} \cap \mathcal{D}_{\mathcal{Q}}|}. \tag{5.6}$$

From the basic estimator of Equation 5.5, $|\mathcal{D}_\mathcal{P}|$ can be estimated by $|\widehat{\mathcal{D}_\mathcal{P}}| = |\mathcal{P}|\,\widehat{W_{\mathcal{P},\mathcal{D}}}$. An estimate of $|\mathcal{D}_\mathcal{Q}|$ follows the same lines.

To estimate $|\mathcal{D}_\mathcal{P} \cap \mathcal{D}_\mathcal{Q}|$, Broder et al. note that

$$\begin{aligned} \frac{|\mathcal{D}_\mathcal{P} \cap \mathcal{D}_\mathcal{Q}|}{|\mathcal{P}|} &= W_{\mathcal{P}\mathcal{Q},\mathcal{D}} \\ &= \mathrm{E}_{q\in\mathcal{P}} \left(\sum_{d\in(\mathcal{D}_\mathcal{P}\cap\mathcal{D}_\mathcal{Q})} w_d^\mathcal{P} \right); \end{aligned}$$

or the mean weight of documents that match at least one query from each of $\mathcal{P}$ and $\mathcal{Q}$. Combining the estimates of $|\mathcal{D}_\mathcal{P}|$, $|\mathcal{D}_\mathcal{Q}|$, and $|\mathcal{D}_\mathcal{P} \cap \mathcal{D}_\mathcal{Q}|$ in Equation 5.6 gives an estimate of N.

This technique is similar to capture-recapture (Section 5.1.1 and Equation 5.2), but with a pool taking the place of individual queries.

As with the random documents method, errors will be introduced with differences in query or text parsing. Further errors in estimation may be due to correlation between $\mathcal{P}$ and $\mathcal{Q}$; Broder et al. provide bounds for this error, which decreases as the correlation decreases. If we let $p_{\mathcal{P}|\mathcal{Q}} = \Pr(d \in \mathcal{D}_\mathcal{P} \mid d \in \mathcal{D}_\mathcal{Q})$, or the conditional probability of a document matching $\mathcal{P}$ given it matches $\mathcal{Q}$, then when

$$\begin{aligned} c_1 p_\mathcal{P} \leq\ & p_{\mathcal{P}|\mathcal{Q}} \leq c_2 p_\mathcal{P} \quad \text{, where } c_1 \text{ and } c_2 \text{ are arbitrary constants in } [0,\infty], \\ c_1 W_{\mathcal{P},\mathcal{D}} \leq\ & \quad N \quad \leq c_2 W_{\mathcal{P},\mathcal{D}}\,. \end{aligned}$$

Note that when $\mathcal{P}$ and $\mathcal{Q}$ are entirely uncorrelated, $p_\mathcal{P} = p_\mathcal{Q} = p_{\mathcal{P}|\mathcal{Q}}$, $c_1 = c_2 = 1$, and the estimate is accurate.

Evaluation again used the .GOV collection. The two pools used for the evaluation of the random documents method proved too correlated, so two alternative pools were defined: documents with *exactly one* five-digit sequence and documents with exactly one medium-frequency word. Errors ranged from 28%, with 1000 samples, to 21% with 5000. Although this is significantly more error than seen with random documents, no random samples are assumed; however a pair of uncorrelated query pools is required and again this suggests either some knowledge of the collection or a pre-processing stage to investigate correlation.

5.1.6 Relative sizes and overlap

Rather than estimate the absolute size of a collection, Bharat and Broder [1998] attempt to estimate relative sizes of two or more servers indexing the same collection (for example, two or more Web search engines). They suggest taking a random sample from each server, then using a checking procedure for each sampled document from one server to determine whether it is indexed by the other; then it is possible to say, for example, that 20% of the documents indexed by server A are also indexed by server B. By combining two of these ratios, and assuming independence, an estimate of relative size follows. Further, if the size of one overlapping server is known then the size of any other can be estimated directly.

This technique is useful, if random or near-random samples are available, for estimating the relative size of Web search engines. In metasearch more generally, however, it will not be the case that servers are independently indexing the same collection and the technique is therefore not considered here.

In a similar vein, Henzinger et al. [2000] use their PAGERANK-SAMPLE algorithm to estimate the relative, but not absolute, size of eight whole-of-Web search engines. Lawrence and Giles [1998, 1999], Gulli and Signorini [2005], and Bar-Yossef and Gurevich [2006] also investigated the relative size of Web search engines. Again, all of these studies assumed that each search engine indexed an overlapping subset of the same collection.

5.2 Size estimation experiments

Although each estimator described above has been evaluated when introduced, there does not appear to be any published work which compares them all across the same testbed. The experiments reported here do this, and consider the following questions:

1. How do the size estimates compare across methods, and across collections?

2. How do the size estimates vary given samples of different sizes and different quality?

3. Which, if any, of the methods is appropriate for personal metasearch?

The six collections used, which are typical of those likely to be included in a personal metasearch tool, are described in Section 4.5.1 on p. 40. These testbeds vary significantly from those used in earlier work; in particular, they cover a much wider range of sizes than previously considered, including a collection (.GOV) with more documents than any collection used by Liu et al. [2001] or Shokouhi et al. [2006].

Six methods are considered: capture-recapture, sample-resample, multiple capture-recapture, capture history, and the two basic estimator methods. In addition, following Shokouhi et al. [2006], corrected variants of multiple capture-recapture and capture history are included. These are represented by starred forms (such as "capture history*") in the results below, and are defined by $\widehat{N^*} = 10^a$, where

$$a = \frac{\log_{10} \hat{N} - 1.5767}{0.5911}$$

for multiple capture-recapture and

$$a = \frac{\log_{10} \hat{N} - 1.4208}{0.6429}$$

for capture history.[5] These corrections were derived from observed errors on seven TREC-derived testbeds, but as observed in Section 5.3.1 below may be overfitted.

In keeping with earlier work, *relative error* [Liu et al. 2001] is reported in the results below as an indication of quality. The relative error of a size estimate $\hat{N}$, given the true size N, is $|N - \hat{N}|/N$. (Note that this metric can range above 100% only for overestimates; an extreme underestimate can never produce a relative error greater than $|N - 0|/N$, or 100%.) In cases where no estimate is possible — for example, because there is no overlap in the capture-recapture method — an error of 100% is recorded. Ten runs for each method and collection allow observation of the distribution of these errors.

PADRE [Hawking et al. 2000] was again used for indexing and searching, and no stemming or stopping was performed. HTML was converted into plain text by lynx.

5.2.1 Samples and parameter settings

Each estimator was given three sets of samples, taken from three different sources. The first set were random samples taken uniformly from the entire collection, using knowledge of the extent of each collection; this is not possible in practice and represents an upper bound on the quality of samples and hence on the quality of size estimates. Two more sets of samples were drawn from each collection using the multiple queries sampler (Section 4.3), the best-performing sampler in previous experiments, and the query-based sampler of Callan et al. [1999], which has been used in earlier work.

Each estimator has several parameters. In a first set of experiments (Section 5.3.1), parameters were set as originally described and as summarised above; later experiments (Sections 5.3.2–5.3.4) varied these parameters to investigate the effect on the generated estimates.

5.3 Results

The estimates from the capture-recapture, multiple capture-recapture, and capture history methods depended upon the quality of the document samples used. Sample-resample and the basic estimator methods require more knowledge of the collection, but perform better with poorer samples.

The multiple capture-recapture, capture history, and sample-resample techniques are improved if more interactions are allowed with the server. With good quality samples and sufficient interactions, the multiple capture-recapture and capture history methods in particular can produce better estimates than other techniques.

5.3.1 Baseline methods

The first experiment provided a baseline for comparison. Each method was run with parameters set as described by the original authors.

[5] Personal communication from Milad Shokouhi.

- Capture-recapture used two true random samples of 1000 documents each, regardless of the size of the collection. (Liu et al. [2001] used 1000 to 4000 documents.)

- The multiple capture-recapture and capture history methods both used 100 true random samples of ten documents each. (Shokouhi et al. [2006] used 100 to 5000 samples. The effect of higher numbers of samples is considered in Section 5.3.2.)

- Sample-resample used language models built from a random sample of 300 documents, then five resamples using terms chosen uniformly from the learned model [Callan and Connell 2001; Si and Callan 2003b]. The limit on the number of results, k, was set to 10,000, and any query which overflowed was ignored as these could otherwise limit the estimate $\hat{N}$.

- The random documents method used true random samples of 1000 documents each to estimate $p_{\mathcal{P}}$, then 1000 queries to estimate $W_{\mathcal{P},\mathcal{D}}$. Following Broder et al. [2006], the pool $\mathcal{P}$ was all 100,000 five-digit sequences.

- The uncorrelated query pools method used two sets of 1000 queries each to estimate pool weights. $\mathcal{P}$ was all five-digit sequences, as earlier, and $\mathcal{Q}$ was up to 100,000 medium-frequency terms from each collection.

The input to each method represents the best case: for example, samples are drawn at random and query pools are constructed with full knowledge of the collection. Subsequent experiments reported in Sections 5.3.2 and 5.3.3 use samples of different sizes and more realistic quality, and experiments in Section 5.3.4 examine the effects of other parameters.

Each method was run ten times on each collection, using different samples and models for each run. Errors for each combination of method and collection are summarised in Figure 5.2.

The errors recorded in this baseline experiment are broadly consistent with published results. The multiple capture-recapture and capture history methods perform better here than originally described; in fact they perform as well "as is" as did the corrected versions of Shokouhi et al. [2006]. These methods were originally tested with biased samples, however, and improved performance with random samples is unsurprising. Results also vary for the capture-recapture method on the WSJ collection, where estimates here are worse than originally reported for comparable collections (40% mean error for the 98,732 document WSJ collection against 3–5% reported error for collections of 100,000 to 300,000 documents). Individual runs on the WSJ collection did however have relative errors as low as 1%, and since earlier figures were reported from single runs it is possible that this variance was not noticed.

Given random samples, the capture-recapture method performs well compared with others of greater complexity. Accurate estimates of the 1.2 million document .GOV collection are impossible, however, since with samples of only 1000 documents apiece it is not possible to estimate more than $\hat{N} = 1000 \times 1000 = 1{,}000{,}000$. In five of

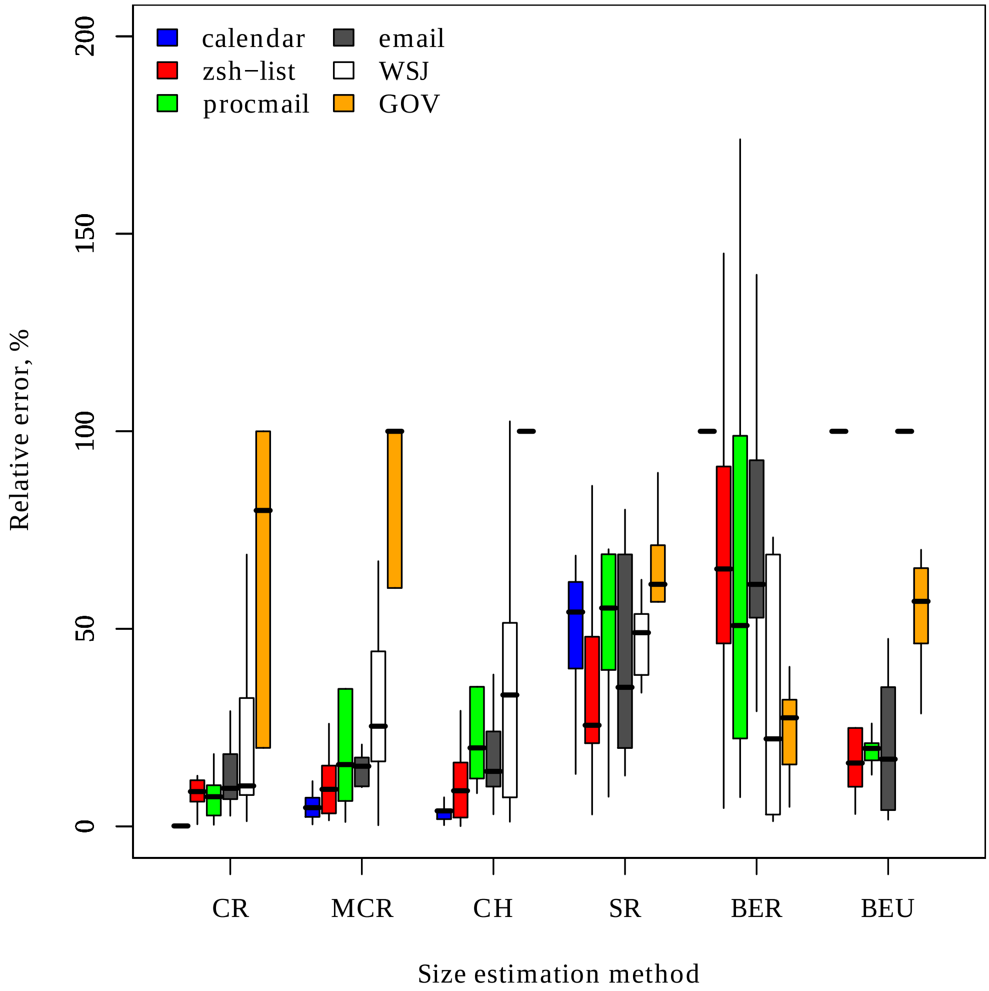


Figure 5.2: Errors in size estimates, given true random samples of sizes specified in earlier work. Methods are capture-recapture (CR); multiple capture-recapture (MCR); capture history (CH); sample-resample (SR); basic estimator with random documents (BER); and basic estimator with uncorrelated query pools (BEU). Ten runs of each method for each collection. In these and subsequent plots, the thick line marks the median, boxes show the interquartile range, and thin lines show the range. For clarity, outliers more than $1\frac{1}{2}$ times the interquartile range from the box are not plotted. CH, BER, and BEU all failed to return estimates for at least one collection (shown as 100% error).

Collection	t test results
calendar	CR < (CH, MCR) < SR < (BER, BEU)
zsh-list	(CR, MCR, CH) < SR
procmail	CR < BEU < SR
email	(CR, MCR, CH) < (SR, BER); BEU < BER
WSJ	(CR, MCR, CH, SR) < BEU
.GOV	BER < (SR, BEU) < (MCR, CH); BER < CR
Overall	(CR, MCR, CH) < (SR, BER, BEU)

Table 5.2: Size estimates compared, using random samples. "$a < b$" indicates the mean error from method a is less than that from b according to a t test ($\alpha = 0.05$). Each method was run ten times on each collection. See text for comments on false alarms.

the ten runs on the .GOV collection, the capture-recapture method found no overlap at all and failed to produce any estimate.

The multiple capture-recapture method offers no improvement on capture-recapture, and because of the smaller size of each sample is less likely to see overlapping documents; in seven of the ten runs over the .GOV collection there is no overlap between any of the 100 samples and the method fails.

The capture history method also fails in seven runs of ten with the .GOV collection, again since there is no overlap. A larger number of samples improves the chance of overlap and produces better estimates (see Section 5.3.2), but it does not seem possible to choose this parameter without some prior knowledge.

The two basic estimator methods failed to complete on the calendar collection, and the uncorrelated queries method failed to complete on the WSJ collection, as the coverage of the query pool $\mathcal{P}$ was low enough that no documents were found.

Methods compared

The estimates from each method, over each collection, were compared using a series of t tests. Results are given in Table 5.2: each entry of the form "$a < b$" indicates that the mean error from method a is significantly less than that from method b ($\alpha = 0.05$). Note that with $\alpha = 0.05$ and 210 comparisons (six collections and an overall summary, each with 30 pairwise comparisons), we can expect around 10 false alarms. Applying, for example, a Bonferroni correction would dramatically reduce the power of the test. Since the broad trends in Table 5.2 seem consistent, errors in individual comparisons need not be a problem.

If random samples are available, the sample-resample (SR) and random documents (BER) methods perform well with the larger collections such as WSJ and .GOV; for smaller collections, capture-recapture (CR) and capture history (CH) perform well. The uncorrelated query pools method (BEU) can produce estimates without any sample, but performs poorly on smaller collections where the coverage of the two pools $\mathcal{P}$

and $\mathcal{Q}$ is relatively low. Of these methods, only capture-recapture and capture history can produce estimates without access to the text of documents, making them more generally applicable.

Both basic estimator methods failed to produce an estimate of the calendar collection. This is due to the definition of the pool $\mathcal{P}$ used by both: there are no documents in the calendar collection which contain even a single five-digit number, so $W_{\mathcal{P},\mathcal{D}}$ and thus $\hat{N}$ will always be zero. Although this may be an unusual case, it is not possible in general to know whether a pool will cover any documents in any given collection. Similarly, unforseen characteristics of the WSJ collection prevented the uncorrelated query pools method from producing an estimate (there is no overlap between documents in each pool). These methods could only be used with enough prior knowledge of the collection to choose a reliable pool.

Section 5.3.4 discusses experiments with alternative pools for both basic estimator methods.

Corrected variants

Figure 5.3 summarises relative error for each collection for the multiple capture-recapture (MCR) and capture history (CH) methods, with and without the correction of Shokouhi et al. [2006]. With true random samples, the correction clearly introduces error: the estimates are significantly worse in all cases except the .GOV collection, for both methods, and the procmail collection for the capture-history method (t test, $\alpha = 0.05$). This is expected, since the correction is designed to counter systematic bias in the sampling method and in this case there is none. Section 5.3.3 below considers the correction with more realistic samples.

5.3.2 Number of interactions

The multiple capture-recapture and capture history methods rely on a series of samples, and the sample-resample method uses a number of resamples to improve the final estimate. A second series of experiments considered whether the estimates from these methods improved if they were allowed more interactions with constituent search engines.

Results are summarised in Figure 5.4, which plots the mean error for each of ten runs on the six collections against the number of interactions allowed. For the multiple capture-recapture (MCR) and capture history (CH) methods, it is assumed that each sample requires only one interaction (so this is the best case); for the sample-resample method (SR), it is also assumed that building a model from 300 documents requires a further 301 interactions (one to generate a sample of 300 documents, and then one to download each).

For reference, Figure 5.4 also indicates the overall mean relative error for the remaining three methods. The capture-recapture (CR) method uses only two samples; the basic estimator methods (BEU and BER) use two much larger samples each, and require additional interactions to download the text of each document sampled.

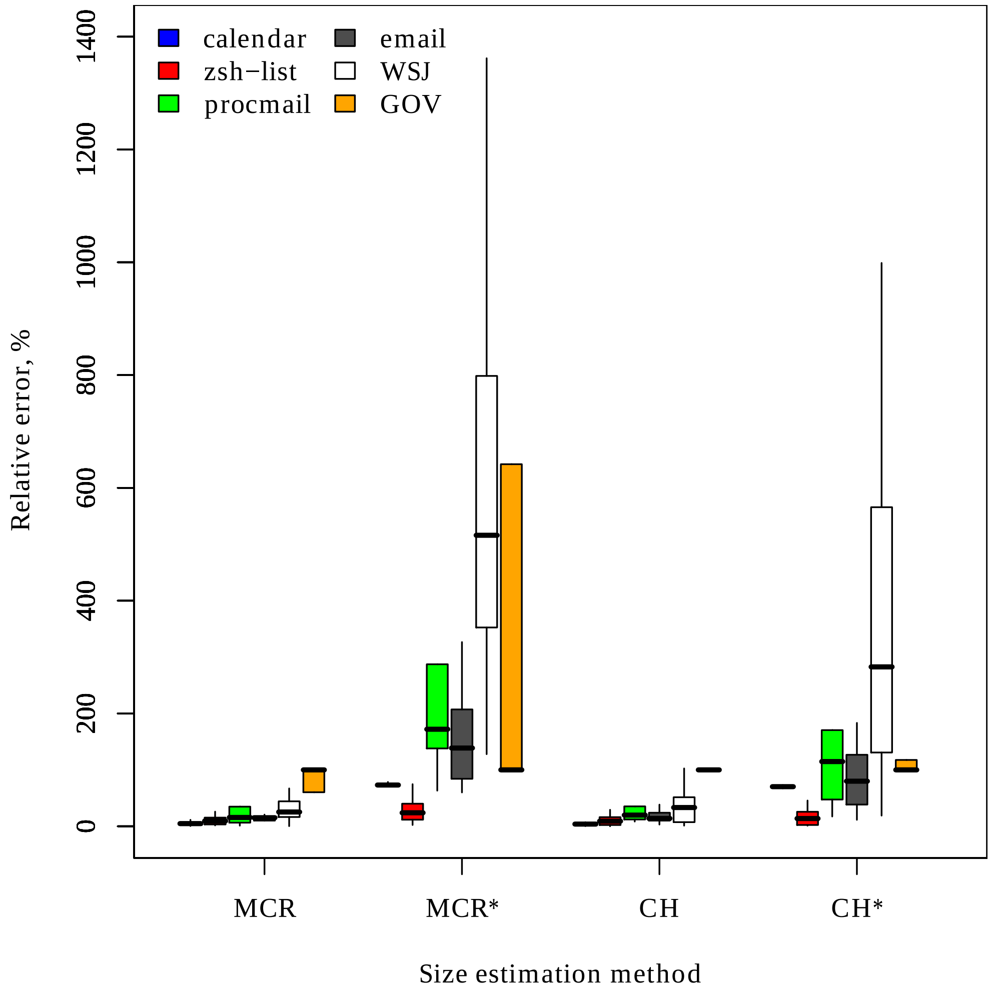


Figure 5.3: Errors in size estimates, given true random samples, with and without the correction of Shokouhi et al. [2006]. Methods are multiple capture-recapture (MCR) and capture history (CH); starred forms use the correction. Ten runs of each method for each collection. See notes to Figure 5.2 on p. 63 for plotting conventions.

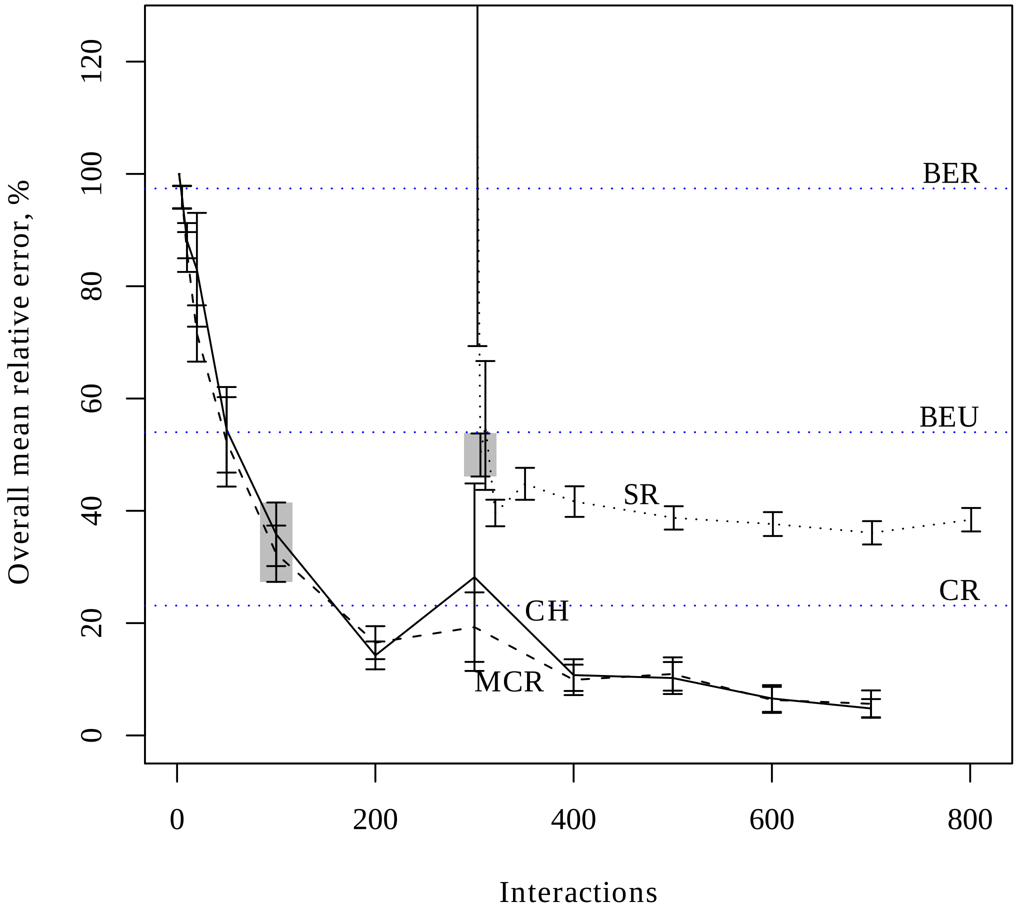


Figure 5.4: Improvement in size estimates from the multiple capture-recapture (MCR), capture history (CH), and sample-resample (SR) methods with more interactions. Each point is the overall mean relative error from ten runs on each of six collections; shaded points are the default number of interactions. Bars are ±1 standard error. Also plotted for reference are overall mean relative errors from other methods, which use a fixed number of interactions.

Errors from the multiple capture-recapture and capture history methods appear sensitive to the number of interactions allowed. With more interactions than the baseline 100, errors are reduced; with fewer, they increase dramatically. Using 400 random samples, the overall mean error from either method is significantly lower than that from capture-recapture or either of the basic estimator methods (one-sided t test, $\alpha = 0.05$).

A similar pattern is seen for the sample-resample method. As originally presented, it uses the mean of five estimates as a final size; allowing for the interactions needed to build the model, this represents 306 interactions. Allowing more interactions improves the final estimate, and with 321 interactions (20 resamples) the overall mean relative error is significantly lower than that from either basic estimator.

The data plotted in Figure 5.4 is the mean relative error over all collections; each individual collection, however, shows similar trends. We can conclude that each of the three methods improves with more interactions, over a variety of collections, and that the multiple capture-recapture and capture history methods in particular can produce better estimates than other techniques if they are allowed enough samples.

5.3.3 Sample quality

A third set of experiments considered the effect of sample quality. Any samples used in a real-world system are prone to bias (Chapter 4), and the questions addressed here are: do biased samples produce poorer size estimates, and, if so, do errors increase with sample bias? Further, since biased samples cover fewer documents than random samples, are biased samples more likely to produce underestimates of collection size?

Two sets of samples were used: samples generated by the multiple queries sampler of Section 4.3, which was the best-performing sampler in previous experiments; and samples generated by query-based sampling (Section 4.2.2; Callan et al. [1999]), commonly used in other work.

To investigate the impact of biased but realistic samples, each method tested in Section 5.3.1 was re-run for each collection in the testbed. Samples for the capture-recapture, multiple capture-recapture, capture history, and random documents methods were drawn using the multiple queries sampler; models were built for the sample-resample method using documents sampled the same way. (The uncorrelated query pools method does not make use of sampled documents, so was not re-run.) Each method was again run ten times over each collection, and a t test was used to compare these sets of ten estimates.

The multiple queries sampler was unable to sample 1000 documents from the calendar collection, and so the capture-recapture (CR), sample-resample (SR), and random documents methods failed to estimate a size for this collection and were assigned a relative error of 100%. Estimates were significantly worse on the calendar collection for the capture-recapture, multiple capture-recapture (MCR), capture history (CH), and sample-resample methods, but in no other cases was there a significant difference between estimates produced with these biased samples and those produced earlier.

Collection	t test results
calendar	CH < MCR < (SR, CR, BER, BEU)
zsh-list	(CR, MCR, CH) < SR
procmail	CR < BEU < (SR, BER)
email	(CR, MCR, CH, BEU) < SR; (CR, CH) < BER
WSJ	(MCR, CH) < SR < BEU; CR < BEU
.GOV	(BER, BEU) < (CR, MCR, CH, SR)
Overall	(MCR, CH) < (SR, BER, BEU); CR < (SR, BER)

Table 5.3: Size estimates compared, using the multiple queries sampler. "$a < b$" indicates the mean error from method a is less than that from b according to a t test ($\alpha = 0.05$). Each method was run ten times on each collection. See text for comments on false alarms.

For most collections and all methods of size estimation, then, it appears that the multiple queries sampler produces estimates as good as those from true random samples.

Each method was again compared with all others, over each collection, and results are listed in Table 5.3. The uncorrelated query pools estimator performs better here, relative to the other methods, than in earlier experiments; this is expected since it does not rely on samples and therefore will not degrade. As noted, the capture-recapture, sample-resample, and random documents (BER) methods perform poorly on the calendar collection. Otherwise, results with these biased samples are similar to those using random samples in Table 5.2; the uncorrelated query pools estimator (BEU) remains useful for large collections. Again, we may expect around 10 false alarms; see Section 5.3.1.

A second experiment considered samples from the query-based sampler of Section 4.2.2. Although this is known to produce significantly biased samples (see Section 4.5.3, and Shokouhi et al. [2006]), it is commonly used in metasearch applications including for size estimation. Again, besides the input samples or models, parameters for each method were unchanged, and t tests compared relative error.

Estimates based on these more biased samples were substantially worse than those based on random samples. The capture-recapture (CR), multiple capture-recapture (MCR), and capture history (CH) methods were each significantly worse on five of the six collections, and the sample-resample method (SR) was significantly worse on three of the six.

Despite being significantly poorer at estimating the size of the .GOV collection, the random documents method (BER) proved more robust and was not significantly worse in any other cases. This is likely due to the choice of query pool; although the set of documents sampled is biased, the bias is not related to queries in the pool and so the sample remains a good basis for estimating $W_{\mathcal{P},\mathcal{D}}$ and $p_{\mathcal{P}}$.

Collection	t test results
calendar	CR < CH < MCR < BEU < SR;
	CR < CH < MCR < BER
zsh-list	(CR, CER, BEU) < (MCR, CH)
procmail	BEU < CR < SR < CH < MCR;
	BEU < BER < CH < MCR
email	BEU < (CR, SR, BER) < CH < MCR
WSJ	(CR, BER) < (MCR, CH) < BEU;
	(CR, BER) < SR
.GOV	(CH, BEU) < (CR, MCR, CH)
Overall	(CR, BER, BEU) < CH < MCR < SR

Table 5.4: Size estimates compared, using the QBS sampler. "$a < b$" indicates the mean error from method a is less than that from b according to a t test ($\alpha = 0.05$). Each method was run ten times on each collection. See text for comments on false alarms.

These results are also shown by the between-methods t tests shown in Table 5.4. The capture-recapture, multiple capture-recapture, and capture history methods are particularly affected by the bias in these samples; again, the uncorrelated query pools method (BEU) performs well with the .GOV collection. Once again, the comments on false alarms in Section 5.3.1 apply to this experiment.

With samples which are good enough, then, the error in size estimates is not much affected; and in this regard the multiple queries method produces samples which are good enough for all but the smallest collection. Samples which are more biased, such as those from the query-based sampler, introduce significant error.

The second question of interest was: are biased samples likely to produce underestimates of size? With fewer documents represented in biased samples, it seems likely that those methods which rely on random samples will estimate smaller sizes overall. This is in fact the case: the capture-recapture, multiple capture-recapture, and capture history methods report smaller average estimates in most cases with the biased samples from the multiple queries sampler, and in all cases with the very biased query-based samples.

Corrected variants

The corrections of Shokouhi et al. [2006] proved of no use given perfectly random samples, but it is reasonable to expect an improvement given more biased samples. However, this is not the case: given biased samples from the multiple queries sampler, "corrected" estimates were significantly worse for four of the six collections for multiple capture-recapture and three of the six for capture history. Given the more biased samples from the query-based sampler, the "corrected" estimates were significantly worse for all collections and both methods (with the sole exception of the WSJ collection for multiple capture-recapture).

The corrections do not appear to be useful in this testbed, although improvements were suggested for TREC data. This may suggest over-fitting in the original work.

5.3.4 Parameter settings

As before, there are a number of parameters for some algorithms. A final set of experiments varied these parameters to investigate possible improvements.

Capture-recapture

The capture-recapture technique had used two random samples of 1000 documents each. Followup experiments used samples of 500 and of 200 documents, again making ten estimates for each collection (so using twenty different samples for each). With samples of 500 documents, estimates for two of the six collections were significantly worse (one-sided t test, $\alpha = 0.05$). With samples of 200, estimates for all but one collection were significantly worse.

This indicates that the capture-recapture technique is, in general, improved by larger samples and made worse by smaller samples. Generating larger samples is costly, however, and may not always be practical.

Basic estimator methods

The random documents method failed to produce an estimate for the calendar collection, since the pool consisted of five-digit sequences and there were no such sequences in any calendar document (so $\mathcal{P}+ = \emptyset$). Further experiments considered other choices of pool.

With a pool for two-digit sequences, which are likely to be much more common, it was possible to estimate the size of the calendar collection; errors in this case were similar to those from the multiple capture-recapture or the capture history methods. ($\mathcal{P}+$ is not empty for this alternative pool, although $\text{VDENSITY}(\mathcal{P})$ is still low.) With a pool of common English words, as used for the mutiple queries sampler, results were similar.

The choice of pool allows a tradeoff. When queries in a pool match more documents, there is less chance of failing to return an estimate; however more effort is needed to estimate $W_{\mathcal{P},\mathcal{D}}$ since each matching document must be downloaded. Similarly, by using more of the pool $\mathcal{P}$ to estimate $W_{\mathcal{P},\mathcal{D}}$ a more accurate estimate can be obtained but at the expense of increased work. It is necessary to find the right tradeoff between increased coverage or sample size (and more accuracy) and lower coverage or sample size (and more speed), but it is not clear how to do this without some prior knowledge of the collection.

The uncorrelated query pools estimator also allows a choice of pool. First experiments, following Broder et al. [2006], used five-digit sequences for $\mathcal{P}$ and up to 100,000 medium-frequency terms from each collection for $\mathcal{Q}$. Without some prior knowledge of a collection, it would not possible to define $\mathcal{Q}$ this way; followup experiments therefore considered an alternative, where $\mathcal{Q}$ is the collection-independent list of common English words. With $\mathcal{P}$ either all five-digit sequences, as before, or all two-digit sequences, errors were similar and in the majority of cases were not significantly different (two-sided t test, $\alpha = 0.05$; only in one case, with $\mathcal{P}$ all five-digit sequences and using the zsh-list collection, was there a significant improvement). Provided each pool covers at least some documents, so $\text{VDENSITY}(\mathcal{P}) > 0$, the estimator appears to be largely insensitive to the exact pools used.

5.4 Conclusions

The experiments in this chapter show that no single method for size estimation works well in all circumstances. There are however broad patterns:

- Capture-recapture provides good estimates of smaller collections. These degrade as samples are more biased but remain better than other methods in many cases, as long as the sampler itself does not fail (as, for example, the multiple queries sampler on the calendar collection; see Section 5.3.3).
- If document text is available, the uncorrelated query pools method provides good estimates of larger collections and does not need a sample of documents.
- If good quality random samples are available, capture-recapture and capture history provide good estimates of smaller collections; sample-resample and the basic estimator methods provide good estimates of larger collections. Multiple capture-recapture and capture history rapidly improve, however, if they are allowed more interactions and can eventually outperform other estimators.
- The two basic estimator methods provide good estimates of larger collections if good-quality samples are not possible but document text is available. This is likely to be the case in many applications.

The “corrected” variants of multiple capture-recapture and capture history do not appear to offer any benefit even with biased samples.

Size estimates produced with multiple queries samples were not significantly different to those produced with random samples, with the exception of some estimates on the calendar collection. This suggests that the multiple queries sampler produces samples which, despite being biased, are of high enough quality for metasearch applications. This is not true of the commonly-used query-based sampler.

Size estimates are used at several points in the metasearch process. In language models, considered in the following chapter, they are used for normalisation. Estimates are also used to inform server selection, which is discussed in Chapter 7.

Chapter 6

Server characterisation: subject matter and language

Server selection, discussed in Section 2.2 and in Chapter 7 following, makes use of characterisations of the available servers. As well as size, overlap, and other characteristics already considered, an important part of this characterisation is the subject matter of each collection: the types of document which are available and the types of queries for which it would be a good choice.

Past work has proposed several techniques for summarising subject matter. Section 6.1 describes several techniques, as well as variants and related work.

Unigram language models, outlined in Section 6.1.2, can be built with no cooperation from servers and are commonly used in server selection algorithms. These models are examined in experiments in Sections 6.2 and 6.3, where the quality of a model is shown to depend on the quality of the sampling mechanism used. While previous work has generally used TREC or Web data, these experiments again concentrate on collections representative of those likely to be used in personal metasearch. Across a range of collections, models built with the multiple queries sampler are largely indistinguishable from those built with true random samples; those built with query-based sampling are of lower quality. Examination of the correlation and relative power of three measures suggests that one measure, Kullback-Leibler divergence, captures model quality more usefully than other proposed alternatives.

6.1 Related work

A large body of work has addressed the problem of characterising the subjects covered by collections. There have been two major approaches: classifications in a predetermined taxonomy, such as a cataloguing scheme (Section 6.1.1), and models of the terms used in documents (Section 6.1.2). In each case, some proposed techniques require server cooperation; some require the text of documents; and alternatives assume that only a query interface is available.

6.1.1 Classifications

A number of systems have been developed to classify collections according to their subject matter using a pre-determined taxonomy. If queries can also be classified accurately, such schemes simplify server selection: the only operation is to select for each query those servers which cover a relevant part of the topic hierarchy.

Sheldon et al. [1994] described a hierarchical metasearch system which used "content labels" to describe cooperating servers. These labels, which were manually assigned, included information on servers (location, administrator, name, etc.) and collections (searchable fields and possible query completions). Labels were exposed to the user through a search tool and could be used for manual server selection and for search.

The more involved Pharos system [Dolin et al. 1996] used several pre-defined taxonomies to represent the contents of each collection and to suggest appropriate servers for a given query. Collections could be on- or off-line. Metadata provided by each server represented that server's holdings according to four parallel hierarchies: subject matter, according to Library of Congress classifications (LCC);[1] geographical coverage of the collection; historical coverage of the collection; and the publication dates of included documents.

This metadata was collated by each server from per-document information. The classification of each document could be manual, for example for library holdings already classified with the LCC, or automatic; Dolin et al. reported using latent semantic indexing [Deerwester et al. 1990] for automatic classification. Once collated, the metadata was distributed using a scheme similar to Usenet news [Kantor and Lapsley 1986]. "Mid-level" servers collected and replicated metadata regarding particular classifications or time periods, and an overview of the whole distributed system was collected at each of a number of "high-level" servers.

The hierarchical classification of servers was used by Pharos to suggest a set of servers for a user's query. Using latent semantic indexing or some equivalent method, the query could be mapped to a node, or set of nodes, in a relevant hierarchy; or the user could navigate the hierarchy and choose a node directly. With the vocabulary controlled in this way, queries to high-level servers would return a coarse-grained view of the available resources, and a user could indicate relevant mid-level servers and eventually individual collections.

Simulation experiments with random collections and queries [Dolin et al. 1997] and with newsgroup data [Dolin et al. 1999] suggested that the classification scheme enabled fairly precise selection of collections, although the authors noted that it was not clear how it would scale to larger holdings. Although Pharos was expected to scale to millions of servers [Dolin et al. 1997], it appears that it has not been used in full-scale experiments.

Pharos relied on cooperating servers and a shared classification scheme. Alternatives have assumed uncooperative servers, and used a query interface to classify collection contents.

[1] `http://www.loc.gov/cataloging/classification/`

Profusion [Gauch et al. 1996], described in Section 2.3.2, used a set of 13 categories chosen to be representative of users' interests, including "science and engineering", "computer science", and "travel". About four queries per category, 48 in total, were issued to each of the six servers in use; each returned document was manually judged for relevance. The servers' performance across each category was later used to influence server selection and result merging.

The Profusion strategy, although appropriate for a small number of servers and classifications, could not scale well to large numbers of either. The more scalable "probe and count" algorithm [Ipeirotis et al. 2001] uses the size of result sets, not document text, to classify collections in a heirarchical taxonomy. In a first step, a rule-based classifier such as RIPPER [Cohen 1995; Cohen 1996] is trained with a set of documents, each in a known category. The classifier outputs rules of the form "if the document contains $term_i$ and $term_j$ and …, then it is in $category_k$".

Each learned rule is then transformed into a Boolean "probe query" of the form "$term_i$ AND $term_j$ AND …" and forwarded to the server; the size of the result set is used to approximate the number of documents about $category_k$ in the collection. It is not clear how the probe and count algorithm deals with query overflow, inaccurate size estimates from servers, or servers which do not have a Boolean query language.

Document counts for each category are adjusted according to the classifier's accuracy on the training data, and the adjusted counts are finally used to find the node in the hierarchy which best represents the collection. The algorithm can be adapted to prefer specificity — classifications which cover only those topics in the collection — or coverage — classifications which cover as much of the collection as possible. Experiments with two sets of data, from newsgroups and Web-based databases, demonstrated the probe and count algorithm required fewer interactions with the server than alternatives and that its performance, on analogues of precision and recall, was reasonable.

The probe and count algorithm, renamed "QProber", was extended by Gravano and Ipeirotis [2003]. While probe and count relied on classifiers themselves producing rules of the form "if $term_i$ and $term_j$ and …", QProber included an algorithm for generating such rules from the output of other classifiers including support vector machines [Joachims 1998] and naïve Bayesian classifiers [Duda and Hart 1975]. A further set of experiments confirmed that these alternative classifiers produced server characterisations of similar accuracy.

Meng et al. [2002] described a similar system, also using probe queries to assign collections to a heirarchical taxonomy. Probe queries were created from the names of each category and all of its sub-categories, and the similarity between each query and documents in the collection was used to inform the assignment.

Meng et al. introduced three algorithms for calculating this similarity. The first, "high similarity with database centroid", uses information on term occurrences in each document to calculate an overall centroid for the collection. This centroid is compared, using a cosine similarity measure, with the description of each classification and the most similar classifications are retained. Although this technique requires access to accurate term frequency data for each document in the collection, and is therefore not useful for uncooperative servers, Meng et al. suggest that a sample of documents could be used instead. This was however not considered further.

The second algorithm, "high average similarity over retrieved documents", is a variant on the first. Rather than calculate the similarity between the entire collection and each description, this variant translates each description to a query, takes the text of the top few documents, and uses the similarity between these documents and the description as the ranking criteria. This algorithm requires access to the text of returned documents, and assumes servers are effective, but can otherwise work with an uncooperative server.

The third and final algorithm of Meng et al. [2002] uses singular value decomposition (SVD). Again, term information is assembled for each of a number of training collections. This, plus a manual classification of each collection, is given as input to an SVD classifier, which finds correlations between terms and classifications [Yang and Chute 1994]. These correlations, with information on term frequencies, can be used to classify a new collection.

Experiments using classifications from Yahoo![2] and the Open Directory Project[3] compared automatic classifications of 24 collections — 18 newsgroups and six Web collections from two universities — to a manual classification. An analogue of 11-point average precision indicated that the retrieved documents variant outperformed the other two algorithms; in a case study [Meng et al. 2002], a prototype which used this algorithm correctly classified a Web-based database.

6.1.2 Language models

A second line of research has described collections by building *language models*. These models consider documents as the result of a stochastic process of language generation, and describe a probability distribution which captures this. In principle, these distributions can be arbitarily complex but models have most commonly been simple: a set of individual terms with associated frequency information [Ponte and Croft 1998]. Tables 6.1 and 6.2 have examples with, respectively, term frequency and document frequency. Such language models have also been referred to as "representatives" of collections [Liu et al. 2001; Shokouhi et al. 2007] and as "content summaries" [Ipeirotis et al. 2001; Ru and Horowitz 2005].

Language models were introduced to information retrieval by Ponte and Croft [1998], who described a use of models to rank documents in response to a query. A model is first inferred from the text of each document. Assuming each term occurence is independent of all others, the probability of generating the query q from the process underlying document d is the product of the probabilities of generating each term in the query and the probabilities of *not* generating every other term:

$$\Pr(q|d) = \prod_{t \in q} \Pr(t|d) \prod_{t \in \mathcal{T}_d \setminus q} \left(1 - \Pr(t|d)\right), \tag{6.1}$$

[2] http://dir.yahoo.com/

[3] http://dmoz.org/

where $\mathcal{T}_d$ is the set of terms in the document.

These per-document language models are smoothed in two ways. A naïve calculation of $\Pr(t|d)$ is to use the relative frequency of t in d and assign $\Pr(t|d) = \text{tf}_d(t)/|d|$, where $\text{tf}_d(t)$ is the term frequency of t in d — the number of times t occurs in d — and $|d|$ is the number of terms in the document. If a term t does not occur at all in d, this leads to the conclusion that $\Pr(t|d) = 0$ and that t is never a possible output from the underlying stochastic process. As Ponte and Croft observe, this is a strange conclusion: not seeing something should not mean it is impossible. It is also not useful, since $\Pr(q|d)$ will be zero in all such cases. Instead, where a term does not occur we smooth the model by approximating

$$\Pr{}_{\text{zero}}(t) = \frac{\sum_{d\in\mathcal{D}} \text{tf}_d(t)}{\sum_{d\in\mathcal{D}} |d|},$$

so if the term does not appear in a document at all then the mean rate of occurrence over the whole collection is used instead.

Models are also smoothed by including, for each term t, a component based on the mean probability of t in those documents where it appears: this is

$$\Pr{}_{\text{avg}}(t) = \frac{\sum_{d\in\mathcal{D}}(\text{tf}_d(t)/|d|)}{\text{df}(t)}.$$

Given a mixing parameter $\hat{R}$, this and the previous correction finally give

$$\Pr(t|d) = \begin{cases} (\text{tf}_d(t)/|d|)^{1-\hat{R}} \times \Pr{}_{\text{avg}}(t)^{\hat{R}} & \text{if } \text{tf}_d(t) > 0; \\ \Pr{}_{\text{zero}}(t) & \text{otherwise.} \end{cases}$$

This definition is substituted into Equation 6.1, and documents are ranked according to $\Pr(q|d)$. Experiments by Ponte and Croft [1998], using TREC ad hoc data, showed significant improvements in recall and precision using this model-based ranking instead of a standard ranking from INQUERY [Callan et al. 1992].

Similar smoothing can be seen in the Kullback-Leibler method for server selection, described in Section 7.1.6. Ipeirotis and Gravano [2004] have suggested a further smoothing technique, which uses mixtures of models according to a collection's place in a given topic heirarchy; this topic-based method increased the performance of the CORI server selection algorithm (Section 7.1.2) in their experiments.

Ponte and Croft described unigram language models for individual documents; more commonly, models are used to summarise entire collections. Language models for collections are used in a number of server selection techniques including CORI [Callan et al. 1995], CVV [Yuwono and Lee 1997], GlOSS [Gravano et al. 1999], and Kullback-Leibler divergence [Si et al. 2002; Xu and Croft 1999], and are considered in the experiments below.

Gravano et al. [1997a] describe STARTS, a protocol for communication between a metasearch application and cooperating servers which includes this sort of collection-scale model. Each STARTS-compatible server makes available metadata on each collection it indexes; this metadata includes a term list with associated frequency information. This could be used by a metasearch application to inform server selection and result merging. Harvest [Bowman et al. 1994] similarly distributed metadata, in this case at the document level, between cooperating components. SearchDB-ML [Powell and Fox 1998], a simpler alternative, did not include term data but did include manually-assigned descriptions of each server which could be used to inform server selection.

In metasearch applications with uncooperative servers, language models for collections can be built from a sample of documents. Query-based sampling, for example, was originally proposed as a means to build language models [Callan et al. 1999], and recent work has measured the quality of this technique by the quality of the models produced (see for example Baillie et al. [2006b]; Shokouhi et al. [2007]; and Section 6.2 below). Since these models are built from a sample — a subset of a collection — they are constrained to reporting relative term frequencies, such as the mean number of term occurrences per document sampled. Absolute counts can be estimated with the addition of a collection size estimate (Chapter 5); Ipeirotis and Gravano [2002] also suggest an algorithm which estimates total term counts while sampling as long as the overall shape of the term distribution is known. This alternative could for example be used to estimate df in the CORI algorithm (Section 7.1.2), although it does not appear to have been considered.

6.2 Language modelling experiments

Unigram language models as described above are used in a number of server selection techniques including CORI [Callan et al. 1995], CVV [Yuwono and Lee 1997], GlOSS [Gravano et al. 1999], and Kullback-Leibler divergence [Si et al. 2002; Xu and Croft 1999], and are therefore considered here. Experiments address three questions: how good are language models built with sampled documents? How can this quality be measured? And can models be improved with more effort?

These experiments do not consider the classification strategies of Section 6.1.1. Most successful server selection techniques to date have used unigram language models, and it is not presently clear what form of topic heirarchy or query classification technique would be appropriate for personal metasearch.

As in Chapters 4 and 5, the experiments reported here use six collections which represent the range of size and subject matter in personal metasearch. They are summarised in Table 4.2 on page 40.

6.2.1 Samples

Models were built from documents returned by each of three different samplers. As a baseline, models were built using documents selected entirely at random; as noted in Chapter 4, this is not possible in practice but provides a best case for comparison. Models were also built from documents sampled by the multiple queries sampler of Section 4.3, which was the best performing sampler in earlier experiments, and from documents sampled by the query-based sampler of Section 4.2.2 [Callan et al. 1999], which has been commonly used in previous work [Baillie et al. 2006b; Callan and Connell 2001; Hawking and Thomas 2005; Shokouhi et al. 2007; Si and Callan 2003b].

6.2.2 Model quality

Each experiment considers the quality of language models built from each set of samples. Given access to the full collection, and hence accurate term frequency information, three different measures have been proposed to measure model quality: the collection term frequency ratio, differences in term ranking, and Kullback-Leibler divergence.

(Baillie et al. [2006a] have suggested a fourth measure of model quality, the likelihood of a model generating a pool of queries $\mathcal{P}$. By letting $\mathcal{P}$ be different for each user, model quality can be calculated relative to a that user's information needs; pools can also be collection independent. Experiments by Baillie et al. showed a moderate to strong correlation between likelihood and Kullback-Leibler divergence, suggesting this could measure quality without any knowledge of true term distributions; however a pool must be defined before this measure can be calculated, and since there is no pool of likely queries the likelihood measure is not considered in the experiments below.)

Collection term frequency ratio

The collection term frequency (ctf) ratio was introduced by Callan et al. [1999] to measure the quality of models built by query-based sampling, where these models are simple sets of terms with associated frequency information. It measures the correspondence between the vocabulary of a model m and a collection, weighted to give more emphasis to more common terms.

The *term frequency* of a term t, $\mathrm{tf}(t)$, is the total number of times it occurs in the collection. $\mathcal{T}$ denotes the set of terms in a collection; ctf ratio then measures the proportion of total term occurrences accounted for in the model:

$$\text{ctf ratio} = \frac{\sum_{t \in m} \mathrm{tf}(t)}{\sum_{u \in \mathcal{T}} \mathrm{tf}(u)}.$$

The ratio falls in the range $[0, 1]$, and higher values are assumed to represent a better quality model.

For example, consider Table 6.1, which extracts from the email collection used in the following experiments. In this subset there are a total of 51,540 term occurrences. A model which included only the terms "au" and "years" would have a collection term frequency ratio of $(23187 + 2728)/51540$, or 0.5028, regardless of the frequency of these terms in the model. If it also included "anu" and "target", it would produce a ctf ratio of $(23187 + 20219 + 2728 + 2706)/51540 = 0.9476$, despite including only four of eight terms.

	Collection		Model	
Term t	$\mathrm{tf}(t)$	$\Pr(t\|c)$	$\mathrm{tf}_m(t)$	$\Pr(t\|m)$
au	23187	0.4499	235	0.9589
anu	20219	0.3923	—	≈ 0
		…		
years	2728	0.0529	10	0.0408
target	2706	0.0525	—	≈ 0
face	2697	0.0523	—	≈ 0
		…		
zywczak	1	≈ 0	—	≈ 0
zz9s	1	≈ 0	—	≈ 0
zzz	1	≈ 0	—	≈ 0
Total	51540		245	

Table 6.1: A subset of term frequencies, extracted from the email collection. See the text for calculations.

Term rankings

Collection term frequency ratio considers the proportion of terms included in the model, but does not distinguish between a model with accurate term frequency information and a model without. For example, a single document such as a dictionary may contribute greatly to ctf ratio but without information on relative frequencies such a document would not be particularly helpful in selecting servers [Baillie et al. 2006b]. A second measure introduced by Callan and Connell [2001] addresses this by comparing the ranking of terms in the model and the collection.

The Spearman rank correlation coefficient r_s is used here to measure the correlation between two lists of ranks: it is equivalent to the Pearson product-moment correlation coefficient over ranks [Press et al. 1992; Sheskin 2004]. This is applied by Callan and Connell to measure the quality of a model by first ordering terms in the collection according to document frequency, $\mathrm{df}(t)$; doing the same for terms in the model with $\mathrm{df}_m(d)$; and calculating the correlation

$$r_s = \frac{1 - \frac{6}{n^3-n}\left(\sum_i d_i^2 + \frac{1}{12}\sum_j (f_j^3 - f_j) + \frac{1}{12}\sum_k (g_k^3 - g_k)\right)}{\sqrt{\left(1 - \frac{\sum_j (f_j^3 - f_j)}{n^3-n}\right)}\sqrt{\left(1 - \frac{\sum_k (g_k^3 - g_k)}{n^3-n}\right)}}.$$

Here n is the number of terms which are in both the model and the collection (so $n = |\mathcal{T} \cap m|$) and d_i the difference, for the ith term, between its rank in the collection and its rank in the model. f_j is the number of terms tied for the jth rank in the model, and g_k is the number of ties in the kth rank for the collection; an example is in Table 6.2. Note that if two or more terms have the same frequency, such as "website" and "march" in Table 6.2, each has a rank adjusted for the number of tied terms.

	Collection			Model			
Term t	df(t)	Rank	g_k	df$_m(t)$	Rank	f_j	d_i^2
paul	8421	1	1	724	1	1	0
http	7197	2	1	564	2.5	2	0.25
time	6697	3	1	564	2.5	2	0.25
				…			
data	914	4	1	11	6	3	4
website	910	5.5	2	11	6	3	0.25
march	910	5.5	2	11	6	3	0.25
open	907	7	1	12	4	1	9
				…			
zywczak	1	9	3	1	9	3	0
zz9s	1	9	3	1	9	3	0
zzz	1	9	3	1	9	3	0
Total							14

Table 6.2: A subset of document frequencies, extracted from the email collection. See the text for calculations.

Spearman's r_s takes values in $[-1, 1]$, where 1 indicates a perfect positive correlation (so the terms are in exactly the same order), -1 indicates a perfect negative correlation (so the terms are in exactly the opposite order), and 0 indicates no correlation. Since the terms in the model may only be a subset of the terms in the collection, r_s is only calculated over the terms in the model. Therefore, if terms — even common ones — are missing from the model, r_s will not reflect this.

Results reported here use a simpler calculation [Sheskin 2004]:

$$\begin{aligned} r_s &= \frac{T_f + T_g - \sum_i d_i^2}{2\sqrt{T_f T_g}}, \\ \text{where } T_f &= \frac{1}{12}\left(n^3 - n - \sum_j (f_j^3 - f_j)\right) \\ \text{and } T_g &= \frac{1}{12}\left(n^3 - n - \sum_k (g_k^3 - g_k)\right). \end{aligned}$$

Table 6.2 includes partial data from the email collection and a sample model. Considering only the ten terms shown, we can calculate

$$\begin{aligned} T_f &= \tfrac{1}{12}\left(n^3 - n - \textstyle\sum_j (f_j^3 - f_j)\right) \\ &= \tfrac{1}{12}\left(10^3 - 10 - [(2^3 - 2) + (3^3 - 3) + (3^3 - 3)]\right) \\ &= 78. \end{aligned}$$

$$\begin{aligned}
\text{Similarly, } T_g &= \tfrac{1}{12}\left(n^3 - n - \textstyle\sum_k (g_k^3 - g_k)\right) \\
&= \tfrac{1}{12}\left(10^3 - 10 - [(2^3 - 2) + (3^3 - 3)]\right) \\
&= 80, \\
\text{and } r_s &= \frac{T_f + T_g - \sum_i d_i^2}{2\sqrt{T_f T_g}} \\
&= \frac{78 + 80 - 14}{2\sqrt{78 \times 80}} \\
&= 0.9115.
\end{aligned}$$

Baillie et al. [2006b] suggest that r_s may be a superior measure to ctf ratio, since if term frequencies follow a known distribution, such as hyperbolic rank-frequency [Zipf 1949, Chapter 2] or Waring [Wolfram 1992], then it may be possible to reconstruct frequency information from a ranking.

Kullback-Leibler divergence

A model's ctf ratio considers the proportion of terms included, and r_s the relative ranking of each. An alternative measure considers both, as well as the frequency of each term.

Kullback-Leibler divergence [Kullback and Leibler 1951] was introduced to this context by Xu and Croft [1999] as a means to compare a query to a collection; it is used here to compare a model to a collection, and was independently suggested for the same purpose by Baillie et al. [2006b]. The collection and model are each considered as discrete probability distributions over $\mathcal{T}$, the terms in the collection, and the divergence is then the relative entropy between the two. (Other information theoretic measures, such as Jensen-Shannon divergence or Jeffreys divergence, also describe the difference between two distributions but have been little used in information retrieval.) The divergence is defined by[4]

$$\mathrm{D}_{KL}(c\|m) = \sum_{t\in\mathcal{T}} \Pr(t|c)\log_2 \frac{\Pr(t|c)}{\Pr(t|m)}, \tag{6.2}$$

where $\Pr(t|c)$ is the probability distribution over the collection c

$$\Pr(t|c) = \frac{\mathrm{tf}(t)}{\sum_{u\in\mathcal{T}} \mathrm{tf}(u)}$$

and $\Pr(t|m)$, the probability distribution over the model m, uses Laplace smoothing in case terms are missing or the sample is small:

$$\Pr(t|m) = \frac{\mathrm{tf}_m(t) + \alpha}{\sum_{u\in\mathcal{T}} \mathrm{tf}_m(u) + |\mathcal{T}|\alpha}.$$

[4] The choice of base for the logarithm is not critical. The data presented in this thesis uses base two, and measures divergence in bits.

The experiments here follow Xu and Croft [1999] in assigning $\alpha = 0.01$. (An earlier use by Ipeirotis and Gravano [2004] did not smooth $\Pr(t|m)$, and was therefore constrained to considering only terms in both the model and the collection.)

Kullback-Leibler divergence falls in $[0, \infty)$. A value of zero indicates complete concordance; larger values indicate more divergence between the distributions. This is not a true distance metric since it is not symmetric (it is possible that $\mathrm{D}_{KL}(a\|b) \neq \mathrm{D}_{KL}(b\|a)$) and does not obey the triangle inequality (it is possible that $\mathrm{D}_{KL}(a\|c) > \mathrm{D}_{KL}(a\|b) + \mathrm{D}_{KL}(b\|c)$) [Kullback 1959, Chapter 1].[5]

For example, consider again the model in Table 6.1. Using $\alpha = 0.01$ and with $|\mathcal{T}| = 8$, the contribution for the term "years" is

$$\Pr(\text{“years”}|c) \log_2 \frac{\Pr(\text{“years”}|c)}{\Pr(\text{“years”}|m)}.$$

In this case $\Pr(\text{“years”}|c) = 2728/51{,}540 = 0.0529$ and $\Pr(\text{“years”}|m) = (10 + 0.01)/(245 + 8 \times 0.01) = 0.0408$, so the contribution is $0.0529 \log_2(0.0529/0.0408) = 0.0198$. Repeating this for every other term in the collection and summing the contributions gives an overall divergence.

Unlike ctf ratio and r_s, Kullback-Liebler divergence directly measures the difference between two discrete distributions, and considers both vocabulary and absolute frequency. This suggests it will be a more useful measure of model quality, and the results of these experiments confirm this.

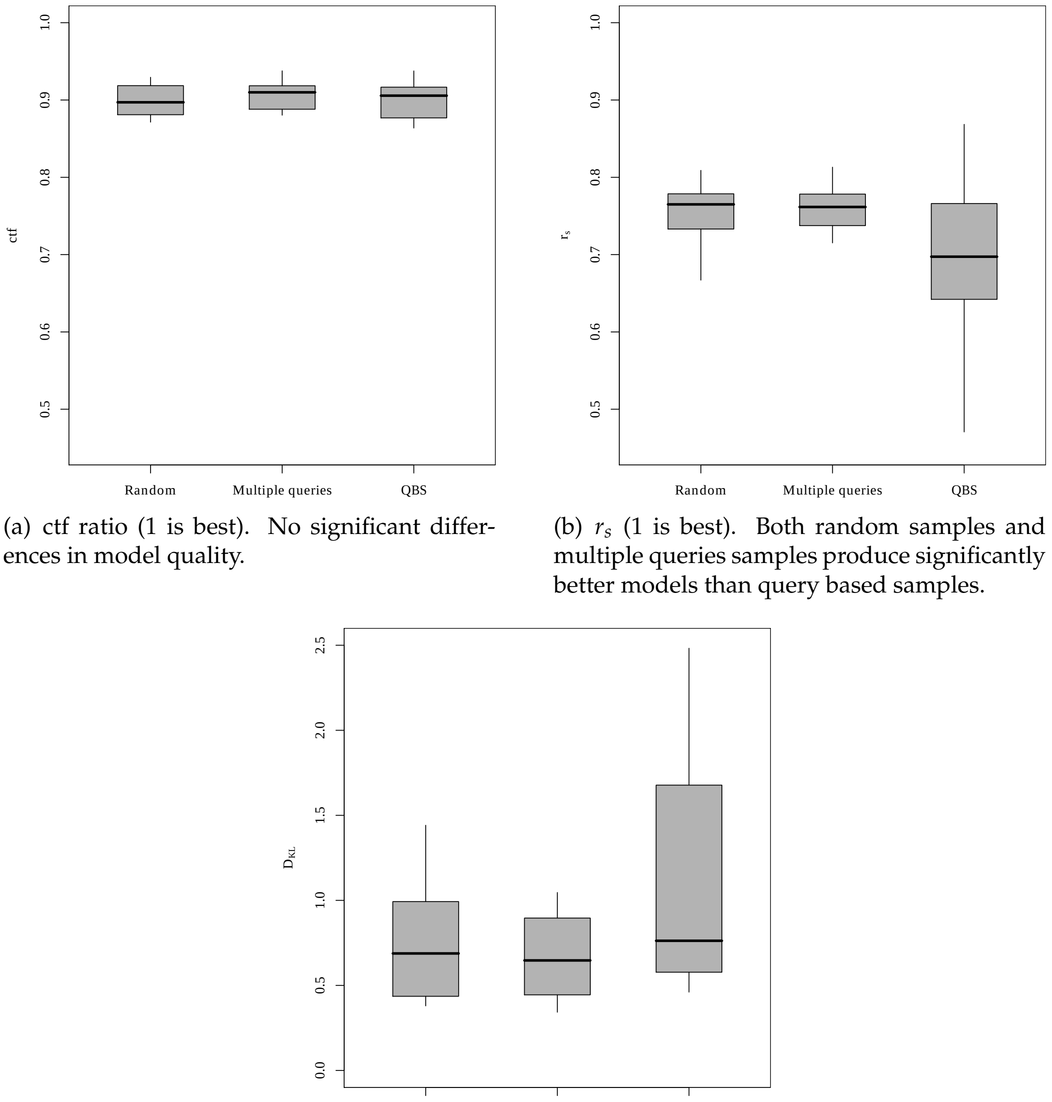
6.3 Results

Figure 6.1 summarises the ctf ratio, r_s, and D_{KL} measures over ten models for each of the six collections. In all but one case, a model was built using 300 documents selected randomly, by the multiple queries sampler, or by query-based sampling, and no stemming or stopping was carried out. The multiple queries sampler was unable to sample a full 300 documents from the calendar collection using the default query pool, so models used 200 documents in this case.

On all three measures, models built by random sampling and multiple queries samples are remarkably similar; there are no significant differences (t test, $\alpha = 0.05$). Models built by the query-based sampler differ significantly from both the others on the r_s measure ($p < 0.0007$ in both cases) and on the D_{KL} measure ($p < 0.0006$ against random sampling, $p < 0.02$ against multiple queries sampling).

Like the results of experiments in Chapter 5, these results suggest both that a better quality sample makes an appreciable difference to metasearch algorithms and that collection characterisations based on the multiple queries sampler are essentially indistinguishable from those based on true random samples.

[5] As originally described by Kullback and Leibler [1951], Kullback-Leibler divergence is symmetric: $\mathrm{D}_{KL}(c\|m) = \sum_{t \in \mathcal{T}} \left(\Pr(t|c) \log_2 \frac{\Pr(t|c)}{\Pr(t|m)} + \Pr(t|m) \log_2 \frac{\Pr(t|m)}{\Pr(t|c)} \right)$. The form given above, a "directed divergence" measure [Kullback 1959], is typical of current use. Jensen-Shannon and Jeffreys divergence are symmetric.



(a) ctf ratio (1 is best). No significant differences in model quality.

(b) r_s (1 is best). Both random samples and multiple queries samples produce significantly better models than query based samples.

(c) D_{KL} (0 is best). Both random samples and multiple queries samples produce significantly better models than query based samples.

Figure 6.1: Quality of models built from different samples. Ten models were built from 300 documents for each of six collections. See notes to Figure 5.2 on p. 63 for plotting conventions.

These results also suggest that ctf ratio may not have much discriminative power, since it scores models built from a biased sample as well as it does those from a truly random sample. This is not surprising: a large fraction of the ctf ratio will be gained from seeing the most common words, even once, and these words are likely to be represented in even a biased sample of documents. This in turn suggests that ctf ratio is not likely to be a useful measure of model quality.

6.3.1 Evolution of models

The models used in the first experiment were based on samples of 300 documents, following Callan et al. [1999]. It is reasonable to hypothesise that models built from more documents will be of higher quality; however, it is also likely that the quality of models will be constrained by any bias in the document samples themselves.

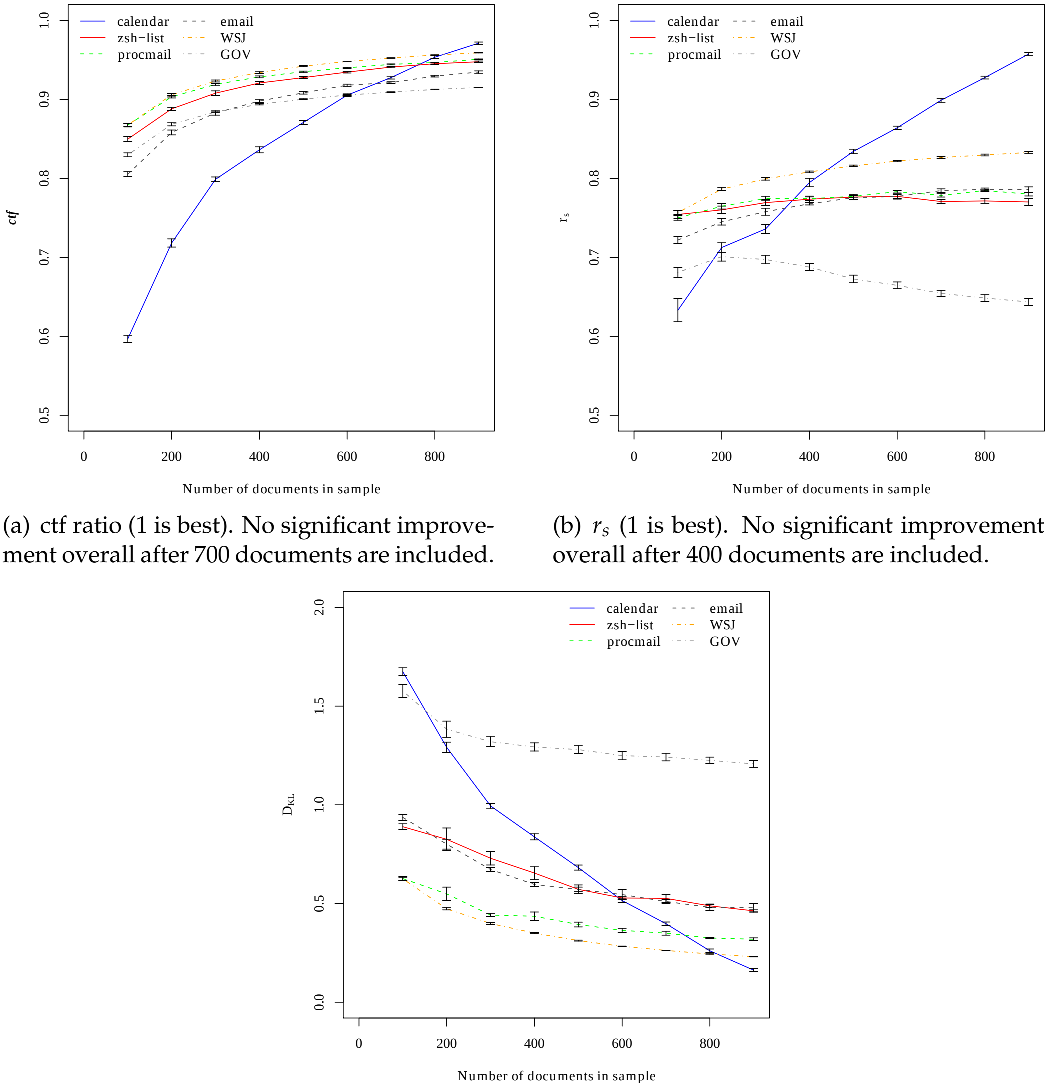
Figures 6.2 to 6.4 show improvements in ctf ratio, r_s, and D_{KL} for models built with increasing numbers of documents sampled randomly (Figure 6.2), with the multiple queries sampler (6.3), and with the query-based sampler (6.4). In each case, the lines plotted are the mean measure of each of ten models; the bars give one standard error to either side. As noted earlier, the calendar collection is particularly difficult to sample and neither the multiple queries nor the query-based method were able to produce large samples. The data for this collection is correspondingly cut off at 200 and 400 documents, respectively.

There are clear trends in each case. Models tend to improve on all metrics, and with all samplers, as more documents are included. This improvement does however slow down, and when models are built from random documents, there is no significant improvement in ctf ratio across all collections with more than 700 documents (t test, $\alpha = 0.05$). Spearman's r_s does not improve significantly after 400 documents are used, and D_{KL} does not improve significantly after 500 documents. Using documents from the multiple queries sampler, there is no significant improvement in ctf ratio after 700 documents, in r_s after 200, or in D_{KL} after 400; for the query-based sampler there is no significant improvement in ctf ratio after 700, in r_s after 100, or in D_{KL} after 400 documents have been included in the model.

Models of the calendar collection improve fastest in all cases, although from a relatively poor base. This is likely due to the nature of the documents in this collection: they are very short, with a mean of only four terms, and there is little overlap between terms in any two documents. With such a collection, there are very few common terms, so models built from a small number of documents will be of low quality; however, since there is little overlap between documents, each additional document will contribute to the terms seen and hence to the quality of the model.

Models of the .GOV collection are also of relatively low quality, although the ctf ratio is similar to other collections. This can be explained by the size of the collection: with many more documents than other collections, a fixed size sample will cover less of the collection.

The effect of biased samples, seen earlier, is confirmed in these experiments as models built with documents from the query-based sample continue to have lower

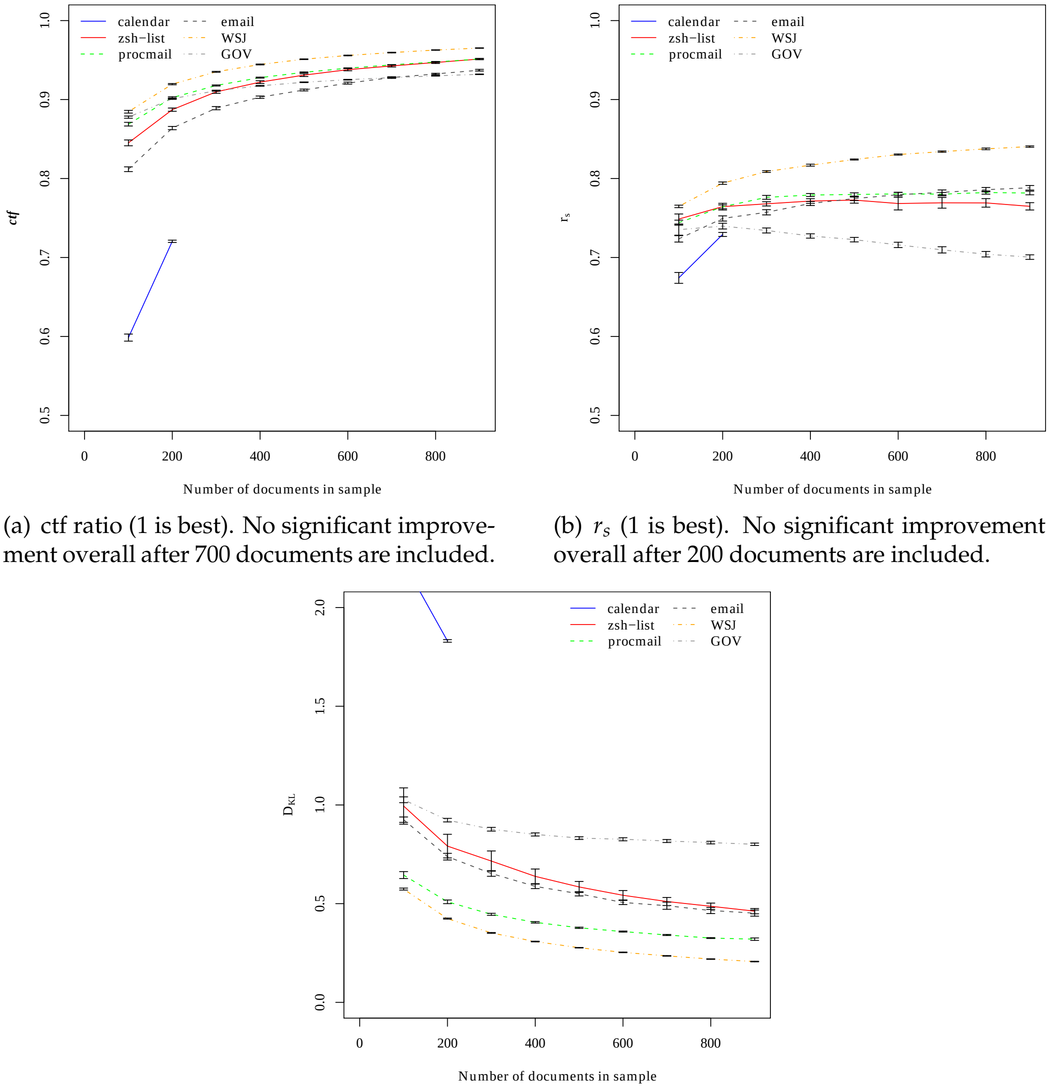


(a) ctf ratio (1 is best). No significant improvement overall after 700 documents are included.

(b) r_s (1 is best). No significant improvement overall after 400 documents are included.

(c) D_{KL} (0 is best). No significant improvement overall after 500 documents are included.

Figure 6.2: Improvement in quality measures with more documents from random samples. Plotted figures are the means from ten models each built from the specified number of documents, selected at random. Bars are ±1 standard error.

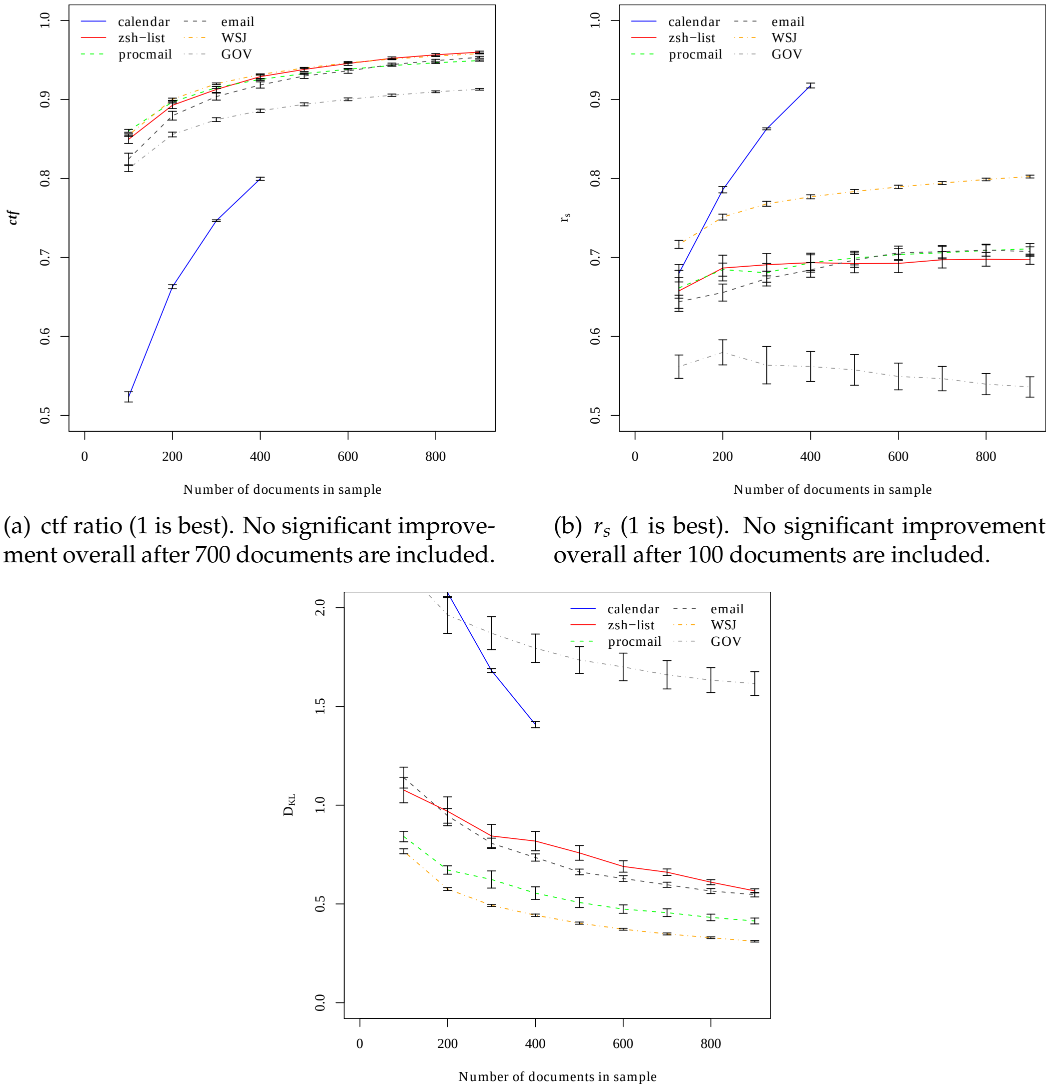


(a) ctf ratio (1 is best). No significant improvement overall after 700 documents are included.

(b) r_s (1 is best). No significant improvement overall after 200 documents are included.

(c) D_{KL} (0 is best). No significant improvement overall after 400 documents are included. Divergence is very high (2.2 ± 0.03) for the calendar collection at 100 documents sampled.

Figure 6.3: Improvement in quality measures with more documents from multiple queries samples. Plotted figures are the means from ten models each built from the specified number of documents, selected by the multiple queries sampler. Bars are ±1 standard error.



(a) ctf ratio (1 is best). No significant improvement overall after 700 documents are included.

(b) r_s (1 is best). No significant improvement overall after 100 documents are included.

(c) D_{KL} (0 is best). No significant improvement overall after 400 documents are included. Divergence is very high for the calendar collection at 100 (2.7 ± 0.06) and 200 (2.1 ± 0.02) documents sampled.

Figure 6.4: Improvement in quality measures with more documents from query-based samples. Plotted figures are the means from ten models each built from the specified number of documents, selected by the query-based sampler. Bars are ±1 standard error.

ctf ratio, lower r_s, and higher D_{KL}. With 900 documents per model, there is again no significant difference in overall ctf ratio between the three samplers, but a significant difference remains between models built with the query-based sampler and models built with either of the other samplers if the r_s or D_{KL} measures are considered ($p < 0.02$ in each case).

Overall, these observations support the two hypotheses on p. 85. First, models do improve on all metrics as the number of documents used increases, although improvements slow down and are not significant past around 700 documents. Second, the quality of the samples used is a constraint on the quality of the models which are built: with the more biased samples of documents from query-based sampling, all three measures stop improving earlier and are significantly worse than alternatives.

6.3.2 Correlation between measures

From the results in Figures 6.2 to 6.4, it seems that the three measures are in agreement: as the ctf ratio improves, so does r_s and Kullback-Leibler divergence. To investigate further, Pearson's product-moment correlation [Sheskin 2004] was computed between all three measures across all 1500 models built for the experiments above.

Over all models, ctf ratio and D_{KL} are moderately negatively correlated (Pearson's $r = -0.75$). Since a lower D_{KL} score represents a better model and a higher ctf ratio represents the same, this indicates that the two measures broadly agree on the quality of each model. r_s is less correlated with the other two measures: $r = 0.23$ with ctf ratio and $r = -0.65$ with D_{KL}.

In similar experiments Baillie et al. [2006b], using query-based sampling of TREC data, also observed a negative correlation between ctf ratio and D_{KL}. Unlike the experiments reported above, however, their results indicated a negative correlation also existed beteween ctf ratio and r_s, and a positive correlation (meaning in this case disagreement) between r_s and D_{KL}. It is not clear why this should be the case: seeing terms with a higher document frequency, and hence improving ctf ratio, need not result in terms being ranked poorly. Baillie et al. suggest that with more data, they may have seen a positive correlation between ctf ratio and r_s, and presumably a negative correlation between r_s and D_{KL}.

6.4 Conclusions

Alongside attributes such as collection size, an important aspect of characterisation is the subject matter and language used at each server. This can inform server selection by suggesting what types of documents a server makes available, and the types of queries for which it would be a good choice. Past work has been conducted in two areas: classification according to pre-defined taxonomies, such as library cataloguing schemes or the Open Directory heirarchy, and language models. In particular, unigram language models which record term frequency information are used by many server selection algorithms.

Experiments with unigram models show that the quality of the document sample used makes an appreciable difference to the quality of the eventual model. Models built with the multiple queries sampler of Section 4.3 are largely indistinguishable from those based on true random documents; for example, there is no significant difference in Kullback-Leibler divergence when 300 documents are included. Models built with query-based sampling are of lower quality.

Of the three quality measures considered, ctf ratio and Kullback-Leibler divergence appear to be in agreement over the quality of models, across a range of sizes and sampling methods, while r_s does not agree with either.

With information collected ahead of time on the size and contents of each collection, a metasearch tool is able to select the most promising server or servers for each query. This process of server selection is the subject of the next chapter.

Chapter 7

Server selection

Given appropriate characterisations of the available servers, a metasearch tool may include *server selection* as part of its normal query processing. It is not generally feasible to forward every query to every server, as costs will be incurred in network traffic, delay, and an increased risk of unavailability, and server selection aims to choose the server or servers most useful for answering each query. As well as minimising costs and increasing reliability, there is some evidence that a selection algorithm can improve the quality of a result set even over that from a single index [Abbaci et al. 2002; Craswell et al. 2000; Hawking and Thistlewaite 1999; Powell et al. 2000; Xu and Croft 1999]. Server selection has also been called "collection selection" (for example, Callan et al. [1995]), "subcollection selection" [Baumgarten 1999], "text database discovery" [Gravano et al. 1994], "text-source discovery" [Gravano et al. 1999], and "database selection" [French et al. 1998]. See Section 2.2.2 on p. 13 for notes on terminology.

In past work several server selection algorithms have been proposed, and Section 7.1 discusses these algorithms, past evaluations, and other related work. Selection algorithms have been tested previously either with perfect characterisations (for example, French et al. [1998] and [1999], or Powell and French [2003]), or with characterisations from query-based sampling (Hawking and Thomas [2005], Shokouhi [2007], Si and Callan [2003b], and others). As demonstrated in Chapters 4 to 6, perfect characterisations are not generally possible and alternative sampling techniques outperform query-based sampling in a personal metasearch context. Typical testbeds for server selection evaluation have been based on TREC ad hoc or Web data, which is not representative of personal metasearch. Experiments in Section 7.2 therefore evaluate the performance of twelve algorithms in the testbed used in earlier chapters. Kullback-Leibler divergence (two variants), CORI (three variants), and ReDDE represent state-of-the-art server selection algorithms, which have been well tested in other scenarios; cue validity variance (CVV), GlOSS (two variants), CRCS (two variants) and Zobel's method "I" are also included.

Section 7.3 demonstrates that selection performance varies both between methods and between queries, with many methods prone to selecting large collections regardless of their utility to any given query. Most methods considered rely on good-quality language models, and poorer models — for example from less representative document samples — lead to poorer selection performance. Of the selection methods tested, Kullback-Leibler divergence and CORI are found to have the most promise for personal metasearch applications.

7.1 Related work

A large number of selection methods have been described, drawing on a variety of techniques including language modelling, classification, and characterising past performance. Table 7.1 summarises a number of these methods with regard to the information they use and the support needed from each server. Language models ("lang. model") are unigram in all cases, but different methods make use of absolute document or term frequency counts ("df", "tf"), or relative document or term frequencies ("rdf", "rtf") which have been normalised according to collection (or model) size; note that either of these can be generated from the other given a size estimate.

All the techniques described in this chapter select one or more collections from $\mathcal{C}$, the set of collections available, for each user query q. The majority, and all those considered in the experiments of this chapter, do this by scoring each possibility: given the query, they produce a score $s_c(q)$ for every collection c in $\mathcal{C}$.[1] A metasearch tool can rank the collections by this score, and choose either those collections with scores above a threshold or some number of top-scoring collections. The servers handling these collections are selected to receive the query.

Sections 7.1.1 to 7.1.14 describe a number of methods previously proposed,with particular attention to those methods which make few demands on servers and users and are therefore most likely to be useful in a personal metasearch environment.

As well as the evaluations carried out in the course of proposing selection methods, a number of researchers have reported evaluations of a set of methods or have considered the general applicability of these results. Section 7.1.15 discusses this work.

7.1.1 The GlOSS family

The "glossary of servers server" (GlOSS) family of algorithms [Gravano et al. 1994; Gravano and García-Molina 1995; Gravano et al. 1999] estimates the number of documents in each collection which a user will find interesting, and ranks collections accordingly. Three versions have been described: bGlOSS assumes a Boolean model, vGlOSS (originally called gGlOSS) assumes a vector-space model, and hGlOSS is a hierarchical version.

bGlOSS

The original GlOSS server selection algorithm, later called the Boolean version or bGlOSS [Gravano et al. 1994; Gravano et al. 1999], assumes that servers use a Boolean retrieval model and that the interesting documents in a collection are those which match all terms in the query q. If these terms, $\mathcal{T}_q$, appear independently, the expected

[1] In an earlier survey, Powell and French [2003] use the notation $merit(q, c)$.

	Information needed				
Method	Lang. model	Collection size	Query terms	Past performance	Server support
Language models					
bGlOSS	rdf	Docs	Needed	—	—
gGlOSS/vGlOSS	df, tf	Docs	Needed	—	—
CORI	df	Terms	Needed	—	—
(extension 1)	df	Terms, docs	Needed	—	—
(extension 2)	df	Terms, docs	Needed	—	—
Lexicon inspection	df	Docs	Needed	—	—
CVV	rdf, df	—	Needed	—	—
KL divergence	rdf, df	Docs	Needed	—	—
(extension)	rdf, df	Docs	Needed	—	—
Most similar doc	df, tf	Docs	Needed	—	—
ReDDE	df, tf†	Docs	Needed†	—	Doc text
Central-rank-based	df, tf†	Docs	Needed†	—	Doc text
Probes					
Lightweight probes	—	—	Needed	—	Probes
TRD-CS	—	—	Needed	—	Doc text
Probe queries‡	—	—	—	Doc ranks	Doc text
Classifications					
Profusion	—	—	Needed	Precision	Doc text
RDD	—	—	Needed	Precision	Doc text
QProber‡	—	—	—	—	—
Two-stage	rdf	—	Needed	—	—
Inquirus2	—	—	—	—	—
Others					
Decision-theoretic	—	—	—	R-P curve	—
SavvySearch	—	—	Needed	Various	—
UUM, RUM	dt, tf†	Docs	Needed	Relevance	Doc text
HARP, AWSUM	df, tf	Docs	Needed	—	Anchor-text
Manual selection	—	—	—	—	—
Controls					
Random	—	—	—	—	—
Size	—	Docs	—	—	—

Table 7.1: Characteristics of server selection methods. “Doc(s)” abbreviates “document(s)”; see the text for details of the “lang. model” column. “Language model methods” and “controls”are considered in the experiments of this chapter. † ReDDE, central-rank-based, and UUM/RUM selection depend on ranking documents from a sample; in the ReDDE and central-rank-based implementations used here, this ranking uses data on term occurrence, query terms, and on anchortext where available. ‡ QProber and probe queries are not complete server selection methods, but can be used with any other method.

number of interesting documents — those including all query terms — can be estimated by

$$\begin{aligned} s_c(q) &= N_c \prod_{t \in \mathcal{T}_q} \Pr(t|c) \\ &= N_c \prod_{t \in \mathcal{T}_q} \frac{\mathrm{df}_c(t)}{N_c}. \end{aligned} \tag{7.1}$$

Where $\mathrm{df}_c(t)$ is the document frequency of term t and N_c is the size of collection c, as before. As originally described [Gravano et al. 1994], bGlOSS selects only the collection with the highest score, or all top-scoring collections in the case of a tie. In the experiments described here, it is used in a more conventional way to rank all available collections.

Gravano et al. suggested two definitions of success for the selection problem: either choosing *all* the best collections, where the "best" collections are those with the highest number of matching documents, but possibly choosing other collections as well; or choosing *only* the best collections, possibly missing some. (These are analogous to high-recall and high-precision tasks.) Using these two definitions, six collections, and almost 7000 queries from a Stanford library system, they reported success rates of 84% and 89%; choosing between only two collections success rates were higher still at 92% and 99%.

Although the servers used in the experiments below do not use a Boolean model, bGlOSS is similar to other selection techniques based on different assumptions and is therefore included for comparison.

gGlOSS and vGlOSS

The "generalised" version of GlOSS, gGlOSS [Gravano and García-Molina 1995] — later called the vector-space version, vGlOSS [Gravano et al. 1999] — assumes users are interested not in documents containing all query terms but in those documents whose similarity to the query is more than a threshold l. To estimate this, vGlOSS makes use of a collection-specific weight, $w_c(t)$, for each term t in c. In the original presentation, this weight is calculated at the server, for example using term frequency and inverse document frequency; it is then made available, along with document frequency information, to the metasearcher [Gravano and García-Molina 1995]. In a metasearch application without cooperative servers, however, this data is more likely to be estimated from a sample.

Gravano and García-Molina [1995] suggest two algorithms for vGlOSS to estimate the score of a collection. The first, which assumes a "high correlation scenario", makes two assumptions:

1. If $\mathrm{df}_c(t_1) \leq \mathrm{df}_c(t_2)$, or t_1 is rarer than t_2 in c, then t_2 appears in all the documents that t_1 appears in. For example, if the term "information" appears in 10 documents and "retrieval" in 7, then there are 7 documents with both "information" and "retrieval" and $10 - 7 = 3$ with "information" alone. As the rarer

$$\begin{array}{c} \mathrm{df}_c(t_1) \; < \; \mathrm{df}_c(t_2) \; < \cdots < \; \mathrm{df}_c(t_p) \; < \; \mathrm{df}_c(t_{p+1}) \; < \cdots < \; \mathrm{df}_c(t_n) \\ \underbrace{\textsc{res}_c(t_1) \subseteq \textsc{res}_c(t_2) \subseteq \cdots \subseteq \textsc{res}_c(t_p)}_{\substack{\text{these documents} \\ \text{similar enough to } q}} \subseteq \underbrace{\textsc{res}_c(t_{p+1}) \subseteq \cdots \subseteq \textsc{res}_c(t_n)}_{\substack{\text{these documents} \\ \text{not similar enough}}} \end{array}$$

Figure 7.1: Similarity under the high correlation scenario of Gravano and García-Molina [1995]. The $\mathrm{df}_c(t_p)$ documents containing terms t_p to t_n are similar enough to the query (so $\textsc{sim}_c(q,p) > l$), but the $\mathrm{df}_c(t_{p+1})$ documents containing only terms t_{p+1} to t_n are not (so $\textsc{sim}_c(q,p+1) \leq l$). Equation 7.2 defines $\textsc{sim}_c(q,p)$.

term, no documents will have "retrieval" alone. It follows that $\mathrm{df}_c(t_i) - \mathrm{df}_c(t_{i-1})$ documents include *only* term t_i, with the convention that $\mathrm{df}_c(t_0) = 0$.

2. The weight of a term in a collection, $w_c(t)$, is distributed uniformly over all documents which contain the word. To continue the example, if $w_c(\text{"retrieval"}) = 0.35$, then all 7 documents matching "retrieval" do so with weight $0.35/7 = 0.05$.

Using these two assumptions, Gravano and García-Molina [1995] describe the function $\textsc{max}(l,q,c)$, which scores collections according to the estimated number of documents with similarity to the query of more than l. The algorithm starts by reordering the query terms according to document frequency in c, so $\mathrm{df}_c(t_1) \leq \mathrm{df}_c(t_2) \leq \cdots \leq \mathrm{df}_c(t_n)$. (Continuing the example above, the query ⟨"information", "retrieval"⟩ will be reordered as ⟨"retrieval", "information"⟩ since "retrieval" is the less common term.) Note that, by assumption, the $\mathrm{df}_c(t_i)$ documents which contain term t_i also contain the other $n-i$ terms; and, also by assumption, the collection weight $w_c(t_i)$ is evenly distributed amongst these documents. It is therefore possible to define $\textsc{sim}_c(q,i)$, the similarity between query q and the documents containing terms t_1 to t_i, with

$$\textsc{sim}_c(q,i) = \sum_{j=i\ldots n} w_q(t_j)\,\frac{w_c(t_j)}{\mathrm{df}_c(t_j)}, \tag{7.2}$$

where $w_q(t)$ is the weight of term t in the query.

This similarity function, plus the threshold l, are used to estimate the total number of useful documents in the collection. The $\mathrm{df}_c(t_1)$ documents which include term t_1, the rarest term, by assumption also include all other terms in q; since they contain all terms, they are the documents most similar to q. If these documents are not similar enough to q, if $\textsc{sim}_c(q,1) \leq l$, there are no sufficiently similar documents in the collection. Otherwise, when $\textsc{sim}_c(q,1) > l$, it is possible that there are other documents, without term t_1 but with terms t_2 to t_n, which are also sufficiently similar. The same observation holds for $\textsc{sim}_c(q,2)$, and so on to $n-1$, and the general problem is to find a cut-off p such that $\textsc{sim}_c(q,p) > l$ but $\textsc{sim}_c(q,p+1) \leq l$. Figure 7.1 illustrates this cut-off.

Having found cut-off p, observe that the $\mathrm{df}_c(t_p)$ documents which are assumed to contain all of terms t_p to t_n are sufficiently similar to the query to be interesting; the $\mathrm{df}_c(t_{p+1})$ documents containing only terms t_{p+1} to t_n are not. Since (by assumption) the weight due to each term is evenly distributed amongst all documents containing that term, it is possible to assign a score to the documents containing *only* term t_i: there are $\mathrm{df}_c(t_i) - \mathrm{df}_c(t_{i-1})$ of these, so Gravano and García-Molina [1995] let the score due to these documents be $(\mathrm{df}(t_i) - \mathrm{df}(t_{i-1}))\,\textsc{sim}(q,i)$. Finally, we can assign an overall score for the collection by summing these scores over all terms 1 to p:

$$\begin{aligned} s_c(q) &= \textsc{max}(l,q,c) \\ &= \sum_{i=1\ldots p} (\mathrm{df}_c(t_i) - \mathrm{df}_c(t_{i-1}))\,\textsc{sim}_c(q,i). \end{aligned}$$

The second estimator of Gravano and García-Molina [1995] models disjoint term occurrences. It also makes two assumptions about term distribution:

1. For any two query terms t_1 and t_2, the set of documents containing t_1 is disjoint from that containing t_2; no document contains both.
2. The weight of a term is distributed uniformly over all documents which contain the term, as before.

Again, $t_1 \ldots t_n$ represent the n terms in query q, although in this case they need not be reordered. For each of these terms t_i, there are by definition $\mathrm{df}_c(t_i)$ documents in collection c containing this term. The similarity of each of these documents to q is entirely due to the one query term it has: since the collection weight of a term is assumed to be evenly distributed, we can let this similarity be $w_q(t_i)\,w_c(t_i)/\,\mathrm{df}_c(t_i)$.

Scoring is then very simple: the score of a collection is just the sum of similarities of each matching document, provided that similarity is greater than the threshold l.

$$\begin{aligned} s_c(q) &= \textsc{sum}(l,q,c) \\ &= \sum_{\substack{i=1\ldots n \\ w_q(t_i)w_c(t_i)/\,\mathrm{df}_c(t_i)>l}} \mathrm{df}_c(t_i)\,w_q(t_i)\,\frac{w_c(t_i)}{\mathrm{df}_c(t_i)} \\ &= \sum_{\substack{i=1\ldots n \\ w_q(t_i)\,w_c(t_i)/\,\mathrm{df}_c(t_i)>l}} w_q(t_i)w_c(t_i). \end{aligned}$$

A common application considers any document at all similar to the query to be useful, and sets $l = 0$ [Craswell et al. 2000; French et al. 1999; Powell and French 2003]. In this case, MAX and SUM are identical and document frequencies are not needed.

$$\begin{aligned} s_c(q) &= \textsc{max}(0,q,c) \\ &= \textsc{sum}(0,q,c) \\ &= \sum_{t\in\mathcal{T}_q} w_q(t)\,w_c(t). \end{aligned} \tag{7.3}$$

A series of experiments [Gravano and García-Molina 1995; Gravano et al. 1999] evaluated vGlOSS on a 53-collection testbed based on Usenet newsgroups. Gravano et al. concluded that the best choice of threshold l, and the choice of MAX or SUM as a scoring function, depended on the task: for high-precision tasks, SUM with $l = 0.2$ represented a good choice and for high-recall tasks, $l = 0$ (with MAX and SUM identical) found more relevant documents. In general, vGlOSS appeared to select collections fairly well. Later work, however, has found vGlOSS less effective than alternatives [Craswell et al. 2000; French et al. 1998; French et al. 1999; Powell and French 2003; Yuwono and Lee 1997]; the difference in evaluations may be due to a difference in measures (see Section 7.2.3, p. 115). The experiments reported here, in common with most other work, found vGlOSS to be fairly ineffective.

hGlOSS

Higher-level GlOSS, or hGlOSS, is a distributed variant of the vGlOSS algorithm which selects between vGlOSS servers in the same way as vGlOSS selects between collections [Gravano and García-Molina 1995]. A user query is first sent to the hGlOSS server, then to promising vGlOSS servers, then to promising collections; in principle this could be extended to arbitrary numbers of hGlOSS servers operating in a hierarchy.

Experiments by Gravano and García-Molina used an analogue of MAX to score vGlOSS servers. Document frequency $\mathrm{df}(t)$ was replaced by collection frequency $\mathrm{cf}(t)$, and term weights $w_c(t)$ with the total number of matching documents across all collections. Performance, as measured by an analogue of recall, was good: selecting between five vGlOSS servers, based on the 53 collections described above, hGlOSS achieved recall of 99% with only one server selected.

Since it is unlikely that there will be a hierarchy of servers in a personal metasearch application, hGlOSS is not considered in the experiments of this chapter.

7.1.2 CORI

The collection retrieval inference (CORI) algorithm [Callan et al. 1995] adapts the INQUERY document ranking algorithm [Callan et al. 1995; Turtle and Croft 1991]. It treats each server as a compound "document", using document frequency (df) instead of term frequency and collection frequency (cf) in place of document frequency. Each term in the query is scored separately, according to document frequency (T) and inverse collection frequency (I):

$$\begin{aligned} s_c(t) &= b + (1-b)TI; \\ \text{where } T &= \frac{\mathrm{df}_c(t)}{\mathrm{df}_c(t) + \mathit{df\_base} + \mathit{df\_factor}\,(cw_c/\overline{cw})} \\ \text{and } I &= \frac{\log\left(\frac{|\mathcal{C}|+0.5}{\mathrm{cf}_{\mathcal{C}}(t)}\right)}{\log(|\mathcal{C}|+1)}. \end{aligned}$$

As before, $\mathrm{df}_c(t)$ is the number of documents in collection c which contain term t, cw_c is the number of terms in the collection, and $\overline{cw}$ is the mean number of terms across all collections. By analogy with $\mathrm{tf}_c(t)$ and $\mathrm{df}_c(t)$, $\mathrm{cf}_{\mathcal{C}}(t)$ is the number of collections containing t. The other terms are constants: $b = 0.4$, *df_base* $= 50$, and *df_factor* $= 150$. The score for a query is the mean of the per-term scores, $s_c(q) = \sum_{t \in \mathcal{T}_q} s_c(t)/|q|$.

CORI has been widely used, and has proven effective in selection tasks based on TREC ad hoc data [Callan et al. 1995; French et al. 1999; Powell et al. 2000; Powell and French 2003] and Web data [Craswell et al. 2000; Hawking and Thomas 2005], although Si and Callan [2003a, 2003b] and D'Souza et al. [2004a, 2004b] show poorer performance when collection sizes are highly variable. CORI has also been included in the decision-theoretic framework of Nottelmann and Fuhr [2004] (Section 7.1.13).

D'Souza et al. [2004a, 2004b] have examined the performance of CORI across a variety of testbeds, based on TREC ad hoc data but with varying characteristics. In these studies the best values for b, *df_base*, and *df_factor* — given as constants above — vary greatly with each testbed, query set, and even from query to query. Even with the best settings, CORI was often inferior to other, simpler, methods (including the "H" and "I" methods of Section 7.1.4 below). D'Souza et al. conclude that CORI's good performance is due to the particular testbeds used to date and that it is not possible to tune CORI for general use. (Earlier work by French et al. [1999] had concluded that CORI was robust to changes in these three parameters, but had only considered one testbed.)

7.1.3 Extended CORI

As described above, CORI relies on term frequency data from each collection. In practice, this will be estimated from language models which may be of widely varying size both compared with each other and compared with the underlying collections. With this in mind, later work by Si and Callan [2003a] suggested two extensions to the basic CORI algorithm, both of which make use of the N_m documents sampled from the collection in the course of building a language model. There are four modifications.

1. $\mathrm{df}_c(t)$ counts the number of documents which contain term t. In practice, this can be estimated from $\mathrm{df}_m(t)$, the number of documents in the model which contain t, and a scaling ratio: $\mathrm{df}_c(t) \approx \mathrm{df}_m(t) \times (N_c/N_m)$;

2. cw_c, the number of terms in the collection, can also be estimated by scaling: so $cw_c \approx cw_m \times (N_c/N_m)$. $\overline{cw}$ can be scaled similarly. (Note however that Heaps's law suggests a linear growth in vocabulary is unlikely, at least for text in English [Williams and Zobel 2005].)

3. *df_base* can also be scaled by N_c/N_m;

4. finally, so can *df_factor*.

In the experiments below, the basic CORI technique uses term information from models and size estimates from one of the techniques of Chapter 5. Following Si and Callan [2003a], "CORI (extension 1)" uses the scaling of modifications 1 and 2 above, and "CORI (extension 2)" uses all of modifications 1 to 4. Since these variants expressly adjust for estimated size, they may be expected to perform better over the large variety of collections in personal metasearch, although Si and Callan found inconsistent performance over four testbeds based on TREC ad hoc data.

7.1.4 Lexicon inspection

Zobel [1997] sets out four desiderata for selection algorithms. According to this list, collections should be ranked highly if and only if they contain the query terms; the terms are common relative to other collections; the collection contains a high proportion of documents with the terms; and the collection is likely to contain documents which feature the terms frequently.

On the basis of these criteria, Zobel describes and evaluates four simple selection algorithms: "C", "S", "H", and "I". Algorithm "C" derives from the cosine similarity measure; it is the sum, for each query term t, of $w_q(t)w_c(t)$ with normalisation for query or document length. Algorithm "C" uses information about each collection in isolation; the "skew" algorithm, "S", uses information about term occurrence across collections to try to identify terms which are common in c but uncommon elsewhere. Algorithm "H" again makes use of the cosine measure to estimate the highest likely similarity for an average-length document from c.

Experiments using two testbeds, one from the TREC data fusion track [Voorhees 1995] and one based on a random assignment of documents from TREC ad hoc data, showed that these measures were able to outperform a static ranking based on size alone (although performance differed according to whether similar documents, or documents which had been judged relevant, were considered useful). Algorithms "C", "S", and "H" were however consistently outperformed by Zobel's algorithm "I", a simple inner product without length normalisation:

$$s_c(q) = \sum_{t \in \mathcal{T}_q} w_q(t)\, w_c(t), \tag{7.4}$$

where the weights are based on analogues of term frequency and inverse document frequency: the query weight $w_q(t) = \log(\mathrm{tf}_q(t) + 1)w(t)$; the collection weight $w_c(t) = \log(\mathrm{df}_c(t) + 1)w(t)$; and, as an analogue of inverse document frequency, $w(t) = \log(N/\,\mathrm{df}_{\mathcal{C}}(t) + 1)$. Here $\mathrm{df}_{\mathcal{C}}(t)$ describes the total number of documents containing a term t across all collections in $\mathcal{C}$. The function in Equation 7.4 is identical to $\textsc{max}(0)$ and $\textsc{sum}(0)$ from vGlOSS (Equation 7.3), although in the case of vGlOSS weights are calculated with collection-specific data only and may vary, even for the same term, from collection to collection. In the experiments described below, this difference in weights results in a significant difference in performance.

Zobel's second testbed included 91 collections which varied in size by three orders of magnitude, from 14 documents to 23,000. This wide range of sizes is likely to be typical of personal metasearch applications, which suggests that algorithm "I" may be of general use. This is explored further in the experiments below.

Moffat and Zobel [1994] earlier suggested a similar scheme, designed for a single large collection, in which "blocks" of several documents each were indexed and ranked as if they were single large documents — a simple ranking was suggested based on cosine similarity — and the highest-ranked blocks were selected for further processing. Moffat and Zobel were however interested in efficient searches over single indexes, not in metasearch, and used a fixed number of documents per block rather than the highly variable distribution more typical of metasearch. Experiments with the TREC ad hoc collection and blocks of one to 1000 documents showed that as long as small numbers of documents were requested, this blocking was effective at reducing effort at the expense of some loss of effectivness. The blocking scheme was however less effective on tasks which required more documents to be retrieved. Blocking was not used in a metasearch context, but is similar to Zobel's algorithm "C" above.

7.1.5 CVV

Cue validity variance (CVV), introduced by Yuwono and Lee [1997], is a server selection technique which weights terms according to their power to discriminate between collections. The technique begins by calculating the *cue validity* of each term t at each collection c, $\mathrm{CV}(t,c)$, which measures the extent to which t distinguishes c from other collections [Goldberg 1995]:

$$\mathrm{CV}(t,c) = \frac{\Pr(t|c)}{\Pr(t|c) + \Pr(t|\mathcal{C} \setminus c)}.$$

As before, $\Pr(t|c)$ can be approximated by the relative frequency of t in c, $\mathrm{df}_c(t)/N_c$. $\Pr(t|\mathcal{C} \setminus c)$, the probability of t occurring in a "contrasting concept", is the relative frequency of t in all *other* collections: $\sum_{k \in \mathcal{C} \setminus c} \mathrm{df}_k(t) / \sum_{k \in \mathcal{C} \setminus c} N_k$. (In neither case is any smoothing used, which means $\mathrm{CV}(t,c)$ is undefined for any term which is not present in any collection. Yuwono and Lee [1997] do not appear to have considered this possibility. The implementation used in the experiments described below assigns $\mathrm{CV}(t,c) = 0$ in such cases, meaning the term will not contribute to any collection's score.)

The cue validity of t at c gives an indication of how well t distinguishes documents in c from documents from all other collections. $\mathrm{CVV}(t)$, the cue validity variance of term t, is a measure of how useful a term is in distinguishing collections in $\mathcal{C}$ in general. $\mathrm{CVV}(t)$ is just the variance of $\mathrm{CV}(t,c)$:

$$\mathrm{CVV}(t) = \frac{\sum_{c \in \mathcal{C}} \left(\mathrm{CV}(t,c) - \overline{\mathrm{CV}(t)} \right)^2}{|\mathcal{C}|}.$$

where $\overline{\mathrm{CV}(t)}$ is the mean value of $\mathrm{CV}(t,c)$ across all collections, $\sum_{c \in \mathcal{C}} \mathrm{CV}(t,c)/|\mathcal{C}|$.

Finally, collections are scored according to the frequency of each query term, each weighted by its cue validity variance so that terms with more discriminative power are considered more important:

$$s_c(q) = \sum_{t \in \mathcal{T}_q} \mathrm{CVV}(t)\, \mathrm{df}_c(t).$$

Initial evaluation by Yuwono and Lee [1997] compared CVV, bGlOSS, vGlOSS, CORI, and other baseline methods on a four-collection, 7000-document testbed from the SMART system [Salton and McGill 1983] with simulated vector-space retrieval. bGlOSS performed relatively poorly in this vector-space environment, and the remaining methods were outperformed by CVV across a range of conditions from homogeneous to heterogeneous collections. With simulated Boolean retrieval over the same collections, CVV appeared slightly less effective than bGlOSS but the other techniques fared poorly.

This initial result has been called into question by later experiments by Powell and French [2003]. This later work repeated the comparison on three different testbeds drawn from TREC ad hoc collections. Unlike Yuwono and Lee, they found that CVV performed close to, but worse than, CORI. Craswell et al. [2000], experimenting with their "probe queries" technique (Section 7.1.12), also found CVV was poorer at the selection task than other methods. Craswell et al. suggest that the assumption underlying CVV — that terms with more variability are more useful to discriminate between collections — may lead to poor results when a term is very rare. In these cases, a term may appear a small number of times in a small number of collections, and otherwise not at all; this leads to a low CVV score and little emphasis on what could have been a very useful term.

7.1.6 Kullback-Leibler divergence

Kullback-Leibler divergence, described earlier as a means to compare two language models, was suggested for server selection by Xu and Croft [1999] and interpreted in a language modelling framework by Si et al. [2002]. The language modelling interpretation, following Ponte and Croft [1998], ranks each collection c according to its probability of being generated given a model of query q: i.e., $s_c(q) = \Pr(c|q)$. By Bayes's theorem,

$$\begin{aligned} s_c(q) &= \Pr(c|q) \\ &= \frac{\Pr(q|c)\Pr(c)}{\Pr(q)}. \end{aligned} \tag{7.5}$$

Note that $\Pr(q)$, the prior probability of the query, does not depend on the collection and can therefore be ignored. Similarly, $\Pr(c)$, the prior probability of collection c, is normally considered constant and can also be ignored.

With these two simplifications, and assuming that terms occur independently, we can say

$$\Pr(q|c) = \prod_{t \in \mathcal{T}_q} \Pr(t|c). \tag{7.6}$$

(Compare this with Equation 6.1, p. 76, which ranks documents.) $\Pr(t|c)$ can be estimated from a model m as usual with $\mathrm{df}_m(t)/N_m$.

This can be shown to rank collections equivalently to Kullback-Leibler divergence, given that collections with lower divergence are more similar to the query and should be ranked first. Recall from the discussion of divergence in Section 6.2.2 on p. 82 that

$$\mathrm{D}_{KL}(q \| c) = \sum_{t \in \mathcal{T}_q} \Pr(t|q) \log_2 \frac{\Pr(t|q)}{\Pr(t|c)} \tag{7.7}$$

(this is Equation 6.2, with queries substituted for collections and collections for models). As before, $\Pr(t|q)$ depends only on the query, not the collection, and can be ignored for the purpose of ranking collections; what remains from Equation 7.7 is proportional to $-\sum_{t \in \mathcal{T}_q} \log_2 \Pr(t|c)$. Taking the logarithm of Equation 7.6 — which will not modify the ranking — gives $\sum_{t \in \mathcal{T}_q} \log_2 \Pr(t|c)$. Ranking by Equation 7.6 is therefore equivalent to ranking by (negative) Kullback-Leibler divergence.

As before, smoothing can be used to account for infrequent terms. Si et al. [2002] use a global model, m_g, which captures term occurrences across all collections:

$$\Pr(t|m_g) = \frac{\sum_{c \in \mathcal{C}} \mathrm{df}_c(t)}{\sum_{c \in \mathcal{C}} N_c}.$$

Term data from this global model is then combined with data from each collection in a modification of Equation 7.6:

$$\Pr(q|c) = \prod_{t \in \mathcal{T}_q} \left(\lambda \Pr(t|c) + (1 - \lambda) \Pr(t|m_g) \right). \tag{7.8}$$

The parameter λ controls the mixing and is typically 0.5 [Si et al. 2002; Si and Callan 2003a]. Note that when $\lambda = 1$, Kullback-Leibler divergence ranks equivalently to bGlOSS.

Experiments with Kullback-Leibler divergence have had mixed results. Xu and Croft [1999], using a 100-collection testbed based on TREC ad hoc data, reported recall of around 50% optimal at 10 collections selected, significantly worse than a single centralised index. On a similar testbed, Si et al. [2002] reported precision of about 40%, although this represented better performance than CORI on the same data. Further experiments by Si and Callan [2003a] found that Kullback-Leibler divergence performed poorly on testbeds with a few relatively large collections, which prompted the development of an extension.

7.1.7 Extended Kullback-Leibler divergence

Si and Callan [2003a] observe that there is no explicit control for collection size in the Kullback-Leibler method described above. Their extended algorithm assumes that larger collections are more likely to be relevant and assigns the prior probability of collection c being relevant, $\Pr(c)$, according to collection size:

$$\Pr(c) = \frac{N_c}{\sum_{i \in \mathcal{C}} N_i}.$$

Otherwise, this variant uses Equations 7.5 and 7.8 as before. The experiments below consider this alternative as "extended Kullback-Leibler divergence".

Evaluation of this extension by Si and Callan [2003a], over TREC ad hoc data, found it improved recall on testbeds where collection sizes were particularly skewed; the impact was less when collection sizes were more homogenous. Further evaluation by Hawking and Thomas [2005], over TREC Web track data with a wide variety of collection sizes, demonstrated extended Kullback-Leibler divergence performed well across the board but especially with "topic distillation" tasks where the goal was to return a set of key Web pages.

7.1.8 Most similar document

The most similar document method, like vGlOSS, tries to locate documents which are similar to the query on the assumption that these are the most likely to be relevant [Meng et al. 2001; Yu et al. 1999]. The algorithm has been described in two forms: one uses a model of each collection as well as a global model describing term occurrences overall; another uses a hierarchy of models, from those representing individual collections through those representing several collections up to the global model. Only the first of these appears to have been implemented.

For each term t in each collection c, the algorithm records two pieces of data: the maximum normalised weight $\mathrm{mnw}_c(t)$ and the average weight $\mathrm{aw}_c(t)$. The maximum normalised weight, $\mathrm{mnw}_c(t) = \max_{d \in \mathcal{D}_c}(\mathrm{tf}_d(t)/|d|)$, is the highest density of occurrences of t in any document in the collection. $\mathrm{aw}_c(t)$ is the average of these proportions over all documents in the collection, even those which do not contain t: $\mathrm{aw}_c(t) = (\sum_{d \in \mathcal{D}_c} \mathrm{tf}_d(t)/|d|)/N_c$. The intuition is that the most similar document to a query term is likely to be that with the maximum normalised weight for that term. Weighting each term according to a variant of tf·idf, $\mathrm{tf}_q(t)/\,\mathrm{df}_{m_g}(t)$ (recall that m_g is the "global" model), collections are scored with[2]

$$s_c(q) = \max_{t \in \mathcal{T}_q} \left(\frac{\mathrm{tf}_q(t)}{\mathrm{df}_{m_g}(t)} \mathrm{mnw}_c(t) + \sum_{t' \in \mathcal{T}_q \setminus t} \frac{\mathrm{tf}_q(t')}{\mathrm{df}_{m_g}(t')} \mathrm{aw}_c(t') \right).$$

[2] Yu et al. [1999] normalised these scores by dividing by the number of terms in the query q. Since this normalisation does not effect the final selection, it is ignored here.

The first part of the bracketed expression is the maximum normalised weight for this term; this is, by assumption, the weight of this term in the most similar document. The second part accounts for other terms, assuming they appear with average frequency. In both parts, terms are weighted as above. The best score for any one query term is used for the score of the collection as a whole.

Experiments by Yu et al. [1999] used the same collections and queries as Gravano and García-Molina [1995]. For the same number of documents retrieved, the most similar document method consistently retured more documents similar to the query than the high correlation (MAX) version of vGlOSS. It is not however clear exactly which variant of MAX was used in these experiments.

This method is not included in these experiments, since it is not possible to deduce $\mathrm{mnw}_c(t)$ from the unigram language models used to date.

7.1.9 ReDDE

The relevant document distribution estimation method, or ReDDE, is also due to Si and Callan [2003b]. As with the GlOSS family, ReDDE attempts to estimate the distribution of relevant documents across all collections based on a simple approximation of relevance. The algorithm computes an estimate of $|\textsc{rel}_c(q)|$, the number of documents in c which are relevant to the query q; this forms the basis of the eventual ranking.

$$\widehat{|\textsc{rel}_c(q)|} = \sum_{d \in \mathcal{D}_c} \Pr(\text{relevant}|d) \Pr(d|c) N_c$$

(Recall that $\widehat{\ }$ represents an estimate and that $\mathcal{D}_c$ is the set of documents in collection c.) If m, the model of c, is representative, it is possible to estimate $\Pr(d|c)$ by assuming a uniform distribution: let $\Pr(d|c) = 1/N_m$. Still assuming that the model is representative, we can also estimate $\Pr(\text{relevant}|d)$ using the documents sampled when building the model, $\mathcal{D}_m$:

$$\widehat{|\textsc{rel}_c(q)|} = \frac{N_c}{N_m} \sum_{d \in \mathcal{D}_m} \Pr(\text{relevant}|d) \tag{7.9}$$

Note that this is the expected number of relevant documents; the formulation is similar to that used by bGlOSS (Equation 7.1). The remaining task is to estimate the probability that each document d in $\mathcal{D}_m$ is relevant to q.

ReDDE estimates this from a hypothetical ranking of all documents in all collections. Each of the top-ranked documents in this complete ("central") list has a fixed chance of relevance and contributes to the score of the collection it comes from:

$$\Pr(\text{relevant}|d) = \begin{cases} k & \text{if } \textsc{rank\_central}(d) < rN_{\text{all}} \\ 0 & \text{otherwise} \end{cases} \tag{7.10}$$

where $\textsc{rank\_central}(d)$ is the rank, over all documents in all collections, of d; and N_{all} is the total number of documents in all collections, $\sum_{c \in \mathcal{C}} N_c$. The parameters r and k are tunable.

Collection A:	Twelve sampled documents, $a1$ to $a12$. Total size 300,000 documents.
Collection B:	Four sampled documents, $b1$, $b2$, $b3$, and $b4$. Total size 3,000,000 documents.

Step	Calculations		Score
1	$\textsc{rank\_sample}(a1) = 0;$	$\textsc{rank\_central}(a1) = 0$	$A \leftarrow 1$
2	$\textsc{rank\_sample}(a2) = 1;$	$\textsc{rank\_central}(a2) = 25,000$	$A \leftarrow 1$
3	$\textsc{rank\_sample}(b1) = 2;$	$\textsc{rank\_central}(b1) = 50,000$	$B \leftarrow 1$
4	$\textsc{rank\_sample}(b2) = 3;$	$\textsc{rank\_central}(b2) = 850,000$	—

Figure 7.2: Example calculations for the ReDDE selection algorithm. See the text for details.

To compute this complete ranking would require a copy of every document in every collection, which is of course infeasible. $\textsc{rank\_central}(d)$ can instead itself be estimated using the samples collected in the course of building models for the collections. All documents sampled from all collections are indexed by the metasearch tool itself, and ranked for each query by some effective method; the ranking of sample document d, from a subset $\mathcal{D}_m$ of $\mathcal{D}_c$, is $\textsc{rank\_sample}(d)$. With this sample rank, $\textsc{rank\_central}(d)$ can be estimated from the total number of documents ranked ahead of d: the intuition is that each document sampled from c stands for N_c/N_m documents in the complete ranking.

$$\widehat{\textsc{rank\_central}}(d) = \sum_{\substack{d' \in \mathcal{D}_m \\ \textsc{rank\_sample}(d') < \\ \textsc{rank\_sample}(d)}} \frac{N_{c_{d'}}}{N_{m_{d'}}}, \tag{7.11}$$

where $c_{d'}$ is the collection from which d' is drawn, and $m_{d'}$ the sample from which d' is drawn.

Substituting the estimates in Equations 7.10 and 7.11 into Equation 7.9 lets us estimate the total number of documents in each collection which are relevant to q. The final score is a normalised version of this:

$$s_c(q) = \frac{|\widehat{\mathrm{REL}_c(q)}|}{\sum_{c' \in \mathcal{C}} |\widehat{\mathrm{REL}_{c'}(q)}|.}$$

For example, consider the problem of selecting between the two collections A and B of Figure 7.2. First, given a query q, the sixteen sampled documents are ranked; assume the result of this ranking is $\langle a1, a2, b1, b2, b3, b4, a3 \ldots a12 \rangle$. N_{all}, the total number of documents across all collections, is $3,300,000$. For this example we use $k = 1$ and $r = 0.03$, which means only the top 3% of documents, or $99,000$ documents, will contribute to the final score (Si and Callan [2003b] use k between 0.002 and 0.005, but with much larger samples).

The highest-ranked sample document is *a1*. By definition, $\text{RANK\_CENTRAL}(a1) = 0$, and collection A is awarded a score of 1.

The next sample document is *a2*. Since there are twelve documents in the sample *a1* came from ($N_{m_{a1}} = 12$), representing a collection of 300,000 documents ($N_{c_{a1}} = 300,000$), there are an estimated $300,000/12 = 25,000$ documents above *a2* in the complete ranking and $\text{RANK\_CENTRAL}(a2) = 25,000$. This is less than the 99,000 document threshold, so collection A is again awarded a score of 1.

The third highest-ranked sample document is *b1*. The first 25,000 documents in the complete ranking are already assumed to be from A; and again $N_{c_{a2}}/N_{m_{a2}} = 25,000$, so there are an estimated 50,000 documents ahead of *b1* in the complete ranking. This is still less than the 99,000 document threshold, so collection B is credited for including relevant documents.

The next document, *b2*, is at rank 3 on the sample list but is estimated to be at rank $300,000/6 + 300,000/6 + 3,000,000/4 = 850,000$ in the complete list. This is well past the 99,000 document threshold, so *b2* and all lower-ranked documents are assumed to be insufficiently interesting and no collection will gain any further score. The final scores are 2 for A and 1 for B, normalised to $\frac{2}{3}$ and $\frac{1}{3}$.

Si and Callan [2003a, 2003b] used a variety of testbeds based on TREC ad hoc data to compare ReDDE to CORI (and extensions) and Kullback-Leibler divergence (and extensions). Methods were evaluated on recall: the ability to find collections with the highest number of relevant documents. On those testbeds where collection sizes were more or less uniform, ReDDE appeared to perform as well as or slightly better than CORI; on those testbeds where collection size was more variable, however, with some collections both larger than others and with more relevant documents, ReDDE performed markedly better than all alternatives except extended Kullback-Leibler divergence. Further evaluation by Hawking and Thomas [2005] also found ReDDE to perform well across a variety of Web tasks.

7.1.10 Central-rank-based

The central-rank-based collection selection (CRCS) algorithms [Shokouhi 2007] are similar to ReDDE, and also make use of an index of sample documents. As for ReDDE, these documents are assumed to be representative of the collections they are drawn from; they are ranked for each query, and those collections which contribute highly-ranked sample documents are selected.

ReDDE awards collections a fixed score for each highly-ranked sample document (Equation 7.10). Shokouhi notes that this does not reflect the documents' likely utility — when ordered by an effective system, top-ranked documents are generally more useful than those of lower ranks [Joachims et al. 2005; Manmatha et al. 2001]. CRCS therefore allocates a rank-based, rather than fixed, score to each of the top γ sample documents. Shokouhi gives two variations. In the linear version, CRCS(l),

$$R(d) = \begin{cases} \gamma - \text{RANK\_SAMPLE}(d) & \text{if } \text{RANK\_SAMPLE}(d) < \gamma \\ 0 & \text{otherwise,} \end{cases}$$

where $\textsc{rank\_sample}(d)$ is the rank of sample document d, as before. In the exponential version, CRCS(e),

$$R(d) = \begin{cases} \alpha \, e^{-\beta \, \textsc{rank\_sample}(d)} & \text{if } \textsc{rank\_sample}(d) < \gamma \\ 0 & \text{otherwise.} \end{cases}$$

These per-document scores are summed for each collection, in the same manner as ReDDE, and a final score is calculated with

$$s_c(q) = \frac{N_c}{N_{\max} N_m} \sum_{d \in \mathcal{D}_m} R(d).$$

$N_{\max}$ is the size of the largest collection, and is used to normalise the scores. Besides this normalisation, CRCS is very similar to ReDDE. Where ReDDE expressly rewards collections with a large number of high-ranked documents, however, CRCS rewards collections with both large numbers of documents and documents with particularly high ranks.

Experiments by Shokouhi [2007] used a variety of testbeds, including a 100-server subset of the .GOV2 Web crawl,[3] and compared CRCS(l) and CRCS(e) with both CORI and ReDDE. The two CRCS algorithms appeared generally as good as ReDDE, although the exponential version CRCS(e) was somewhat better on two of the six testbeds. Differences in performance were however small.

Of the other algorithms, CORI performed poorly on the "relevant" testbed, where two large collections held the majority of relevant documents, and ReDDE performed well on the TREC4-kmeans testbed [Xu and Croft 1999], where collections were topically focussed.

7.1.11 Query-time probes

The technique of Hawking and Thistlewaite [1999] collects information on term occurrences at query time, and uses this to score collections. Servers are required to support special *lightweight probes*, sent for each user query; in response to these probes, each server returns statistics including the number of documents in the collection, the number of documents which contain each probe term, the number of documents in which terms co-occur, and the number of documents which contain probe terms within a specified proximity. These statistics are combined to generate a score $s_c(q)$, which is used to rank the available collections.

Hawking and Thistlewaite suggested two ways to generate the terms used in a probe: manual assignment and an automatic system which chooses terms with low expected frequency. In a series of experiments, the manual variant of lightweight probes achieved the highest recall of four selection techniques, and this performance appeared to be generalisable over a variety of collections. These experiments however used very long queries (mean 65 terms), which is not typical of current environments.

[3] `http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm`

The Top-Ranked Documents for Collection Selection algorithm, TRD-CS [Rasolofo et al. 2001], also relies on query-time probes of all collections, but does not require server cooperation. Having received a user's query, TRD-CS forwards this query to all servers; each document in each result set is then downloaded and scored according to a ranking formula which rewards term occurrences early in the document and prefers the first two query terms. Similarly to ReDDE and to the probe queries technique (Section 7.1.12 following), TRD-CS credits servers which provided the top-ranked returned documents. In experiments by Rasolofo et al., over collections formed from TREC ad hoc data, precision achieved by TRD-CS and an associated merging strategy was significantly higher than that achieved by CORI and its associated merging strategy [Callan et al. 1995] and close to the precision of an optimum selection. These experiments did not however examine selection performance alone.

Query-time probes are not considered in the experiments below. Since both methods forward a probe query to all servers, for all user queries, these probe queries must be cheap; this is not the case with real-world servers. Lightweight probes also require the support of servers, since they rely on information that is not available via a simple query interface.

7.1.12 Probe queries

Craswell et al. [2000] observe that there is little point selecting a collection with useful documents if the server responsible for that collection performs poorly and cannot return them. Their "probe queries" technique uses documents sampled from each server to estimate effectiveness.

A pre-defined probe query is issued to each server, and each document in each result set is downloaded and added to a local collection. All documents in this local collection are then indexed and ranked against the probe query. On the assumption that highly-ranked documents are useful, and that effective servers will be able to return many useful documents, servers are given credit for each of their documents in the top 20 of this central ranking. (The figure 20 appears to have been chosen arbitarily.) Overall effectiveness is calculated as the mean score for each server over a set of probe queries.

Craswell et al. describe combining server effectiveness scores, calculated in advance of any query, with per-query collection scores from CORI. This approach did not however produce significant improvement and probe queries are not evaluated in the experiments in this thesis. The ReDDE and CRCS algorithms, which are evaluated, also make use of a central index of sampled documents.

7.1.13 Decision-theoretic

Starting from assumptions regarding the independence of relevance judgements and the cost of retrieving documents, Fuhr [1999] derives a decision-theoretic formulation of the server selection problem. The aim is to minimise the expected cost of the retrieval process; this cost includes the cost of interrogating each server, fetching each document, and separate costs for presenting relevant and irrelevant documents. If the performance of each server is known, as a precision-recall curve, an expected optimal solution can be found.

Early versions of the decision-theoretic framework (DTF) learned a linear approximation of the precision-recall curve from training queries. Later work [Nottelmann and Fuhr 2003; Nottelmann and Fuhr 2004] considered alternative approximations including scoring documents in a central sample index, in a similar manner to ReDDE and CRCS; modelling the distribution of relevant documents by estimating the frequencies of terms in a collection; and estimating the distribution of relevant documents using CORI.

Experiments by Nottelmann and Fuhr, using the UBC-100 testbed of 100 collections from TREC ad hoc data, have compared selection and merging by CORI with that by DTF in several variants. The simplest variant, DTF-rp [Fuhr 1999], assumes a linear precision-recall curve for each server, and performs about the same as CORI. Other more complex variants [Nottelmann and Fuhr 2003; Nottelmann and Fuhr 2004] show greater precision than CORI, but this appears to depend upon the length of queries and is less pronounced with queries which are shorter. Some key elements of the model, including the cost of retrieval, have also not yet been considered.

In each case, each of the DTF algorithms was trained on half the available queries to learn the parameters for, for example, the approximated precision-recall curve. It is not possible at present to do this for personal metasearch collections: while it may be feasible to pool judgements for shared collections, such as the Web or large corporate databases, many collections are private and judgements would need to be made by each metasearch user. There is no such set of standard queries and judgements presently available for experimentation in personal metasearch.

7.1.14 Other methods

Several selection methods have been described which rely on classifications, on estimates of past performance or manual relevance judgements, or on characteristics of particular environments.

Profusion [Gauch et al. 1996] offered automatic as well as manual server selection. The automatic selection algorithm looked up each query term in a dictionary of several thousand terms: each term was assigned to one of 13 categories such as “art”, “travel”, or “music”. Each of the six search engines used in Profusion had been assigned a score in each category, based on manual relevance judgements on a test set of queries, and these scores from the appropriate category were used to inform selection. The dictionary could not cover all possible query terms, and a fixed set of three search engines served as a default.

The relevant document distribution (RDD) method Voorhees et al. [1994] also uses manual relevance judgements over training queries, but does not use predefined categories. The training queries most similar to the current query are identified instead and the judgements for these are used to estimate each server’s performance.

Ipeirotis and Gravano [2002] propose a technique ("QProber") for selecting one or more promising collections, if collections are first categorised in a hierarchical scheme such as those in Section 6.1.1. The technique starts by building a language model for each collection and for each category, by propagating terms up the hierarchy; then, starting at the top-level category, selecting between lower-level classifications or collections using for example CORI or bGlOSS. The selection process continues down the hierarchy until there are few enough collections included, or until no sub-category appears promising. Experiments with a testbed of 50 Web-based search engines demonstrated this hierarchical scheme offered greatly improved precision over a flat scheme using either CORI or bGlOSS, a result which may be due to the ability to "forget" a large number of collections which are not mostly about the topic at hand while not discarding models which happen to exclude one or more query terms.

Yang and Zhang [2006] also make use of collections arranged in a topical hierarchy. They propose a two-stage process for selection: first, models of sub-hierarchies are used to score each. Only collections from top-ranked sub-hierarchies are then considered in the second stage, which does the final scoring. Both stages use a scoring technique based on unigram language models and very similar to that of Kullback-Leibler divergence (equation 7.5). Experiments using Reuters data and a collection of Web data showed clear improvements over CORI; however, the two-stage method was allowed a query expansion phase and CORI was not.

The Inquirus2 system [Glover et al. 1999] relied on a classification of queries as well as of servers. At the time of issuing a query, users were asked to specify one of seven categories such as "research papers", "organisational home page", or "topical current events". The query would be forwarded to a fixed set of four to six Web search engines according to this classification.

Server selection in the SavvySearch Web metasearch tool used a "metaindex" which recorded for each ⟨term, server⟩ pair the number of times a user had selected a document to view [Dreilinger and Howe 1997; Howe and Dreilinger 1997]. This was used at selection time to estimate each server's effectiveness on similar queries in the past; scores from this index were combined with estimates of recent performance as measured by the size of result sets and the servers' response times.

As with ReDDE and CRCS, the unified utility maximisation (UUM) technique of Si and Callan [2004] uses a central index of sampled documents, ranked for each query, to estimate the distribution of relevant documents. UUM uses a set of queries and relevance judgements ahead of time to learn a logistic model which maps scores from the central index to probabilities of relevance. At query time, this model is used along with scores from the central index to estimate probabilities of relevance for documents in each collection; this in turn can be used to select servers. Unlike other methods, UUM distinguishes between high-precision and high-recall tasks, and Si and Callan provide an algorithm for each.

The returned utility maximisation (RUM) method [Si and Callan 2005] builds on UUM but like Craswell et al. [2000] takes into account the effectiveness of each server as well as the likely relevance of documents in each collection. Evaluations by Si and Callan using a number of testbeds demonstrated that both UUM and RUM perform as well as, or better than, ReDDE, and that RUM performs better still in situations where more servers are ineffective.

Finally, several selection methods have been proposed which rely on characteristics of particular collections, or particular metasearch environments. COSCO [Hernandez and Kambhampati 2005] and ROSCO [Chokshi et al. 2006] use estimates of overlap between collections to inform selection; both algorithms attempt to select collections which can contribute new documents, rather than duplicates of those already seen. The HARP and AWSUM algorithms for hybrid Web metasearch [Hawking and Thomas 2005] use anchortext found in the locally-indexed portion as surrogate documents, and rank these surrogates to rank collections.

None of the methods of this subsection are considered in the experiments of Section 7.2, since in this instance there are no classifications of collections; overlap between collections is minimal; anchortext is available in only one collection; and manual feedback or judgements are not available (although this is considered further in Section 8.2.1).

A small number of systems have included manual server selection, including "content labels" [Sheldon et al. 1994] and Profusion [Gauch et al. 1996]. Dolin et al. [1997] and Meng et al. [2002] also proposed manual selection, after first classifying servers according to collection content. These manual alternatives are not considered in this thesis, although the PIS tool used in Chapter 9 offers manual selection as part of its query syntax.

7.1.15 Evaluations

Each of the methods described above have been evaluated by their authors, using testbeds of varying type. A number of studies have also compared two or more methods directly, on the same test data.

In a series of papers [French et al. 1998; French et al. 1999; Powell and French 2003], French, Powell et al. have developed a set of testbeds based on the TREC ad hoc collections and used it to evaluate CVV, vGlOSS, and CORI for server selection. The testbeds developed are based on data available at TREC-4 in 1995, and include SYM--236 (a division into 236 collections, arranged by source and publication date); UBC-100 (100 collections, arranged to have approximately equal byte counts); and UDC-236 (236 collections, arranged to have approximately equal numbers of documents). Other testbeds included up to 921 databases, but again arranged to keep size approximately uniform. Queries in all these testbeds were based on TREC ad hoc topics and divided into "short" and "long" forms, with mean lengths of 3.5 and 21 terms respectively, and in one investigation [French et al. 1999] the queries were first fed through an automatic expansion process.

Hyusein and Carthy [2004] explored the effect of increased topicality on the CORI, CVV, and vGlOSS methods. 50 collections from the WT10g testbed were pre-processed and formed into 20 clusters apiece; each of these clusters was then scored for each query, and the collection which provided the top-scoring cluster was selected. From this increased topicality, Hyusein and Carthy saw improvements in both precision and recall. They did not however report how the 50 collections were generated; real collections may already be more focussed than the 50 they used. This method also requirs support for document clustering at each server. A baseline run, with the same 50 collections and no clustering, saw CVV and CORI outperform vGlOSS. This is also seen in the experiments in this chapter.

Relatively little work has been done to confirm the applicability to other environments of server selection results obtained on TREC ad hoc data. Rasolofo et al. [2001] compared CORI and their own method for server selection and for results merging using an eight-way division of the TREC WT10g collection of Web data. The authors expressly considered the nature of Web queries and used either a very short form of the TREC topics (two terms on average) or queries garnered from logs of the Excite search engine (2.4 terms on average). Craswell et al. [2000] evaluated CORI, vGlOSS, and CVV in a testbed based on the 2GB, 951 server WT2g crawl of the Web. They concluded that CORI, and a modified version of the CORI algorithm, performed reasonably effectively at the server selection task. Similar experiments by Hawking and Thomas [2005], using the 18GB .GOV test collection divided according to the presence of local search engines, compared CORI and extensions, ReDDE, Kullback-Leibler divergence and extension, AWSUM, and HARP. ReDDE and extended Kullback-Liebler divergence were the best-performing of the established methods, although performance varied according to the type of task.

7.2 Selection experiments

In general, the performance of the selection techniques above has been evaluated using TREC ad hoc resources. The experiments reported below consider the likely performance of these techniques in a personal metasearch tool, and consider questions similar to those in Chapter 5:

1. How does selection performance compare across methods, and across queries?

2. How does selection performance vary, given samples and size estimates of different quality?

3. Which, if any, of the methods is appropriate for personal metasearch?

Twelve selection methods are considered in these experiments: bGlOSS, vGlOSS, CORI and two extensions, lexicon inspection using algorithm “I”, CVV, Kullback-Leibler divergence and one extension, ReDDE, and the linear and exponential variants of CRCS. Of these, CORI and extensions, CVV, vGlOSS, Kullback-Leibler divergence and extension, and ReDDE have been well-tested in other scenarios, but never all at once and never on a personal metasearch testbed. bGlOSS, lexicon inspection, and CRCS are less well-studied.

Two further methods are included as baselines. The “random” method ranks each server randomly for each query, without using any information about each server.

The "size" method simply ranks servers by their estimated size, largest to smallest. Neither of these baselines make use of the query text.

The collections used are again the six of Table 4.2 on p. 40, which are representative of a range of document types and sizes:

- The "calendar" collection contains 1049 documents (appointments) from a calendar application. Documents are typically short sentence fragments.
- The "zsh-list" and "procmail" collections are the archives of public mailing lists discussing narrow technical topics, and have around 9000 and 24,000 documents respectively.
- The "email" collection includes around 25,000 documents from a personal email archive with much broader topics.
- "WSJ" collects several years of articles (around 99,000) from the Wall Street Journal.
- ".GOV", the largest collection used here, is a 1.2 million page crawl of Web hosts in the `.gov` domain.

7.2.1 Queries

No standard query sets have been collected for personal metasearch, so queries for these experiments were created with the intention of representing the variety of topics covered in personal metasearch collections. Past work has often used long queries; for example, Xu and Croft [1999] report a series of experiments with a mean 34.5 terms per query. This is evidently much larger than the 1.7–2.6 terms typical of queries to web search services [Beitzel et al. 2004; Jansen et al. 2000; Silverstein et al. 1999; Spink et al. 2002; Zhang and Moffat 2006], and the queries used here were accordingly kept short.

Queries were generated in a number of ways and formed six sets of 20, each based on one of the six collections in the testbed.

- Subject and location fields were extracted from calendar entries, stopwords were removed, and 20 of these reduced fields selected at random made up the first set of queries. Queries had a mean length of 1.95 terms, with a standard deviation of 0.59 terms.

  PADRE [Hawking et al. 2000] was used to retrieve the top five documents from each collection for each of these 20 queries, and relevance judgements were conducted manually for each returned document. (This is a pooling technique in the manner of Harman [2005], with depth five.) The vast majority of relevant documents came from the calendar collection, with a number also from the email collection; since these two collections overlap somewhat in subject matter, and many pieces of email also contain dates and places, this is expected.

- A similar process was used to construct queries from subject lines in the email collection. Queries had a mean length of two terms and a standard deviation of 0.55 terms, and top results were manually judged. Most relevant documents were drawn from the email collection, with some from the calendar collection.
- The procmail and zsh-list collections represent mailing lists on narrow technical subjects. The same process was used to construct queries based upon these two collections, but there was no topical overlap; relevant documents from each set of queries came only from the original collection.

  As may be expected from more focussed sets of documents, these two sets of queries were longer on average (mean 3.90 terms for procmail, 3.30 for zsh-list; standard deviation 3.47 terms and 1.23 terms).
- The "topic" field of twenty topics from the TREC ad hoc track, all of which had at least one relevant document in the WSJ collection, were used as a fifth set of queries. Relevance judgements were made over the top results from the calendar, email, procmail, zsh-list, and .GOV collections; relevance judgements from TREC were used for documents from the WSJ collection. These queries were rather long (mean 4.20 terms, standard deviation 2.52 terms).
- The "topic" field of twenty topics from the TREC Web track made up the final set of queries, and relevance judgements were made in a similar way to the previous set. This set had a mean length of 3.30 terms and standard deviation of 1.23 terms.

Over all 120 queries, the number of relevant documents per collection varied from 45 for the calendar collection to 101 for WSJ.

Queries in other test collections have typically been generated manually. The first four sets of queries used in these experiments were instead created automatically from identifying fields; the intention was to mimic the terms a user might type if they were familiar with the contents of each collection. It is not clear how well these automatically-generated queries match the queries metasearch users would really type, but nor is this clear for the manually-generated queries of the last two sets. Experiments with test users, reported in Chapter 9, suggest that this test collection can correctly identify high- and low-performing selection methods.

7.2.2 Models and size estimates

The selection techniques used in these experiments rely on language models and size estimates of the collections involved (Table 7.1). As in Chapters 5 and 6, this data is drawn from three sources.

As a baseline, models were built using all documents in each collection, and "estimates" of collection size were entirely accurate. Although not possible in practice, this provides a best case for comparison. A second set of models were built from 300 documents from each collection sampled by the multiple queries sampler of Section 4.3;

size estimates for this set were produced by the multiple capture-recapture method [Shokouhi et al. 2006]. These represent the best-performing algorithms in earlier work. A final set of models were built from 300 documents from each collection returned by the query-based sampler of Section 4.2.2 [Callan et al. 1999], with size estimates from the sample-resample algorithm of Section 5.1.4 [Si and Callan 2003b]; although not as accurate as other techniques, models and size estimates built this way have been used in previous selection experiments [Hawking and Thomas 2005; Shokouhi et al. 2007; Si and Callan 2003a; Si and Callan 2003b]. The choice of 300 documents follows Hawking and Thomas [2005], Si et al. [2002], and Si and Callan [2003a, 2003b]; as seen in Chapter 6, this is also around the point where model quality is no longer improved by adding further documents.

7.2.3 Measures

Previous evaluations of server selection have used two strategies: a whole-system approach, where a tool which includes selection but also result merging is considered as a single system, and an isolated approach which evaluates selection alone.

The whole-system approach has been taken by, for example, Craswell et al. [2000], Powell et al. [2000], Rasolofo et al. [2001], Si et al. [2002], Si and Callan [2003b] (who also evaluate selection alone), Xu and Croft [1999], and Zobel [1997]. In the experiments reported here, however, it is desirable to measure the performance of selection alone; especially since this thesis does not propose any particular merging algorithm.

Callan et al. [1995] measured the effectiveness of the CORI selection algorithm with the mean squared error between CORI's ranking and an ideal ranking. If $\textsc{opt}(c)$ is the rank of c in some optimal ordering (Callan et al. define this optimal ordering in terms of the number of relevant documents in each collection) and $\textsc{est}(c)$ is the rank of c in an estimated ordering, based for example on the scores $s_c(q)$, then the mean squared error is

$$MSE = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} (\textsc{opt}(c) - \textsc{est}(c))^2.$$

The mean squared error has the disadvantage that in case of tied scores $s_c(q)$, both $\textsc{opt}(c)$ and $\textsc{est}(c)$ can vary arbitrarily. For example, if two collections have no relevant documents, it will make no practical difference which is ranked higher in either $\textsc{opt}$ or $\textsc{est}$ but MSE will vary either way. Further, since MSE only considers relative ranks it does not distinguish small errors from large: for example, the error in ranking a collection with 20 relevant documents above one with 100 is the same as if it were ranked above one with 25, despite the former arguably being more significant.

A widely-used alternative without these drawbacks provides a rough analogue to classical precision and recall [Gravano and García-Molina 1995; Lu et al. 1996]. Let each collection c have an associated merit, denoted $\textsc{merit}_c(q)$, which is a measure of how good a choice c is for this query. $\mathcal{R}_n$ is the proportion of this merit captured by the n top-ranked collections:

$$\mathcal{R}_n = \frac{\sum_{i=1\ldots n} \text{MERIT}_{est_i}(q)}{\sum_{i=1\ldots n} \text{MERIT}_{optimal_i}(q)},$$

where est_i is the i'th ranked collection in the estimated ranking and $optimal_i$ is the i'th ranked collection in the optimal ranking.[4] $\mathcal{R}_n$ ranges from 0, meaning there is no merit in the first n collections selected, to 1, meaning the first n collections selected are as good as possible. $\mathcal{R}_n$ has been used by Gravano and García-Molina [1995], Shokouhi [2007], Si and Callan [2003a], and Si and Callan [2003b].

A related measure [French et al. 1998] gives the amount of merit in the first n selections as compared with the *total* merit available over all collections:

$$\hat{\mathcal{R}}_n = \frac{\sum_{i=1\ldots n} \text{MERIT}_{est_i}(q)}{\sum_{c \in \mathcal{C}} \text{MERIT}_c(q)},$$

The two measures are equivalent if n collections or fewer have non-zero merit. $\hat{\mathcal{R}}_n$ has been used by French et al. [1998], French et al. [1999], Hawking and Thistlewaite [1999] (where it is called R_n), and Powell and French [2003]. Similar analogues of precision, $\mathcal{P}_n$ and $\hat{\mathcal{P}}_n$, have also been described and used particularly by Gravano and García-Molina [1995].

$\text{MERIT}_c(q)$, the (real) utility of a collection c, has been defined in a number of ways. Gravano and García-Molina [1995] define MERIT in terms of the total similarity between c and q: the sum, over all documents $d \in \mathcal{D}_c$, of the similarity between d and q. If servers work on a vector-space model, this is an approximation of what an effective server might retrieve in response to a question. This has the effect of including an aspect of server performance, but as well as assuming a vector-space model at the servers it assumes that documents similar to the query are likely to be useful. This definition has also been used by French et al. [1998] and French et al. [1999] (who also use the relevance-based definition discussed below), and Gravano et al. [1999].

An alternative definition of MERIT simply counts the number of relevant documents at each collection, $\text{MERIT}_c(q) = |\text{REL}_c(q)|$.[5] Since the relevant documents in a collection may or may not be returned by a server, this measure is independent of server performance. It has been commonly used for this reason [French et al. 1998; French et al. 1999; Hawking and Thistlewaite 1999; Si and Callan 2003a; Si and Callan 2003b].

Further alternatives for measuring the performance of selection techniques include *success*; α (the proportion of wrong answers) and β (the proportion of correct but partial answers) [Gravano et al. 1994]; and cosine similarity [Yuwono and Lee 1997], but these have been little-used.

The experiments in this chapter use $\mathcal{R}_n$, with merit defined as the number of relevant documents seen while judging. This measure is independent of the final performance of any server, does not assume that only documents similar to the query

[4] Gravano and García-Molina [1995] and most subsequent authors call this measure $\mathcal{R}_n$. Lu et al. [1996] use the notation R_n.

[5] Gravano and García-Molina [1995] refer to this as *Rel_All*.

are relevant, and has a straightforward interpretation as the proportion of possible merit achieved with n collections. For the testbed used here, a random ordering has an expected $\mathcal{R}_n$ of around $n/6$ since there are few queries where more than one collection holds relevant documents. The best ordering, ranking collections according to the number of relevant documents, has a score of $\mathcal{R}_n = 1$ for all n.

The results below do not include a comparison of selective metasearch systems with systems using a central index, as have for example Craswell et al. [2000], Rasolofo et al. [2001], and Xu and Croft [1999]. There are two reasons for this: first, a central index is not a plausible model for personal metasearch, as discussed in Section 3.3.3. Secondly, an optimum ranking function for one collection is likely to be poor for another, and in these cases a central index will perform poorly. This is evident in the results for ReDDE and CRCS in Section 7.3.1 below, where the sample index suffers similarly.

7.2.4 Validation of implementations

Previous work has presented performance figures for CORI and extensions, Kullback--Leibler divergence and extension, and ReDDE on a common testbed [Si and Callan 2003a]. The implementation of each of these six algorithms was validated against this data.

The UBC-100 testbed [Powell et al. 2000], made up of 100 collections from TREC CDs 1–3, was used along with TREC ad hoc queries 51–100. $\mathcal{R}_n$ scores from the present implementations were compared with those of Si and Callan [2003a, Figure 1]; the values obtained were often higher than those previously published and were always within the small margin to be expected given uncontrolled differences in indexing and retrieval.

7.3 Results

Selection experiments based on perfectly accurate models and size estimates demonstrated that Kullback-Leibler divergence performed well, as to a lesser extent did CORI and variants, extended Kullback-Leibler divergence, and bGlOSS. Many methods however are prone to ranking larger collections highly, regardless of their usefulness for any particular query; only Kullback-Leibler divergence and vGlOSS were found to be able to select small collections better than chance.

With less accurate models and size estimates built using the techniques of Chapters 4 to 6, almost all methods were significantly less accurate. CORI, which is less affected by inaccuracies in the model, and Kullback-Leibler divergence, which is less prone to size-based ranking, seem appropriate choices for selection algorithms in real-world tools.

7.3.1 Baseline methods

The first experiment provided a baseline for later comparisons. Each method was run with parameters set as described by the original authors; each method was also allowed a perfectly accurate model built from all documents in the collection, and a perfectly accurate size "estimate". In the case of tied scores $s_c(q)$, collections were ordered arbitrarily.

- vGlOSS used threshold $l = 0$, so any documents at all similar to the query were considered interesting; note that this means the alternatives $\textsc{max}(0, q, c)$ and $\textsc{sum}(0, q, c)$ are equivalent. Following Gravano et al. [1999], weights were assigned to terms in the query ($w_q(t)$) and in the collection ($w_c(t)$) according to term frequency and inverse document frequency:

  $$\begin{aligned} w_q(t) &= \mathrm{tf}_q(t) \log(N_c / \mathrm{df}_c(t)) \\ \text{and } w_c(t) &= \mathrm{tf}_c(t) \log(N_c / \mathrm{df}_c(t)). \end{aligned}$$

  Any query terms which did not appear at all in c were assigned a collection weight $w_c(t)$ of zero. Each of $w_q(t)$ and $w_c(t)$ were normalised by dividing by the Euclidean norm.

- Zobel's algorithm "I" used weights as described in Section 7.1.4. If a query term did not feature in any collection (so $\mathrm{df}_C(t) = 0$), the weight for that term $w(t)$ was assigned zero.

- The CVV algorithm, as described above, will fail if a query term t does not appear in any of the collections (since both $\Pr(t|c)$ and $\Pr(t|\neg c)$ will be zero, and $\mathrm{CV}(t, c)$ will be undefined, for all collections c). In these cases, $\mathrm{CVV}(t)$ is assigned zero and the term is not used to calculate the score of any collection.

- The Kullback-Leibler divergence implementation used here smooths the language model of each collection by mixing in a global language model, as in Equation 7.8. Following Si et al. [2002] and Si and Callan [2003a], the mixing parameter λ was set to 0.5.

- Si and Callan [2003b] suggest that ReDDE is robust with values of r from 0.002 to 0.005 (the top 0.2% to 0.5% of all documents considered relevant); the experiments here used $r = 0.003$. Note that the other parameter k, the score given to a collection for each highly-ranked sample document, is irrelevant due to the normalisation step at the end of the ReDDE algorithm.

  Rather than correct models, ReDDE used a sample index of all documents in all collections. Documents were ranked for each query by PADRE [Hawking et al. 2000], which uses a slight variant of Okapi BM25 [Robertson et al. 1994].

- Following Shokouhi [2007], both CRCS methods give no points to any document not in the top 50 (so $\gamma = 50$). For the exponential scoring of CRCS(e), $\alpha = 1.2$

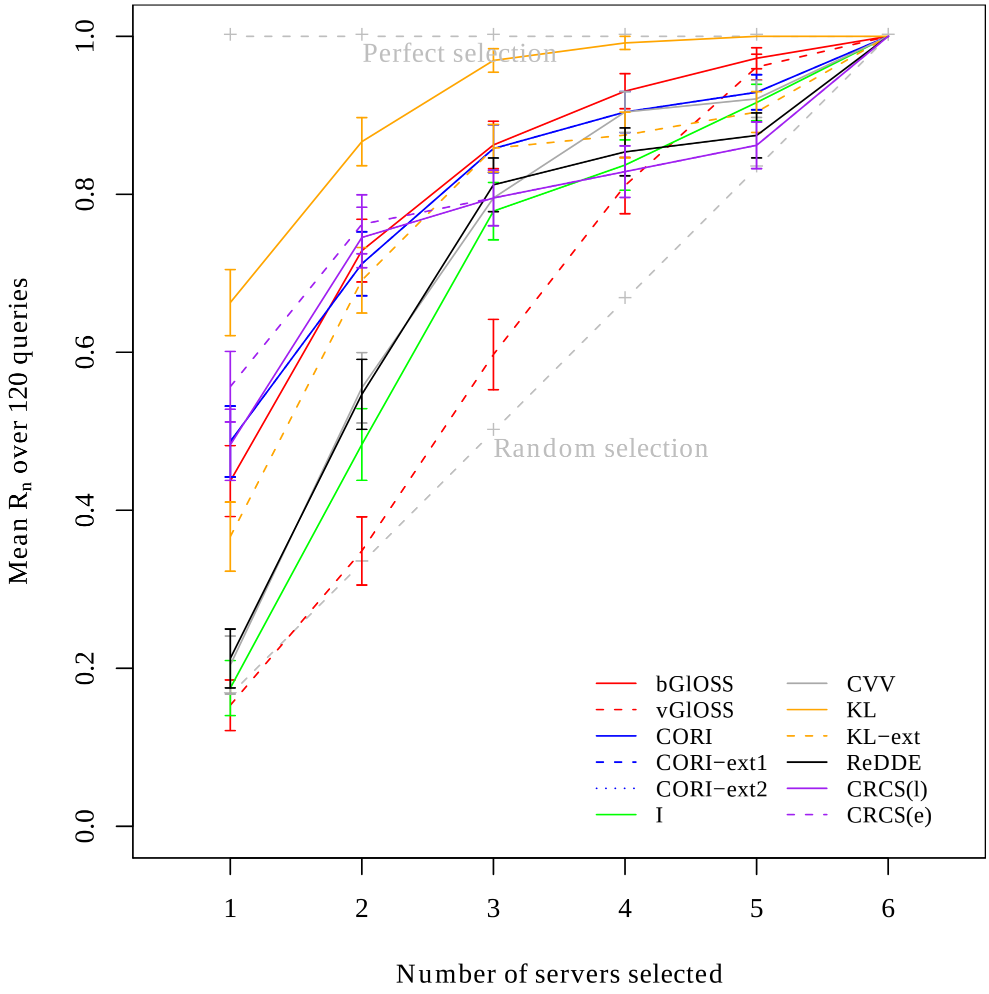


Figure 7.3: Performance of server selection methods, with one to six servers selected; models built from all documents; and correct size estimates. Bars are ± one standard error.

and $\beta = 0.28$. (The figure $\beta = 2.8$ originally published by Shokouhi [2007] was in error.[6])

- The bGlOSS, CORI, and extended CORI algorithms have no tunable parameters.

Figure 7.3 summarises the performance of each method given this baseline data. (Note that lines are interpolated for convenience; n must be in $\{1, 2, 3, 4, 5, 6\}$.) Since models are built from all documents, $N_m = N_c$ and so the three variants of CORI are identical.

In this case the Kullback-Leibler divergence method is clearly the best-performing of the techniques tested; $\mathcal{R}_n$ scores from this technique are significantly higher than those from any other method for $n = 1$ to 5 (one-sided t test, $\alpha = 0.05$). vGlOSS, on the other hand, does relatively poorly and has significantly lower scores than all

[6] Personal communication from Milad Shokouhi.

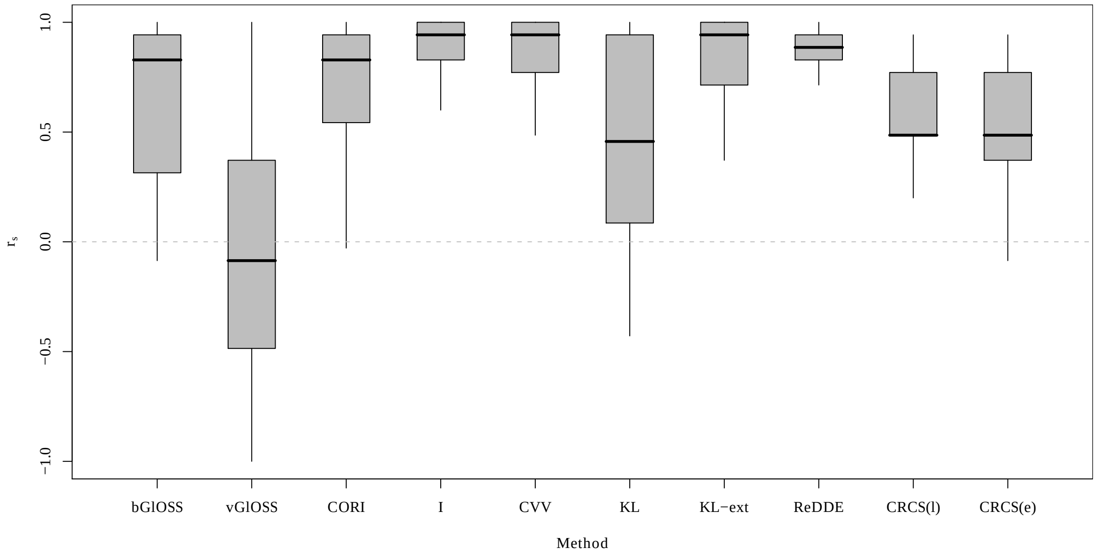


Figure 7.4: Correlation (Spearman's r_s) of collection rankings with rankings based on size alone, over 120 queries. CORI-ext1 and -ext2 are identical to CORI. See notes to Figure 5.2 on p. 63 for plotting conventions.

other methods for $n = 2$ or 3. bGlOSS, CORI and variants, extended Kullback-Leibler divergence, and CRCS variants also perform well.

7.3.2 Correlation with size-based ranking

French et al. [1999] noted a very strong correlation between collection ranks from vGlOSS and ranks based on collection size alone. The survey in Section 3.3 suggests that collection sizes in personal metasearch may range from a few hundred documents (for small email archives, for example) to billions of documents (for the Web or Dialog). If smaller collections contain useful documents, any tendency to select larger collections could result in poor overall performance. A second set of experiments investigated this bias.

Figure 7.4 plots the correlation between rankings based on server scores $s_c(q)$ and those based on collection size N_c, over 120 queries for each selection method. Spearman's rank correlation coefficient r_s is used (see Section 6.2.2) and takes values in $[-1, 1]$. A correlation of -1 indicates collections are ranked smallest to largest, 0 indicates there is no correlation between N_c and $s_c(q)$, and a correlation of 1 indicates that the collections are ordered strictly by size.

It is clear that most methods do in fact correlate highly with a size-based ranking. From Figure 7.4, it seems only vGlOSS and Kullback-Leibler divergence are relatively weakly correlated, which suggests that bGlOSS, CORI and extensions, "I", CVV, extended Kullback-Leibler divergence, and the CRCS variants are all prone to ranking large collections highly regardless of the query. For example, over all 120 test queries, bGlOSS ranks .GOV highest for 80 queries despite there only being 20 queries for which this collection actually contains the largest number of relevant documents. Most other methods are similar, ranking .GOV first for between 58 and 119 of the 120 queries. vGlOSS, which is less prone to ranking by size, ranks .GOV first for only 4 of the 120 queries, and Kullback-Leibler divergence does the same for only 37.

A straightforward examination of the methods demonstrates why large collections are favoured. bGlOSS and extended Kullback-Leibler divergence include an explicit adjustment for N_c, the size of the collection; CORI and extensions, "I", and CVV use either df_c or tf_c, which may be expected to correlate highly with collection size. Although ReDDE and CRCS do not explicitly adjust for collection size in this instance (since $N_m = N_c$), the sample index will be dominated by documents from larger collections. In both vGlOSS and Kullback-Leibler divergence frequency information is normalised on a collection-by-collection basis and larger collections are not favoured.

As an example of this bias, Figures 7.5 presents performance data for each method over that subset of queries for which .GOV (the largest collection, with 1.2M documents) is the best answer, and Figure 7.6 over those for which calendar (the smallest, with 1k documents) is the best answer. Performance on the first set, where success corresponds to choosing the largest collection, is near-perfect for most methods: this is expected, since most methods are highly correlated with a size-based ranking which would rank .GOV first. vGlOSS, however, which is less strongly correlated, is only significantly better than random selection for $n = 5$. The situation is reversed for the second subset of queries, where the task is to choose the smallest collection; only Kullback-Leibler divergence and vGlOSS, the two methods least prone to size-based ranking, are significantly better than random selection for any n. Other methods tend to rank the small calendar collection low despite it being the best choice in these instances.

French et al. [1999] report much higher correlation between vGlOSS and a size-based ranking on their 236-collection testbed (mean $r_s = 0.97$) than is seen here (mean $r_s = -0.06$). This may be due to differences in the distribution of relevant documents: French et al. observe that the larger collections in their testbed tended to have more relevant documents, which would lead them to be highly ranked by any effective selection technique, and a relevance-based ranking of collections in their testbed correlates moderately well with a size-based ranking ($r_s = 0.54$ on average, and $r_s \geq 0$ for all queries). The distribution of relevant documents in the testbed used here, however, is much more uniform (Section 7.2.1). A difference in calculating the weights $w_c(t)$ and $w_q(t)$, for example by normalising across all collections rather than across a single collection at a time, could also give rise to this disagreement. Data from French et al. and these experiments are in broad agreement however on the correlation between CORI and a size-based ranking, and this correlation was also mooted by D'Souza et al. [2004a].

7.3.3 Samples and size estimates

The experiments described in the previous subsection use perfect models, or complete samples, and exact size "estimates". This provides a baseline, but is of course imprac-

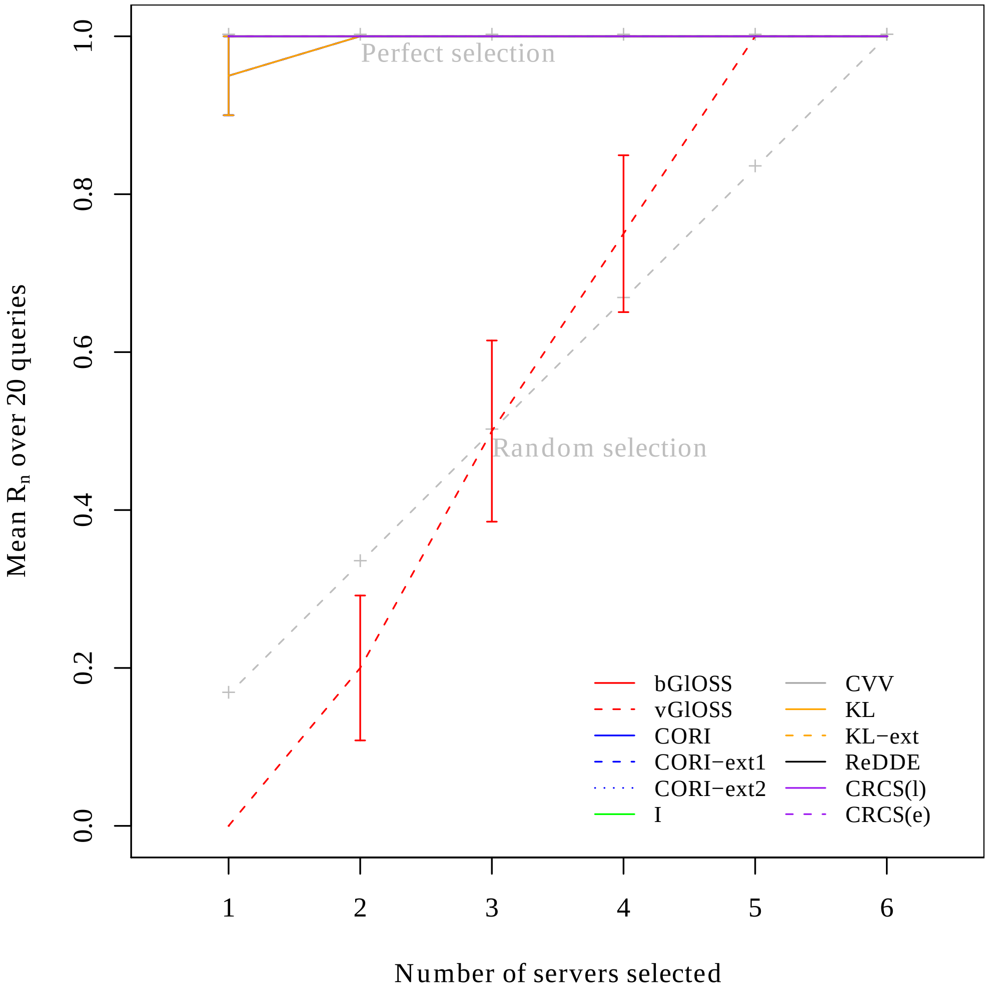


Figure 7.5: Queries for which .GOV (1.2M documents) is the best collection. bGlOSS, "I", CVV, extended Kullback-Leibler divergence, ReDDE, and CRCS all perform perfectly, as do CORI and extensions and Kullback-Leibler divergence for $n > 1$. Models built from all documents; and correct size estimates. Bars are ± one standard error.

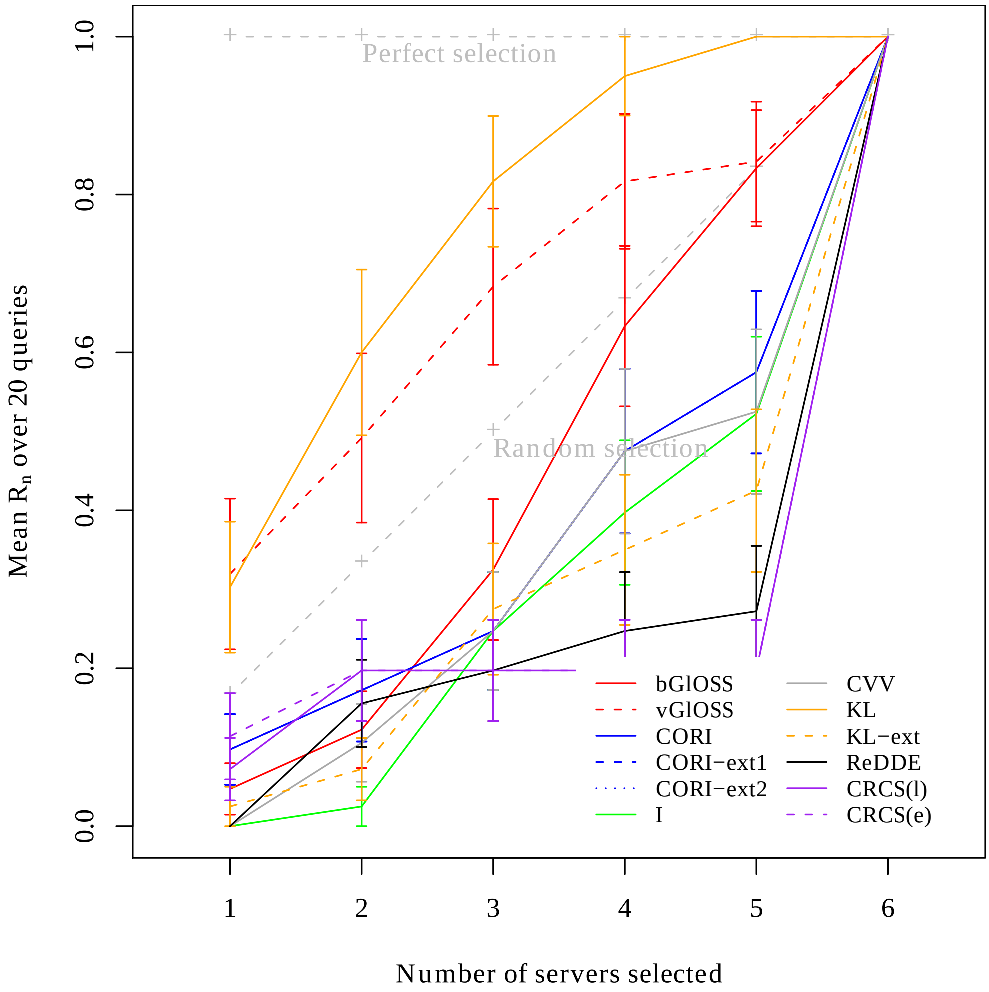


Figure 7.6: Queries for which calendar (1k documents) is the best collection. Most methods perform worse than random selection. Models built from all documents; and correct size estimates. Bars are ± one standard error.

tical in real metasearch applications; a third set of experiments therefore considered selection performance with models built from document samples and collection sizes estimated using the techniques of Chapter 5.

Two sets of samples and size estimates were used. The first set used samples of 300 documents per collection generated by the multiple queries sampler of Section 4.3, and size estimates from multiple capture-recapture (Section 5.1.2; Shokouhi et al. [2006]); these were the best-performing sampler and size estimator in previous experiments. (As noted in Section 6.3, only 200 documents could be sampled from the calendar collection.) The second used samples of 300 documents generated by query-based sampling (Section 4.2.2; Callan et al. [1999]) and size estimates from (single) capture-recapture (Section 5.1.1; Liu et al. [2001]), which have been commonly used in previous work.

The choice of a 300-document sample size follows previous work [Callan et al. 1999; Hawking and Thomas 2005; Nottelmann and Fuhr 2003; Si et al. 2002; Si and Callan 2003a; Si and Callan 2003b] but is essentially arbitrary. A more sophisticated approach may attempt to sample documents until, for example, the learned language model appears stable [Baillie et al. 2006a]; on the measures of Chapter 6, and on the testbed used here, this appears likely to be at about 300–400 documents per collection.

Selection methods needed only minor adaptations to operate with models built from sampled documents.

- For bGlOSS, document frequencies were estimated from the model and scaled according to the ratio $\hat{N}_c/N_m$.

- Document and term frequencies for vGlOSS were estimated similarly. As before, any query terms which did not appear at all in c were assigned a collection weight $w_c(t)$ of zero.

- The two extended versions of CORI differed from unmodified CORI, as the language models in these experiments were built from a subset of each collection and $\hat{N}_c \neq N_m$. The extensions explicitly adjust for this, so no further modifications were made.

- Zobel's algorithm "I" again used weights as described in Section 7.1.4. If a query term did not feature in any collection (so $\mathrm{df}_C(t) = 0$), the weight for that term $w(t)$ was assigned zero. Frequencies were estimated from the model and scaled as for bGlOSS.

- CVV used document frequencies estimated from the model.

- The Kullback-Leibler divergence implementation again smoothed models with the global model using $\lambda = 0.5$. Document frequencies were estimated.

- ReDDE explicitly adjusts for differences in model and collection size, so no further modifications were made. As before, $r = 0.003$.

- The two CRCS variants also explicitly adjust for differences in model size, so no modifications were necessary.

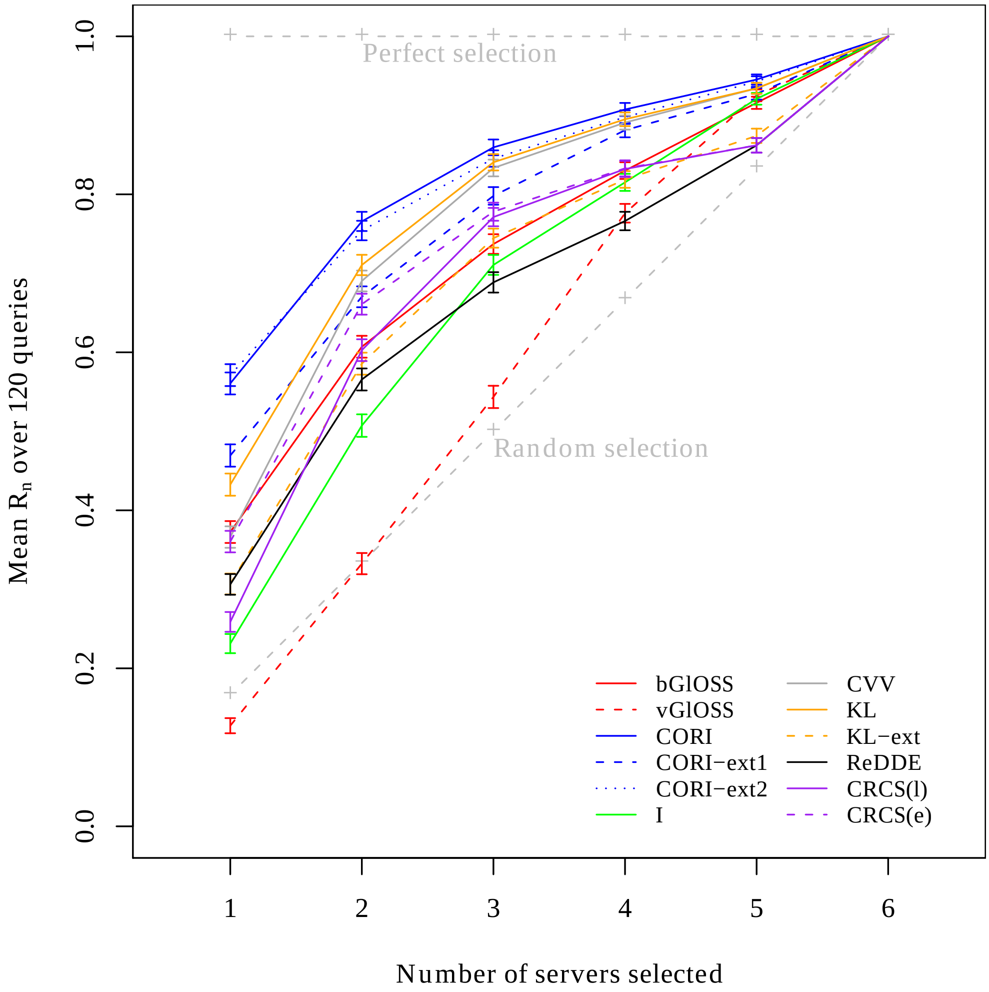


Figure 7.7: Performance of server selection methods, with one to six servers selected; models built from documents sampled with multiple queries; and collection sizes estimated with multiple capture-recapture. Bars are ± one standard error.

Figures 7.7 and 7.8 summarise the performance of each method with these two sets of samples. Most methods perform worse with these lower-quality models than with the perfect models of Section 7.3.1 for at least some values of n. Kullback-Leibler divergence was previously the best-performing method; with models built from 300--document samples from the multiple queries sampler and size estimates from multiple capture-recapture, it is significantly worse for $n = 1$ to 5 (one-tailed Wilcoxon test, $\alpha = 0.05$ [Sheskin 2004]). bGlOSS, another method which performed well with correct data, is significantly worse for $n = 1$ and 3 to 5; extended Kullback-Leibler divergence is significantly worse for $n = 1$ to 5; and both CRCS variants are significantly worse for $n = 1$ and 2. CORI, which performed relatively well in earlier experiments, is not significantly worse for any n.

Similar trends hold for the second set of data, drawn from the query-based sampler and (single) capture-recapture. Again Kullback-Leibler divergence, its extension,

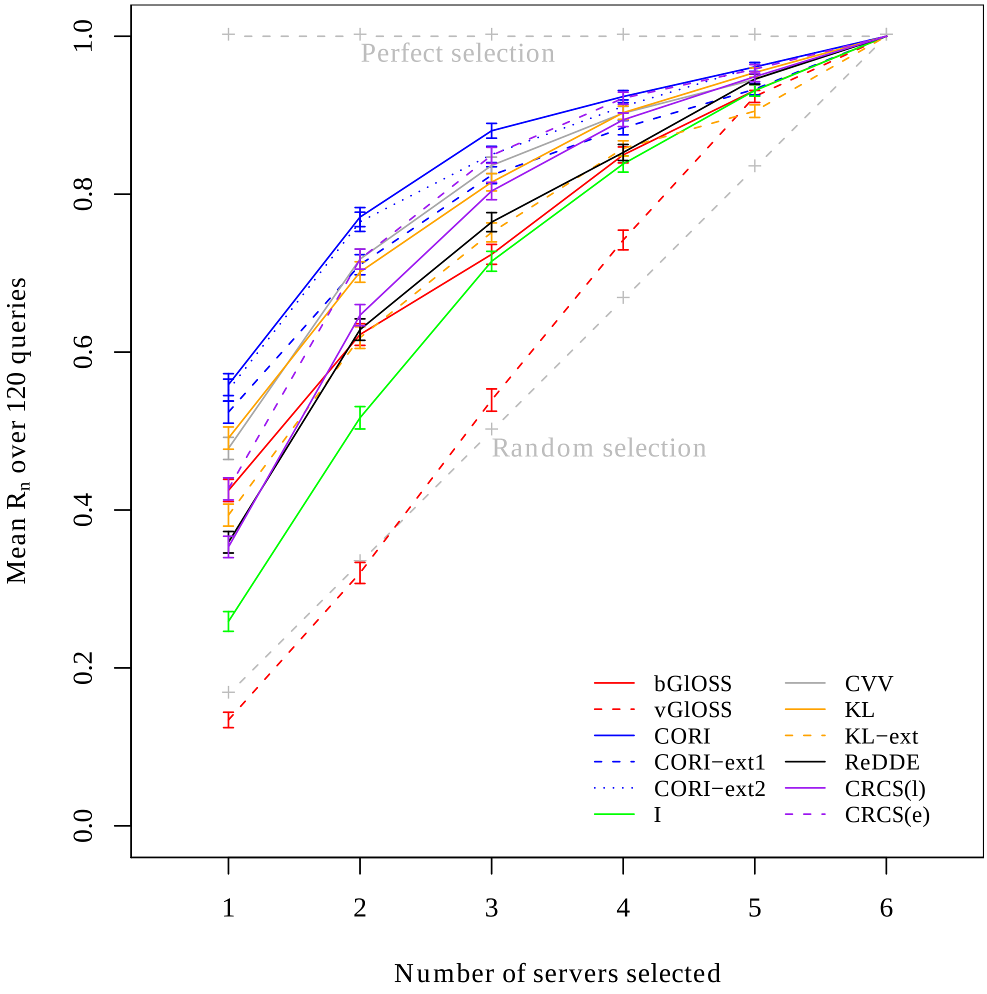


Figure 7.8: Performance of server selection methods, with one to six servers selected; models built from documents sampled with query-based sampling; and collection sizes estimated with capture-recapture. Bars are ± one standard error.

bGlOSS, and CRCS are significantly worse for several values of n, although Kullback-Leibler divergence and the CRCS variants still perform well; CORI is not significantly worse than before at any point, and in fact is significantly better for some n. Taken with the observations on size-based ranking above, this suggests that Kullback-Leibler divergence is an appropriate server selection method for personal metasearch. CORI, CRCS(l), or CRCS(e) may also be appropriate, although they do a poorer job of selecting smaller collections when needed. Choice of selection method may also be influenced by the quality of samples and size estimates.

The extensions to CORI, "CORI-ext1" and "CORI-ext2" in Figures 7.7 and 7.8, have little effect on CORI's overall performance. This is consistent with earlier results from Si and Callan [2003a] and Hawking and Thomas [2005].

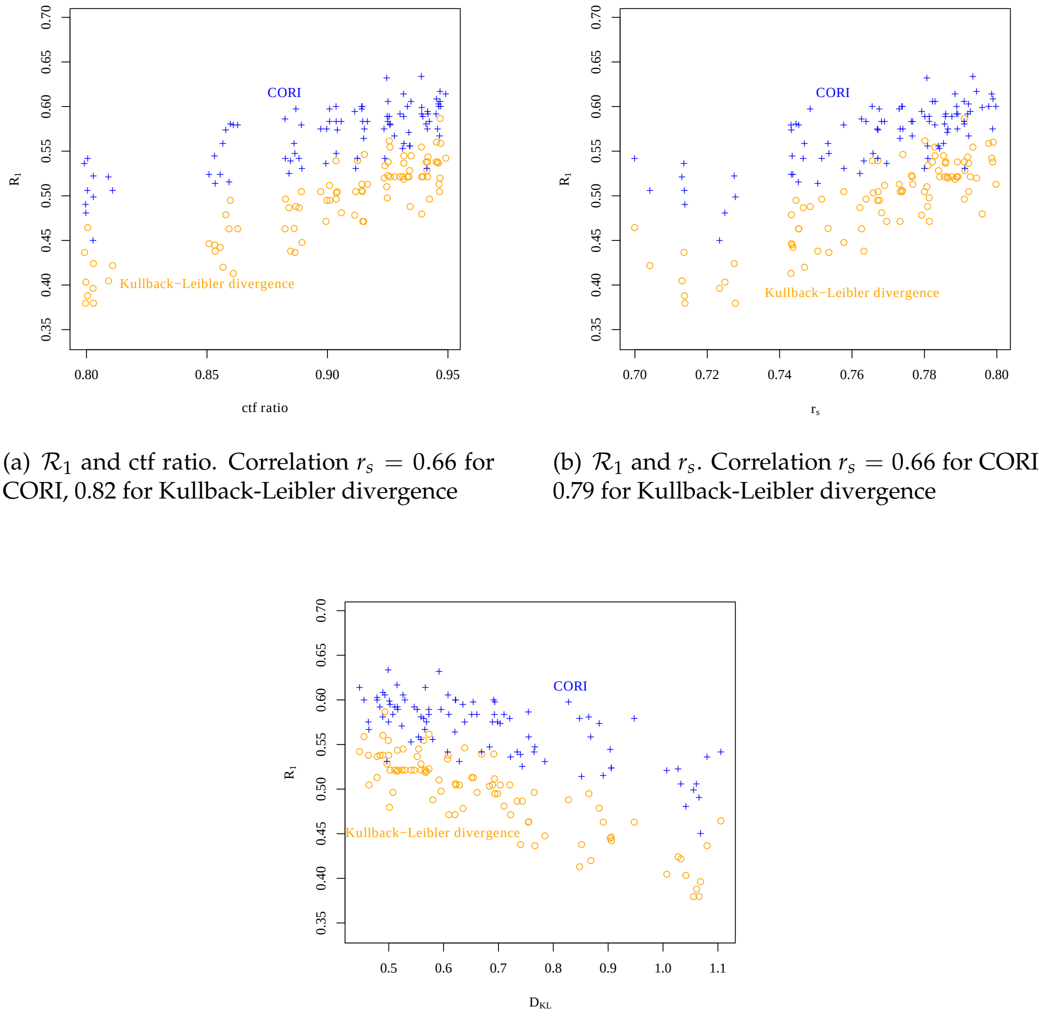
7.3.4 Measures of model quality

Results from earlier experiments suggested that the quality of a language model has a significant effect on the performance of selection algorithms. A final set of experiments investigated this connection, and asked: is there a correlation between the quality of a language model and the performance of selection algorithms? If so, the results from Chapters 4 to 6, concerning techniques to improve the quality of samples and models, should have an impact in improved selection as well as in any other applications.

Ninety sets of samples were generated, ten each of 100–900 documents (in steps of 100) chosen randomly from each collection. Models of each collection were built from each of these ninety sets, and each model's quality was calculated according to the three measures from Chapter 6 (ctf ratio, r_s, and D_{KL}). The mean measure, over each set of six collections, was treated as an indication of the quality of that set of samples. In each case, the mean measure varied from set to set but improved as observed in Section 6.3.1 as more documents were included.

To investigate the correlation between these quality measures and the performance of server selection methods, the CORI and Kullback-Leibler divergence algorithms — two of the most promising selection algorithms from earlier experiments — were run with each set of models over the same 120 queries as before. (Size "estimates" in these runs were correct, to isolate the effect of model quality.) Results for each run are illustrated in Figure 7.9, which plots the performance of the algorithm (measured as mean $\mathcal{R}_1$, or the mean recall at one collection selected over all 120 queries) against each of the three quality measures. Recall that for ctf ratio and r_s, 1 is best and 0 is poor; for D_{KL}, 0 is best and higher values are poorer.

All three measures correlate highly with $\mathcal{R}_1$ for both CORI and Kullback-Leibler divergence, with absolute coefficients of correlation (Spearman's r_s) of between 0.64 and 0.82 ($p \ll 0.05$ in each case). This suggests that selection performance, at least for the CORI and Kullback-Leibler algorithms, will increase if language models are improved and degrade if they are made worse. Further, all three measures discussed in Section 6.2.2 are useful; all three are good predictors of how well selection algorithms will fare.



(a) $\mathcal{R}_1$ and ctf ratio. Correlation $r_s = 0.66$ for CORI, 0.82 for Kullback-Leibler divergence

(b) $\mathcal{R}_1$ and r_s. Correlation $r_s = 0.66$ for CORI, 0.79 for Kullback-Leibler divergence

(c) $\mathcal{R}_1$ and D_{KL}. Correlation $r_s = -0.64$ for CORI, -0.82 for Kullback-Leibler divergence

Figure 7.9: Correlation between measures of model quality and selection performance at one collection selected, for CORI and Kullback-Leibler divergence. Each plotted point is the mean $\mathcal{R}_1$ over 120 queries. $p \ll 0.05$ in each instance.

7.4 Conclusions

Most metasearch tools, including personal metasearch tools, need a query-time process of server selection to identify which servers may be useful. This process can reduce costs, especially if servers charge for access, can improve reliability and efficiency, and if done well enough can improve results even compared to a single index. Many selection methods have been suggested.

Overall, the CORI algorithm is promising for personal metasearch, and is robust to poorer quality models; Kullback-Leibler divergence is also robust and is much more likely to select smaller collections when appropriate.

Experiments in this chapter have shown several trends. Correcting for collection size in a selection technique substantially reduces performance when smaller collections are required, and such corrections are unlikely to improve performance overall unless most queries are in fact answered well by larger collections. The quality of language models and collection size estimates used in selection tasks has been shown to be important: almost all methods suffered a drop in performance when given models inferred from sampled documents. Further, for the Kullback-Leibler divergence and CORI techniques there is a strong correlation between measures of model quality (ctf ratio, r_s, and D_{KL}) and early performance. This suggests that the techniques of Chapters 4 to 6 will be of importance to any real-world tool using any of the most promising selection techniques.

With techniques for sampling, size estimation, modelling, and selection it is possible to build a personal metasearch tool. Questions remain, however: how can such a tool be evaluated? How can choices of algorithm be compared in practice? Work so far in this thesis has used a static testbed and artificial queries, but this may not be the best choice for personal metasearch. The next chapter considers questions of system evaluation.

Chapter 8

Evaluating personal metasearch

The experiments described in earlier chapters follow a standard approach, where a fixed collection and "canned" information needs and judgements substitute for the full system context and user needs. This is a straightforward and convenient technique, and one which allows components to be evaluated in isolation; however, it is not a good match for the sort of personal metasearch system proposed in this thesis. This chapter argues for an alternative method, which is then used for a user evaluation of server selection algorithms in Chapter 9.[1]

There are a number of established approaches to evaluating information retrieval systems, and these are discussed in Section 8.1. Section 8.2 proposes a randomised comparison tool in which alternative result sets obtained in the actual context of a real search are presented in side-by-side panels and the searcher is asked for online comparative judgments. This tool has been implemented in two different forms and has several mechanisms available for recording information about searcher behaviour. Limitations are also discussed.

Section 8.3 reports experiments to validate the use of the tool in Web searching and to test for inherent biases. This section also investigates the value of implicit feedback as a predictor of explicit judgments.

The method also has application to other information retrieval problems. Section 8.4 describes a case study, using the technique to evaluate whole-of-Web search, and Section 8.5 discusses variants of the basic tool and outlines a range of other possible uses.

8.1 Established approaches

There are several broad approaches to the evaluation of information retrieval systems which may be appropriate for the study of personal metasearch: test collections, search log analysis, human experimentation, and naturalistic observation. In this section each approach is described, and its applicability to the task of evaluating personal metasearch is considered.

[1] Some of this chapter has been published as Thomas and Hawking [2006]. The experimental work was original with the author, as was other material reproduced here. "We" in this chapter refers to the author and David Hawking.

8.1.1 Test collection approaches

The approach introduced by Cleverdon [1967] and developed by Spärck Jones and van Rijsbergen [1976] has long been popular; it has notably been taken up by the TREC [Voorhees and Harman 2005], INEX,[2] NTCIR,[3] and CLEF [Peters et al. 2001] conferences. It has been successful in evaluating components of information retrieval systems and has formed the basis for between-component and between-system comparisons. The approach relies on three elements: a standard collection of documents, a large set of information needs (expressed in some useful way) which may be satisfied by documents in the collection, and "complete" lists of relevant documents corresponding to each information need.

If a suitable test collection is available, a new retrieval system can be evaluated by converting the test information needs into queries, processing them against the collection and noting which of the retrieved documents are relevant. From this information, standard measures can be computed: typically these are one or more of precision (P) and recall (R) [Cleverdon and Mills 1963]; the weighted harmonic mean of precision and recall (F) [van Rijsbergen 1979]; precision at n documents retrieved, for various n, R-precision, average precision (AP) and mean average precision (MAP) [Buckley and Voorhees 2005]; mean reciprocal rank (MRR) [Voorhees 1999]; cumulative gain, discounted cumulative gain, and normalised discounted cumulative gain (CG, DCG, and NDCG) [Järvelin and Kekäläinen 2000]; success at n documents retrieved, for various n [Hawking and Craswell 2005]; or Q-measure [Sakai 2004].

In general there are strong advantages of the test collection approach.

- *System evaluations are straightforward once the collection is in place.* Given a test collection and a system which can operate over that collection, it is simple to run the system and calculate document-based performance measures. The cost of each evaluation, once the system is in place, is small enough that many tests can be made in order to (for example) fine-tune parameters.

- *Experiments are reproducable.* With the documents, queries, and judgements fixed, it is possible to re-run evaluation experiments at a later date; or to reproduce others' results given a common system.

- *The collection is reusable.* With a collection in place, it can be reused, unmodified, for experiments at a later date; using a fixed test collection eliminates any problem of documents changing or being removed [Soboroff 2006].

- *Comparisons are robust and reliable.* If a test collection contains a sufficient number of information needs, and enough documents are judged for each (as exemplified by TREC), it can be shown that system comparisons are robust to variations in the particular information needs included [Buckley and Voorhees 2000].

[2] http://inex.is.informatik.uni-duisburg.de/

[3] http://research.nii.ac.jp/ntcir/

There are however several potential problems to be overcome when contemplating the creation or use of a test collection. First, the three elements of the collection and the chosen effectiveness measures must approximate the real situation(s) in which the retrieval system is going to be used well enough to usefully predict outcomes. This can only be done by first understanding real search situations [Cleverdon and Mills 1963; Cooper 1973; Lancaster 1969]. The test collection must also be self-consistent; for example, queries and relevance judgements must be appropriate to the documents [Webber and Moffat 2005].

Second, it is important to minimise any loss of fidelity between the searcher's original information need, the recorded statement of that need (e.g. a TREC topic statement), the query actually processed by the retrieval system, and the need as understood by the person making the judgment. In modern search systems, it is almost always the searcher with the information need who composes the query and who judges the results while the need is still current.

Third, information needs, measures and judgments should accurately reflect the contexts in which searches may be performed. An ideal result set for one searcher may be of little value to another if, for example, they already know the content or are incapable of understanding it.

Fourth, it is difficult to deal with the consequences of independent judgments. In a test collection, each document is judged in isolation from the others, while standard precision and recall measures assume that each additional relevant document retrieved contributes an equal amount to the value of the retrieved set. In reality, the content of two relevant documents may overlap to such an extent that retrieval of the second brings no additional benefit.

Test collections for personal metasearch

Using test collections for personal metasearch also raises problems particular to this application.

Privacy, dynamism, and scale Personal information collections almost always contain private data, such as email folders, which may not be viewable by experimenters. Personal collections may also contain information which is not strictly private but is still restricted in some way, such as proprietary encyclopaedias and content made available by subscription services; these and other collections may have restrictions on copying (Section 2.1.2).

Some collections will also be rapidly evolving and may even change from use to use. A fixed test collection can at best be a snapshot of the documents available at one instant.

Finally, most search will typically cover tens of billions of documents as most searches will include the Web. Distribution of the Web as a document collection is infeasible, and the scale of the Web makes full judgements impossible for practical reasons (although see Section 8.1.1 for alternatives).

Diverse information needs It is likely that future users of personal metasearch systems will use them for a range of purposes including question answering, known-item retrieval, navigational search, service finding, relationship management and informational research. A test collection representing the full range of search types would necessarily be segmented by type since no single measure would reflect searcher satisfaction across the types.

Information needs are typically specified in advance of any tests, and without any tailoring to the capabilities of any one system (although they are tailored to the available collection). These information needs, distilled into possible search terms and a brief discussion of the assumed background and what may count as relevant, may model certain research tasks such as intelligence assessments but it has not been established that they reflect the realities of workplace or personal search.

Typically, the needs are written by researchers or assessors who are not necessarily knowledgeable about the content of the collection. This is appropriate for, e.g., Web search, but for collections such as email archives many users will have a good idea what is covered and what language is used.

A second possible mismatch is between the types of information required to satisfy synthetic and natural needs. For TREC ad hoc and related evaluations, any information on a specified topic is counted as useful; this is not the case for many day-to-day information needs. Even in cases where synthetic needs cover many different types — contact details, summaries of decisions taken, entry points for research, etc. — there is no guarantee these accurately reflect users' real day-to-day experience.

Further, in many cases users may be unable or unwilling to articulate their information need, at least at early stages of the search process [Teevan et al. 2004]. Artificial needs, however, are generally clear and closely specify what sort of information counts as useful.

Set-based and contextual judgements After-the-fact assessments of relevance to a written statement of need are very different to the way a person would judge the results of a search conducted in the course of their usual activities. A quick scan of part of a result set is often enough to judge its utility for the task at hand.

Unlike relevance assessors, searchers very seldom read all the documents retrieved for them by a search engine.[4] A user's goal in performing a search may be satisfied by reading only a single document or without reading any. Instead, for example, they may synthesise an understanding of an area from the result abstracts, or extract a needed telephone number from one of them.

The relevance and utility of a document is sensitive not only to a user's task and information need, but to other aspects of a user's or a document's context. For example, the same information may be contained in two separate documents: the first document is then useful, but the second is not. Similarly, a user may be interested in coverage of a topic from several different angles; documents in the same set may be

[4] Inspection of logs covering ten million queries submitted to the search engine of a busy commercial website showed only around one query per 35,000 leading to clicks on all of the first ten search results.

useful or useless depending on what else is retrieved. In the cases where documents are not accessed and abstracts not examined [Joachims et al. 2005], the presence of relevant documents elsewhere in the list obviously contributes nothing to the judged value of the search. Most techniques which manipulate relevance judgements independently, without reference to other documents, cannot take account of these cases.

Finally, users may not want to go straight to a relevant document even if a system can always find it. Teevan et al. [2004] observed that many users prefer to navigate via a series of known "landmarks", rather than jump straight to a document with relevant information. Most instances discussed were in web search, where presumably people are on less familiar ground than they would be with their own files and archives; however, getting the right answer first time may not be as useful for users as finding appropriate, recognisable, starting points. Studies in the mid 1990s of information searching on desktop computers suggested that such navigation is common even when users are famililar with the data available [Barreau and Nardi 1995; Nardi and Barreau 1997]. This may however have been due to limitations of the search tools then available.

Extensions of test collection methodology

Test collection methods have been extended with an eye to generating collections or judgements with less investigator effort; or to using different metrics or simulated users as alternative evaluation techniques.

Pooling and building collections *Pooling* is commonly used to avoid the very large cost of complete judgments in large test collections. Under a pooling regime, several systems produce result sets for each query and the top n results from each set are added to a pool. (n is called the "depth" of the pool.) Documents in this pool are then judged; documents which are not in the pool, since they were not returned at a high rank by any participating system, are assumed non-relevant [Harman 2005]. Performance measures from pooled judgements are reliable, given sufficient judgements and queries [Buckley and Voorhees 2000; Sanderson and Zobel 2005; Voorhees and Buckley 2002; Zobel 1998].

Pooling is able to dramatically reduce the cost of judging. Measures from pooled judgements are not reliable, however, when judgements do not cover enough of the collection [Buckley and Voorhees 2004], which is almost inevitably the case for large test collections containing millions of documents.

To manage the cost of judging relevance, Soboroff et al. [2001] suggest pooling high-ranked documents from participating systems and marking some of these, at random, as "relevant". System rankings produced this way correlate fairly well with those produced using manual judgements, although high-performing systems are consistently penalised and could not always be distinguished from poorer performers. Since no judgements are needed, this technique could be applied to personal metasearch and private collections to estimate relevance-based metrics. Some prior knowledge of the distribution of relevant documents is however needed to tune the selection process, and with a few exceptions (such as Web search) this information is not available.

A small body of research has aimed to reduce the cost of pool-based methods; in general, more sophisticated techniques for building the pool can offer similar measurements of similar quality after judging fewer documents. Cormack et al. [1998] suggest a "move to front" technique: a system is classified as high-performing as long as its returned documents are judged relevant, and documents returned by high--performing systems are judged first. On a long-run average, systems which return more relevant documents contribute more to the pool and fewer documents need to be judged before sufficient relevant documents are found.

To the same end, Zobel [1998] proposes a method for predicting the number of additional, as yet unjudged, relevant documents that exist for a particular query given counts of the number of relevant documents already seen and the current pool depth. Using this information, it is possible to choose different pool depths for each query, and for a fixed amount of judging effort find a larger number of relevant documents. With more relevant documents found, Zobel suggests pool-based results can be more reliable.

Building on work by Cormack et al. [1998] and Soboroff and Robertson [2003], experiments by Sanderson and Joho [2004] demonstrated satisfactory system rankings if a "pool" was generated from results returned by only a single system. Using relevance feedback to propose documents to judge generated rankings similar to official TREC results with around 10% the effort. Adapting Cormack et al.'s "interactive searching and judging" and again using a single-system "pool", Sanderson and Joho were able to produce rankings acceptably close to TREC results in about half of their experiments.

Rather than increase the number of relevant documents found, Carterette et al. [2006] suggest judging those documents most likely to influence the difference in MAP between two systems. With an algorithm to return these documents, they were able to use a small number of judgements to replicate rankings derived from relatively large pools. The documents chosen for judging in this technique do however depend upon the particular results returned by each system, so the collected judgements are not useful for comparing other systems at a later date.

These techniques attempt to reduce judging effort by building pools in a more sophisticated manner. In contrast, the TREC Web Track [Hawking and Craswell 2005] sought reusability by exploring search tasks for which there was only one right answer defined in advance (homepage finding and named page finding) or only a well-defined small set of right answers. A similar technique was used in the TREC-5 Confusion Track and the TREC-6 Spoken-Document Retrieval Track [Voorhees and Garofolo 2005].

Alternative measures Buckley and Voorhees [2004] attempt to solve the problem of building re-usable large collections with the *bpref* measure, in which unjudged documents are ignored rather than assumed non-relevant. *Bpref* correlates well with MAP when full judgements are available, but system rankings using *bpref* are more stable when document sets change or when fewer judgements are available.

In a similar vein, Yilmaz and Aslam [2006] introduce three new measures which like *bpref* are robust to incomplete or imperfect judgements. Induced AP (*indAP*), subcollection AP (*subAP*), and inferred AP (*infAP*) again correlate well with MAP when full rankings are available (*subAP* and *infAP* are equivalent to AP when all documents are judged), but are still more robust than *bpref* even when very small numbers of judgements are available.

Simulated users Several researchers have used simulated users, some quite sophisticated, in conjunction with TREC judgements and collections to evaluate IR techniques [Harman 1998; Magennis and van Rijsbergen 1997; Ruthven 2003; White et al. 2005]. Although a useful low-cost alternative to full human experimentation, the reliance on fixed collections and judgements for evaluation and for input to the user simulations means these techniques have limitations similar to traditional batch evaluation.

8.1.2 Search log analysis

An alternative to explicit judgements is to consider user selection of a document (a click) as an indication of expected utility, and use clickthrough logging to evaluate systems. This technique is appealing in that it does not entail any extra burden on users, and it can capture judgements for a variety of information needs. Two forms of bias however need addressing.

"Trust bias" leads to more clicks on high-ranked documents, regardless of the documents' utility, as a result of users' faith in IR systems. Observations from enterprise search logs suggest that more than half of "wrong" clicks — that is, clicks on documents not previously judged as useful — are on the top-ranked result.[5] "Quality bias" is the result of users being given a set of documents to choose amongst, not a single document at a time. A click on a document should be interpreted not as a vote for that document's relevance, but rather for its being more relevant than (at least some) others in the set [Joachims et al. 2005].

There are two further weaknesses in clickthrough data which are relevant to the metasearch setting. First, although clickthrough data is known to correlate with utility in the web [Fox 2003; Joachims et al. 2005], it has not been established that this is the case for other types of data.

Second, many queries have no associated clicks. (From observations of a commercial search engine log, this may be the case for a majority of queries.[6]) There could be several possible reasons: the summaries provided may answer the users' information need without any further reading ("brilliant success"); the summaries provided may make it clear the retrieval system, or collection(s), cannot answer the need ("abject failure"); or characteristics of the delivery mechanism may be responsible, such as web

[5] Personal communication from David Hawking.

[6] Personal communication from Tom Rowlands.

page reloads or clicking more than once on the "search" button ("query bounce"). It seems important to at least distinguish brilliant success from abject failure, but this is not possible with clickthrough data alone.

To overcome trust and quality biases, Joachims [2002a] suggests interleaving results from two systems, and using the number of results selected from each as an indication of that system's quality. Tests with two presumed good systems (commercial Web searches) and a presumed poor-quality system (a commercial Web search but with reversed ranking) suggest this method can accurately predict user preference. This method however is restricted to comparing ranked lists; it is not possible to consider sets of documents, for example for coverage or duplication, or lists arranged in some other manner (for example by source).

Finally, search logs are generally maintained by individual search engines. Given two logs from two systems, each representing different user populations and information needs, it is not clear how a reliable comparison could be made.

8.1.3 Human experimentation in the lab

A further method of evaluation involves observing test users in a laboratory setting, conducting searches in response to a simulated information need.

Interactive evaluation

The TREC Interactive Track [Hersh and Over 2001] has attempted to isolate the effect of different IR systems with a sophisticated design which controls for differences in searcher and topic, and for presentation order. This method is independent of any particular questions and collection, but the same questions and collection must be used for each participant in any given experiment. Problems are also caused by variation between subjects and topics, and by effects due to presentation order.

Borlund and Ingwersen suggest an evaluation framework, which they call the IIR (interactive IR) evaluation model [Borlund and Ingwersen 1998; Borlund 2003]. This model makes use of "simulated work task situations", a narrative which describes an information need and its context, and which includes a sample request. By describing a work situation, and not a topic, the aim is to allow individual interpretation of which documents and which system features are useful. Alternative metrics, such as relative relevance (RR) and ranked half-life (RHL), are based on these judgements.

By including users in evaluation, the IIR model is designed to produce realistic judgements; by using the same work situations, these judgements are assumed to be comparable. This latter assumption only holds however in those cases when the same collection is available to each user, and when the task is understood in the same way by each user. If either of these criteria are violated, it is not clear to what extent user judgements would be stable from system to system or experiment to experiment, which makes the calculated metrics less useful long-term. Experiments with the IIR model generally need to control for order, learning, or fatigue effects as well as individual differences.

Borlund and Ingwersen [1998] also distinguish several types of "relevance": algorithmic relevance (relevance of results to request); intellectual topicality (relevance of results to information need); and situational relevance (relevance of results to perceived task). In a series of experiments, the same tasks and retrieved documents were judged for all three types of relevance. Many documents which were highly relevant on the algorithmic measure were still only marginally relevant on intellectual topical or situational measures, which suggests that measures such as precision may not be a good indicator of final user satisfaction.

Validating test collection metrics

A number of lab experiments have investigated the connection between system metrics derived from test collections and user satisfaction or performance. Hersh et al. [2000] found that tuning a system to perform well on a test collection (TREC ad hoc) did not significantly improve the search performance of a group of 24 experienced users on an interactive retrieval task, which suggests that conclusions based on test collection studies should be treated with some degree of caution. This study however used very few queries and only one basic query type.

A later experiment by Turpin and Hersh [2001] again failed to find any difference between a simple IR system and a system tuned for TREC ad hoc, this time considering a greater number and diversity of queries. Results from this latter experiment suggested that users were able to compensate for poorer IR systems by looking further down the ranked result list, and were able to make judgements on which documents to read based on short snippets. If true, this suggests that rank-based metrics such as mean reciprocal rank (MRR) are less informative than other metrics such as P@10. Later experiments by Wu et al. [2004], however, did not show the same phenomenon. They conclude that the differences observed earlier may be due to search engines not being tailored to the task, rather than a deficiency in experimental method.

Further followup experiments [Turpin and Scholer 2006] considered the correlation between system performance, as measured by MAP, and user performance. Test users were given systems with known MAP scores (queries were silently ignored in favour of prefabricated result sets), and two tasks assigned: one recall-oriented and one precision-oriented. In the recall-oriented task, improved MAP did not generally have a significant effect on user performance (although a significant but small difference was found between MAP 0.75 and MAP 0.65 or 0.55). In the precision-oriented task, MAP had no significant effect on the time to find the first relevant document, although there was a suggestion that P@1 may have been important. Although MAP can reliably rank systems, these findings suggest it may not be a ranking which reflects real user concerns.

Similar experiments by Allan et al. [2005] examined the correlation between accuracy, task time, and *bpref* for a facet recall task. They observed some improvement as *bpref* increased, but this was not consistent and only occasionally significant; their experiments suggest that large improvements would be needed before users saw a real benefit.

Other human experiments in the lab

Wu et al. [2001] compared two document-summarising interfaces by asking users which was easier to use or learn and by comparing searcher effort and the quality of final answers. Again, TREC topics were used (here from the Question Answering track) in place of natural information needs. They found significant differences between questions but also between systems, which suggests that interface details have a real impact.

Hersh et al. [1996] suggest that IR evaluation needs to focus on the outcomes of a search, such as how well a person is able to meet an information need. To this end they compared two search interfaces to the Medline database of medical publications,[7] asking subjects to answer clinical questions (such as: "are steroids useful in resolving the acute exacerbation of chronic obstructive pulmonary disease?"). Subjects were also asked how satisfied they were with the interfaces, which were of very different designs (one offering Boolean search, one offering free text queries with document ranking). Although the systems were very different, no significant difference was found either in the quality of answers given or in satisfaction with the interface. However, participants were not given the opportunity to compare the two interfaces.

Other techniques reported in the literature include post-search or post-experiment questionnaries [White et al. 2005] and manual judgements of results or result sets (e.g. [Balmin et al. 2004; Shen et al. 2005]). These can capture individual information needs and ideas of relevance, and are extensible to dynamic collection, but in their common forms impose a significant burden on test subjects.

Evaluating retrieval systems by human experimentation in the lab is complicated by problems due to collections and problems of artificial information needs, as with test collection methods (p. 133). The complex experimental design, and associated overheads, also make this approach problematic.

8.1.4 Naturalistic observation

A small number of studies have placed an experimenter in the field to observe subjects in the course of their day-to-day information seeking, in order to gain better understanding of user search behaviour. Some others report evaluations in which opinions from system users, either on documents retrieved or on entire systems, are solicited at the time of naturally occurring searches.

In a discussion of evaluation, Cooper [1973] suggested that a metric be chosen to approximate an "ideal" measure of utility, which may not be practical to calculate directly. Assuming the document collection, user interface, and users are the same between systems, his "naïve" measure simply involves dividing interactions randomly between two systems and asking users for an explicit estimate of utility (measured in dollars) after seeing each document. The sum of document utility scores provides a search utility score, and the mean of search utility scores provides a measure of the

[7] `http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed`

utility of each system. In this formulation, users are questioned by an interviewer as they use the system; the interviewer must be extremely careful not to disrupt the users' normal search behaviour. Measuring utility directly in dollar figures may not always be easy or appropriate, as Cooper acknowledges, and conducting the interviews would be anything but easy; however it seems likely that explicit judgements from real users are invaluable. It appears that Cooper's methods have not been used for any actual evaluations (although see Lopatovska [2006] for early investigations).

Although relatively rarely used in information retrieval, observation has been used to assess online catalogues in library environments. (See for example Large and Beheshti [1997] for a review.) Hancock-Beaulieu [1990] observed library users as they used catalogue services, either online or via printed indexes, and then browsed the shelves. She noted some differences in behaviour due to the online catalogue, but in general differences which had been expected were not observed. In the same vein, Nordlie [1999] observed similar behaviour between library users who used an online catalogue and those who talked to a reference librarian. (In particular, queries started vague and were disambiguated later, and choice of terms was poor.) Later interactions with librarians and online systems were substantially different, and these differences allowed Nordlie to make recommendations for the design of online tools.

A study reported by Hansen and Järvelin [2000] used a combination of informal interviews, diaries, logs, think-aloud sessions, and structured questionnaires to investigate the information searching behaviour of staff in the Swedish Patent Office. Collaborative retrieval tasks appeared important in this context, and were highlighted for further study.

There are serious problems with applying embedded observational techniques to personal metasearch. First, it seems far too expensive for the benefit gained to employ experimenters to observe search behaviour of enough individual subjects over enough time to obtain an accurate general picture. There is likely to be large variability across the population. Search activities may also occur at any time of the day and usually comprise only a tiny proportion of a person's overall activities.

Furthermore, there are serious risks of altering the behaviour being observed. The mere presence of a search observer hovering in an office or around the home computer may seriously affect what searches are conducted and how. Asking a subject to vocalise their information needs or the process they are following is almost certain to affect search behaviour.

An alternative approach which avoids these objections to some extent is to use instrumented search software, which records aspects of interactions for later analysis. In a comprehensive process, Kelly and Belkin [2004] used monitoring software on specially-configured laptops to gather very extensive information on user's interactions with the Web. A more targeted approach was employed by Claypool et al. [2001] to examine implicit indicators of interest. Their "Curious Browser" recorded users' actions as they browsed the Web and asked for an evaluation of interest, on a five-point scale, the first time each user left each page. Similarly, Dumais et al. [2003] used pre- and post-search questionnaires and recordings of interface actions to evaluate their "Stuff I've Seen" retrieval system with natural information needs in a large organisation, and Fox et al. [2005] used interface recording in conjunction with prompts for explicit relevance judgement. This is similar to the method suggested in the following section, but the technique introduced here considers feedback on sets of results rather than individual results and compares two systems in the context of a particular query.

8.2 Embedded comparisons

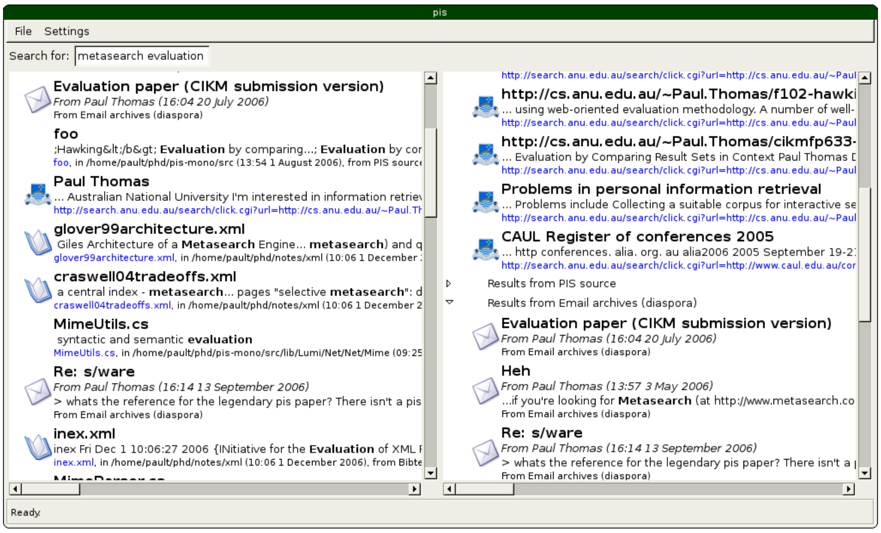
In the future, it may become possible to accurately predict the actual effectiveness of personal metasearch (or other information retrieval) tools from test collections. Experiments with human subjects will also be able to assist in designing and comparing better retrieval tools.

In the meantime, collecting observational data about real search is a prerequisite both for choosing or building appropriate test collections and for designing useful human experiments. Following Hawking et al. [2005], we propose a means by which some types of such data can be easily collected while at the same time allowing experimental conditions to be compared in full context. The approach combines aspects of embedded observation techniques and search log analysis. It supports a large range of real-world information needs and judgements by providing a real, working, retrieval system which takes the place of the searcher's usual system and logs interactions.

Comparisons are however needed between two or more systems, and to have users participate in our tests the additional overhead must be minimal. To this end we propose an interface that provides two or more panels, each of which presents a different retrieval system. (These "different systems" need not be completely independent; for example, the panels may compare different presentations of results from the same system, or variants of the same system with different parameters.) A single field for user queries, which are passed to each system in parallel, provides an interface not much more complex than users are accustomed to. Figure 8.1 shows two examples: the PIS software of Section 3.4, with two panels comparing methods for result ordering, and a web-based version used in most of the experiments described in this chapter.

At the start of each search, the systems are randomly assigned to the panels to control for any bias towards, for example, the left-hand side. By logging interactions with each panel, it is possible to infer which result set or presentation is preferred. It is also possible to ask for explicit judgements of preference.

Using a live search system has several advantages. The collections being searched are those really available to users, and since the experiments need never divulge details of any document it is possible to use private or otherwise restricted collections. Information needs are those users genuinely encounter day to day, and similarly judgements can be made which account for context and the type of information required. Since user satisfaction is recorded for a result set, not relevance of each document, entire result sets can be judged — for coverage, for example, or variety — if this is important to users.



Search for: metasearch evaluation

Evaluation paper (CIKM submission version)
From Paul Thomas (16:04 20 July 2006)
From Email archives (diaspora)

foo
;Hawking</b> **Evaluation** by comparing...; **Evaluation** by cor
foo, in /home/pault/phd/pis-mono/src (13:54 1 August 2006), from PIS source

Paul Thomas
... Australian National University I'm interested in information retriev
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul.Th

glover99architecture.xml
Giles Architecture of a **Metasearch** Engine... **metasearch**) and q
glover99architecture.xml, in /home/pault/phd/notes/xml (10:06 1 December

craswell04tradeoffs.xml
a central index - **metasearch**... pages "selective **metasearch**": d
craswell04tradeoffs.xml, in /home/pault/phd/notes/xml (10:06 1 December 2

MimeUtils.cs
syntactic and semantic **evaluation**
MimeUtils.cs, in /home/pault/phd/pis-mono/src/lib/Lumi/Net/Net/Mime (09:25

Re: s/ware
From Paul Thomas (16:14 13 September 2006)
> whats the reference for the legendary pis paper? There isn't a pis
From Email archives (diaspora)

inex.xml
inex Fri Dec 1 10:06:27 2006 {INitiative for the **Evaluation** of XML F
inex.xml, in /home/pault/phd/notes/xml (10:06 1 December 2006), from Bibte

http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul

http://cs.anu.edu.au/~Paul.Thomas/f102-hawki
... using web-oriented evaluation methodology. A number of well-
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul

http://cs.anu.edu.au/~Paul.Thomas/cikmfp633-
... Evaluation by Comparing Result Sets in Context Paul Thomas D
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul

Problems in personal information retrieval
... Problems include Collecting a suitable corpus for interactive se
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul

CAUL Register of conferences 2005
... http conferences. alia. org. au alia2006 2005 September 19-2
http://search.anu.edu.au/search/click.cgi?url=http://www.caul.edu.au/cor

▷ Results from PIS source

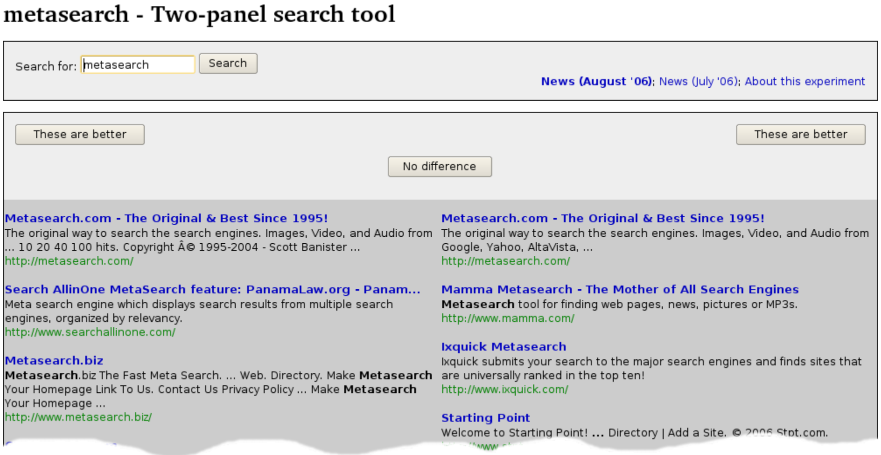
▽ Results from Email archives (diaspora)

Evaluation paper (CIKM submission version)
From Paul Thomas (16:04 20 July 2006)
From Email archives (diaspora)

Heh
From Paul Thomas (13:57 3 May 2006)
...if you're looking for **Metasearch** (at http://www.metasearch.co
From Email archives (diaspora)

Re: s/ware
From Paul Thomas (16:14 13 September 2006)
> whats the reference for the legendary pis paper? There isn't a
From Email archives (diaspora)

(a) Implemented in PIS



metasearch - Two-panel search tool

Search for: metasearch

News (August '06); News (July '06); About this experiment

Metasearch.com - The Original & Best Since 1995!
The original way to search the search engines. Images, Video, and Audio from ... 10 20 40 100 hits. Copyright Â© 1995-2004 - Scott Banister ...
http://metasearch.com/

Search AllinOne MetaSearch feature: PanamaLaw.org - Panam...
Meta search engine which displays search results from multiple search engines, organized by relevancy.
http://www.searchallinone.com/

Metasearch.biz
Metasearch.biz The Fast Meta Search. ... Web. Directory. Make **Metasearch** Your Homepage Link To Us. Contact Us Privacy Policy ... Make **Metasearch** Your Homepage ...
http://www.metasearch.biz/

Metasearch.com - The Original & Best Since 1995!
The original way to search the search engines. Images, Video, and Audio from Google, Yahoo, AltaVista, ...
http://metasearch.com/

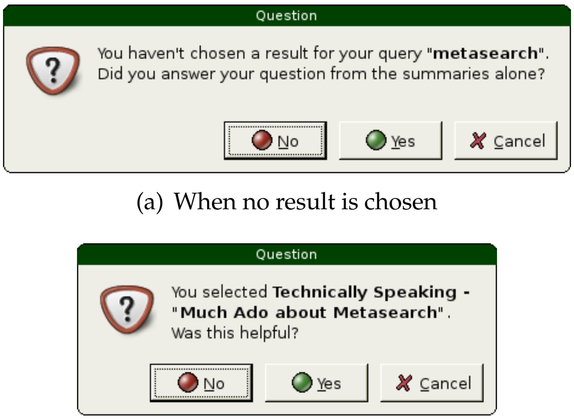
Mamma Metasearch - The Mother of All Search Engines
Metasearch tool for finding web pages, news, pictures or MP3s.
http://www.mamma.com/

Ixquick Metasearch
Ixquick submits your search to the major search engines and finds sites that are universally ranked in the top ten!
http://www.ixquick.com/

Starting Point
Welcome to Starting Point! **...** Directory | Add a Site. © 2006 Stpt.com.

(b) Implemented on the web

Figure 8.1: Sample two-panel interfaces, as implemented in the PIS tool (Section 3.4) and the Web search tool of experiments 1–3.



You haven't chosen a result for your query **"metasearch"**.
Did you answer your question from the summaries alone?

(a) When no result is chosen

You selected **Technically Speaking - "Much Ado about Metasearch"**.
Was this helpful?

(b) When a result has been chosen

Figure 8.2: Extra feedback from the two-panel interface

Presenting two systems side by side and eliciting either explicit or implicit comparisons controls for differences between subjects (because the same person compares both conditions) and also controls for presentational order effects (because the two conditions are presented simultaneously). Furthermore, subjects are required to judge differences rather than to make absolute ratings which will later be compared.

8.2.1 Alternatives and extras

An alternative interface could simply interleave result lists from two or more systems, and use implicit feedback such as clickthroughs to estimate document relevance and hence system performance [Joachims 2002a]. The two-panel interface has some advantages over this approach. First, it is possible to compare alternative, non-list, presentations: for example, it can compare two clustering algorithms or compare graphical presentations such as Kartoo[8] with lists. Secondly, by keeping entire sets of results together the two-panel interface allows judgements to be made of entire result sets. These set-based judgements cannot be made if result sets are commingled.

At the expense of greater intrusion and effort for test subjects, the interface also allows experimenters to prompt for extra information. The PIS software currently does this in two cases: periodically after a result is selected, to ask whether it was useful; and when no result is selected, to distinguish brilliant success from abject failure.[9] Figure 8.2 has examples.

This extra feedback could also be used as training data for algorithms such as decision-theoretic or relevant document server selection [Fuhr 1999; Voorhees et al. 1994]. In cases where one collection is used by several people — for example, public web sites or corporate databases — feedback could be shared and used to tune these algorithms. Search over private collections, such as calendars and email archives,

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

[9] Other logging can detect query bounce, and there is no "reload" command, which completes the set of possibilites described in Section 8.1.2.

would not benefit in this way since many hundreds or even thousands of judgements are generally required.

A variant of this tool could be used to compare systems in a similar manner to TREC ad hoc, by combining the results from two systems into one list and asking users to judge each result for relevance. This bears close similarity to the evaluation of Shen et al. [2005]. Again, however, the advantages of set-based judgements are lost in this scenario, and judging many documents will require more participant effort than comparing two sets.

8.2.2 Limitations of the approach

Like other forms of embedded observation, the method of embedded comparisons is possibly subject to an experimenter effect. Subjects are inevitably aware that they are participating in an experiment and that their actions are being logged for study.[10] Furthermore, even if the metasearcher delivers the same results as their standard service, it presents them in a different way, in less screen area, and likely takes longer to present them. Careful design of the interface is required to minimise these effects. Mechanisms for eliciting feedback must be as unobtrusive as possible and make minimal demands on users.

Second, experiments of this nature are not repeatable in particular ways: for example, without access to the collection we cannot re-run queries with a different IR system. We believe, nonetheless, that the ability to directly compare two IR systems in a natural setting is invaluable.

Third, while it may be easy to show from a set of pairwise comparisons that system A is categorically better than system B, it is much harder to know by how much, or exactly why. It is also difficult to make multi-way comparisons. Dividing a screen into more than two panels is feasible but it magnifies the experimenter effect by forcing larger presentational changes, requiring more time for judging and making the judging interface more complex. The alternative approach of making multiple pairwise comparisons inevitably makes it harder to control for inter-subject variation and order effects, and needs significantly more effort.[11]

Finally, if one panel takes significantly longer than the other to retrieve results, this could be the source of significant bias. This is controlled in the experiments reported here by retrieving all results for both panels before displaying either.

8.2.3 Implementation

The design described above has been implemented in three versions, illustrated in Figure 8.1.[12]

[10] Participants in the experiments described here and in the next chapter were given information on the logging process, and could ask to have their records removed.

[11] To compare three systems requires three times as many judgements as to compare two; to compare four systems requires six times as many judgements.

[12] A fourth version, providing a two-panel web-based front-end to PIS, has been developed but was not used for any work reported in this thesis.

The PIS personal metasearch system, described in Section 3.4 and in Appendix E, can display any number of panels side-by-side with different result merging and/ or server selection strategies for each. It records document selection for each panel, and includes the two pop-ups of Figure 8.2: one appears occasionally after selecting a result (asking whether a selected result was useful) and can inform the use of click-through data as a proxy for user preference, and the other (asking whether document summaries were good enough to answer the user's need) appears if no document is selected and can distinguish brilliant success from abject failure.

The second and third versions are web-based and provide two panels, each with results from Web search engines. Both versions record clicks on results, but the stateless nature of web protocols and limits on scripting prevented the implementation of pop-ups for extra feedback. The two versions are identical, except that one (used in experiments 1–3 below) includes a set of three buttons for explicit judgements of result sets and the other (used in experiment 4) does not.

8.3 Validating the design

Four experiments have verified that the approach described above can provide a useful comparison between two search systems. The first three of these requested explicit feedback on two search systems, one of which was known to be better than the other, and the fourth looked only at implicit feedback given the same two systems.

In the results reported below, one system is considered preferred to another if the number of users reporting it as better overall is greater than the number reporting the other plus the number with no overall preference. (This is a conservative approach.) Users who submitted no judgements were not counted. Significance is measured with a binomial sign test with criterion $p < 0.05$.

The instruments used to recruit participants, elicit demographic information, and carry out searches are described in Appendix C.

8.3.1 First experiment: popular queries

The first experiment addressed two questions:

1. Given two result sets with a difference in quality, do users' judgements reflect this difference?

2. If so, can the two-panel design tell which set is better?

Question 1 acts as a validity check — if the answer is "no", and users' judgements do not seem to distinguish a supposedly high-quality from a supposedly low-quality result set, other results will be very doubtful. Assuming however that users agree there is a real difference, Question 2 is key. In answering this, the first experiment considerd two further questions:

3. To what extent do users tend to prefer the left-hand panel, as the one they read first (or the right-hand panel, as the one they read last)?

Age	22–54 (mean 35, std. dev. 10.5)
Sex	Male: 16, female: 6
Education	Postgraduate degree: 16, first degree: 6
Computer use	Daily: 21, occasional: 1
Web use	Daily: 21, occasional: 1
Search engine use	Daily: 17, occasional: 5
Computer experience	7–37 years (mean 19, sd 8.4)
Web experience	5–12 years (mean 9, sd 2.1)
Search engine exp.	5–11 years (mean 8, sd 2.4)

Table 8.1: User demographics for first experiment (23 users total). Not all users answered all questions.

4. To what extent do click patterns, timing, or other implicit feedback correlate with user judgements?

If the left- or right-hand bias is small, and one or more types of implicit feedback correlates well with stated preference, then it is possible to use this implicit feedback and a two-panel design to compare systems. If, on the other hand, no implicit data seems to correlate with user judgements, a two-panel design can only be used with explicit judgements.

A convenience sample of 23 users were recruited via email and posters. These participants were given the web-based software illustrated in Figure 8.1(b); this simply acted as a proxy to the Google search engine,[13] which is known to return good quality result sets [Vaughan 2004]. One panel, chosen at random, displayed Google's first ten results, which was assumed to be a (relatively) high-quality result set. The other panel, assumed to be of (relatively) low-quality, displayed Google's 21st through 30th results. Test participants were each given ten topics from Google's list of popular searches[14] and after each search were prompted onscreen to indicate which set of results were "better", if either. No further definition of "better" was given.

306 queries were recorded, indicating a small amount of query reformulation or experimenting with the interface (the minimum number of queries, if all 23 users issued only ten queries each, would be 230). 239 judgements were recorded from these queries, of which 183 were judgements in favour of one panel or the other and the remaining 56 were of "no difference".

Participants were asked a number of optional demographic questions prior to the experiment, the responses to which are summarised in Table 8.1.

[13] `http://www.google.com/`

[14] `http://www.google.com/intl/en/press/zeitgeist/archive2005.html`, downloaded 20 December 2005. Top queries for English-speaking countries (Australia, Canada, Ireland, New Zealand, South Africa, and the United Kingdom) from June to September 2005 were selected, duplicates removed, and a small number of apparently pornographic requests removed. 100 queries remained; a sample is given in Section C.2.

Results

As hoped, there was a significant preference for the higher-quality set of results. 19 users preferred the high-quality set overall, 1 the lower-quality set, and 1 had no overall preference (sign test: ratio of 19:2 with $\pi_1 = 0.5$ gives $p \ll 0.01$). (The remaining two users made no judgements and were excluded from the analysis.) There was no significant difference in the preference for result sets in the left-hand or right-hand panels (sign test, $p = 0.09$).

Of the 183 queries where a preference was recorded, 34 included associated clickthrough data with between one and ten clicks per query. To investigate whether implicit feedback correlated with eventual explicit judgements, four attributes of this clickthrough data were considered:

- The panel which received the *first click*; this may be the panel with the most promising single result. This was the panel which received the final preference in 67% of cases.

- The panel which received the *last click*; this may be the panel with the single result which finally resolved the information need. This was the panel which received the final preference in 70% of cases.

- The panel which received the *most clicks*; this may be the panel with the most useful results. This predictor was 73% accurate.

- The panel which received the *highest-ranked click*: for example, if the fourth result from the left-hand panel was clicked, as was the second from the right-hand panel, the right-hand panel would be predicted for the user's final preference as $2 < 4$. This predictor may capture the panel with better results or better ranking, and in this experiment was 79% accurate.

In this experiment, since users were assigned tasks and did not have a real need for information, the clickthrough rate is low and click patterns may be different to those in a more natural setting. This is considered further in other experiments below.

Each of these four predictors performed well above chance (sign test, $p = 0.04$ to 0.01). The web tool also recorded the time taken before the first result was chosen, but there was no apparent connection between this and the final judgement. Finally, there was no significant difference in the number of clicks in each panel ($p = 0.43$).

These first results are very encouraging. In answer to Question 1, users' judgements certainly do reflect differences in quality; this suggests that they are able to judge two result sets when they are presented side-by-side in this manner. Further, it seems the side-by-side design can indicate which set of results, and hence which search service, users prefer (Question 2). There is no significant bias to either side (Question 3), and results suggest that clickthrough data may be able to predict user preference with this design (Question 4). Participants in this experiment did cover a limited demographic range — in particular, they were all well-educated and experienced with search engines and the Web — but early adopters of metasearch tools are likely to be similar.

Age	24–52 (mean 33, std. dev. 7.1)
Sex	Male: 16, female: 4
Education	Postgraduate degree: 14, first degree: 6
Computer use	Daily: 20
Web use	Daily: 20
Search engine use	Daily: 19, occasional: 1
Computer experience	8–37 years (mean 21, sd 6.7)
Web experience	8–13 years (mean 11, sd 1.4)
Search engine exp.	5–13 years (mean 9, sd 1.9)

Table 8.2: User demographics for second experiment (21 users total). Not all users answered all questions.

8.3.2 Second experiment: natural queries

Results from the first experiment were encouraging, but participants had used artificial information needs. Given naturally-occuring needs and natural queries, would results be similar?

A second experiment considered the same questions and used the same technique as the first, but participants were not assigned search tasks: they were instead encouraged to use the web-based tool in place of their regular Web search engine. Since users were issuing their own queries, for their own needs, and in their own time, this gives a good indication of how well a two-panel evaluation method would work for retrieval systems in the field. For privacy, users were reminded that they could choose not to use the experimental software in cases where they would prefer data not be recorded, but we expect this to have had only a small impact on the number and range of tasks represented.

Demographic data was collected for this second experiment in the same manner as for the first. The results were similar and are summarised in Table 8.2. Users were not identified, but there is likely to have been a small amount of overlap between participants in this and the first experiment.

179 queries were recorded and 147 judgements, of which 119 were in favour of one panel or the other and the remaining 28 were for “no difference”.

Results

Despite the differences in queries and information needs, results were similar to the first experiment. 17 of 20 users preferred the assumed high-quality result set overall (a further 2 preferred the low-quality set and 1 had no overall preference; $p \ll 0.01$). Again, there was no significant difference in judgements for, or clicks in, the left- or right-hand panel (judgements $p = 0.32$, clicks $p = 0.12$).

85 of the 179 queries had both clickthrough data and an explicit judgement, a higher proportion than in the earlier experiment. The same four clickthrough attributes were considered as predictors of the final judgement:

- The first click predictor was accurate 81% of the time;
- the last click predictor was accurate 85% of the time;
- the most clicks predictor was accurate 85% of the time;
- and the highest ranked click predictor was 81% accurate.

Again all these results are significantly better than chance alone ($p \ll 0.01$).

The second experiment has a much higher conversion rate from queries to clicks, and a marginally lower rate of "no difference" judgements. This may be explained by observing that in this experiment users are carrying out their own searches to fulfil genuine information needs. The higher click rate may be a reflection of the inadequacy of document summaries for some needs, and the lower "no difference" rate a reflection of a stronger sense of what constitutes a useful set of documents.

8.3.3 Third experiment: overlapping result sets

In the first two experiments, result sets were disjoint by construction. This ensures a large difference in quality, which although useful for validating the embedded comparisons method is perhaps not realistic.

Two systems, operating over the same collections, are likely to produce result sets that are at least somewhat similar. A third experiment therefore considered the case where result sets overlap.

The same method was used as for the second experiment above, but with Google's results 1–10 in one panel and results 6–15 in the other (so there was an overlap between the bottom five results of one set and the top five results of the other). This has two effects, which provide a possibly more realistic test: first, there is significant (50%) overlap between the two result sets, so that useful documents are more likely to appear in both panels. Secondly, the difference in quality is much reduced. The question asked by this experiment is: in these cases, where there will likely be less difference in quality, can embedded comparisons still predict which set is preferred?

348 queries were recorded and 121 judgements, 79 of which were for one panel or the other.

Demographics of the 36 users in this experiment, which included some users from the first two experiments, were similar. Table 8.3 has a summary.

Results

Despite the overlap and the smaller difference in quality, results proved similar to the second experiment. Of the 25 users who offered one or more judgements, 18 preferred Google's results 1–10 overall and 6 Google's results 6–15; the remaining user had no

Age	24–50 (mean 32, std. dev. 5.6)
Sex	Male: 18, female: 17
Education	Postgraduate degree: 26, first degree: 8, other post-school qualification: 1
Computer use	Daily: 35
Web use	Daily: 32
Search engine use	Daily: 30, occasional: 4
Computer experience	4–30 years (mean 14, sd 5.7)
Web experience	4–17 years (mean 10, sd 2.9)
Search engine exp.	3–15 years (mean 9, sd 2.6)

Table 8.3: User demographics for third experiment (36 users total). Not all users answered all questions.

overall preference. Although still significant ($p = 0.02$), this is a smaller difference than observed when the two sets did not overlap. There was no significant difference in the number of judgements in favour of either panel ($p = 0.054$), although there were more clicks recorded on the right-hand than left-hand panel ($p = 0.03$). Since the systems are randomly allocated to panels with each query, and other experiments did not show this bias, this does not seem a significant problem.

The four attributes of clickthrough data considered earlier remained good predictors of the final judgement: the first click predictor was 86% accurate, the last click predictor 84% accurate, the most clicks predictor 80% accurate, and the highest ranked click predictor 86% accurate. All performed significantly better than chance ($p \ll 0.01$).

8.3.4 Fourth experiment: implicit feedback

The first three experiments suggest that in the context of the two-panel design, clickthrough data is a good predictor of final preference; they also suggest that users have a strong preference for a higher-quality result set. These observations suggested a final experiment, using the same system as the second experiment but with the voting buttons removed. Again users were told to use our system as they would any other web search service. The question this experiment asks is: assuming users prefer the high-quality set, does clickthrough data still predict this preference in the absence of the explicit voting step? It is possible, for example, that the presence of the explicit voting prompts users to read both result sets more carefully. Without these their reading, and hence click patterns, may be different.

Demographics of the 18 test users were again similar, and are summarised in Table 8.4.

Age	28–54 (mean 38, std. dev. 9.3)
Sex	Male: 13, female: 3
Education	Postgraduate degree: 12, first degree: 4
Computer use	Daily: 16
Web use	Daily: 16
Search engine use	Daily: 14, occasional: 2
Computer experience	9–37 years (mean 22, sd 6.6)
Web experience	8–16 years (mean 11, sd 2.1)
Search engine exp.	6–14 years (mean 10, sd 2.1)

Table 8.4: User demographics for fourth experiment (18 users total). Not all users answered all questions.

Results

The four attributes of clickthrough data remained good predictors of the high-quality result set. Agreement with the first click predictor was 70%, with the last click predictor was 75%, with the most clicks predictor 68%, and with the highest rank clicked predictor 87% over 129 queries with clickthrough data recorded ($p \ll 0.01$). As in the earlier experiments there was no significant difference in the number of clicks in each panel ($p = 0.21$), suggesting no bias towards one panel or the other on the basis of their position alone. There is no data on judgements for each panel, since there were no explicit judgements in this experiment.

These results strongly suggest that using clickthrough data alone, in place of explicit judgements, in the embedded evaluation method could provide a robust comparison of two retrieval systems. This would significantly reduce the burden on both test users and experimenters.

8.3.5 Observations

Although privacy concerns made it impossible to contact test users directly, a number offered informal feedback on their use of the two-panel design. Comments were positive: none reported finding the two panels distracting (one even found this layout more useful, and requested that the software remain available long-term). Several users commented that they found the process of scanning the result sets easy, but that especially in experiment three there was sometimes no reason to choose one over the other; this is consistent with reducing the difference in quality and allowing overlap.

There has not been any formal investigation, but these comments and the number of queries collected suggest there is minimal extra burden for test users in comparing results side-by-side.

Age	20–54 (mean 33, std. dev. 9.4)
Sex	Male: 33, female: 11
Education	Postgraduate degree: 23, first degree: 18, other post-school qualification: 3
Computer use	Daily: 45
Web use	Daily: 43
Search engine use	Daily: 42, occasional: 2
Computer experience	10–38 years (mean 19, sd 6.9)
Web experience	5–15 years (mean 10, sd 1.9)
Search engine exp.	3–12 years (mean 9, sd 2.1)

Table 8.5: User demographics for whole-of-Web case study (49 users total). Not all users answered all questions.

8.4 Case study

A simple case study has demonstrated the method in action. This compared two major whole-of-Web search engines, as exposed by their public APIs. (These are called "engine A" and "engine B" below.) Again, users were asked to use the web-based tool as they would a regular Web search engine, and to indicate which set of results, if any, they preferred. The result sets were not re-ranked or modified except to ensure consistent display, and overlap between the two was allowed.

49 users participated in this experiment; demographic details are summarised in Table 8.5. 444 queries and 250 judgements were recorded; only 158 judgements were in favour of one search engine or the other and the remaining 92 were of "no difference".

Of the 40 users who recorded one or more judgements, 25 users preferred engine A overall and 13 engine B; two users had no overall preference. This is not a significant preference (binomial sign test, $p < 0.08$). Once again, there was no apparent difference in the number of judgements or clicks for either the left- or right-hand panel (judgements $p = 0.21$, clicks $p = 0.15$).

As in the validation experiments, clickthrough data was examined to determine whether it reflected overall judgements. There were 117 queries recorded with both explicit judgements and clickthrough data; the four predictors again fared well. The first click predictor was accurate in 78% of cases, the last click predictor in 77%, the most clicks predictor in 77%, and the highest ranked click predictor in 84%. These are all significantly better than chance ($p \ll 0.01$).

Using this method it was possible to record user satisfaction directly. This is likely to indicate how users rate search engine usefulness in real situations. By comparison, methods based on precision or similar scores indicate performance only in an abstract way and do not necessarily reflect user satisfaction.

8.5 Applications

The experiments described demonstrate that the embedded comparisons method can capture useful judgements in realistic settings. The primary purpose of experimenting with the method was to enable the evaluation of personal metasearch systems, which are otherwise hard to evaluate. The technique could however be used to shed light on many other questions in metasearch, including:

1. Is it better to merge metasearch results into a single ranked list, to segment by source, or to cluster?
2. Do searchers prefer result sets which take into account aspects of their context such as role or location?
3. Do query-biased summaries help users?
4. Is the exact order of high-ranked results important?
5. How much does it matter which results are "below the fold" and not on the first screen of results?

There are also potential applications outside the immediate domain of metasearch. Questions 3–5 above are also relevant to many other types of search, such as general web search, enterprise search, encyclopædia search, and medical abstract search. The technique could also be used in guiding purchasing decisions for website and enterprise search software.

This method has been used to investigate the influence of branding on the perceived quality of Web result sets [Bailey et al. 2007]. In this case it is valuable to work with real information needs and a changing collection; further, by manipulating the presentation it is possible to record user preferences which are due only to branding (such as search engine name). A traditional evaluation based on counting "correct" documents would not be useful in this instance.

A variant of this method, with three panels and explicit ratings for each, has also been used to elicit judgements in context for bibliographic search [Krumpholz and Hawking 2006].

8.6 Conclusions

After considering familiar evaluation methodologies, it seems that a new technique is more appropriate to the sorts of dynamic and private collections which are likely to be covered by personal metasearch tools. *Embedded comparisons* are based on a tool which takes the place of a user's normal search interface and offers results from two systems side-by-side. This tool can collect queries, interactions, and explicit judgements as they occur, and can be used with private or dynamic collections. Use of the tool avoids many of the costs and biases of familiar evaluation methods.

	Clicks	Preferred	Implicit feedback			
Experiment	LHS/RHS	High/low quality	First	Last	Most	Highest
Popular queries	neither	high	67%	70%	73%	79%
Natural queries	neither	high	81%	85%	85%	81%
Overlap	RHS	high	86%	84%	80%	86%
No judgements	neither	n/a	70%	75%	68%	87%

Table 8.6: Summary of validation experiments. For each experiment: whether the left- (LHS) or right-hand side (RHS) received more clicks; whether the high- or low-quality result set was preferred overall; and the predictive accuracy of the first and last click, the side receiving the most clicks, and the side with the highest-ranked click.

Experiments to validate the method, summarised in Table 8.6, confirm that it is possible to detect a difference in user preferences between a high-quality set of results and a lower-quality set. There is no significant preference for the left- or right-hand panel of results, although in one experiment the right-hand panel received more clicks.

The experiments and case study also demonstrated that clickthrough data is useful in inferring preference in side-by-side presentations of web results, even in the absence of explicit user judgements. Demonstration of the method in a case study has indicated that it is a viable means of gathering preference data with real users and real information needs.

Although designed for personal metasearch, the method is useful for other evaluations of retrieval systems. A comparison of whole-of-Web search engines, for example, has been demonstrated and could be of particular value to Web search providers as it captures user preference directly.

Having demonstrated that this method is a reliable means of comparing systems in the sorts of scenarios imagined for personal metasearch, experiments in the next chapter use embedded comparisons to evaluate the server selection mechanisms discussed in Chapter 7 with a real, working personal metasearch tool operating “in the wild”.

Chapter 9

Evaluating server selection

Experiments in Chapter 8 have demonstrated that embedded comparisons provide a straightforward method to evaluate search over personal, dynamic collections such as those in personal metasearch. The embedded comparisons technique also appears to impose little impact on users, suggesting it can be incorporated in a working metasearch tool and used with searches day to day.

A final experiment demonstrated this technique, and further investigated server selection in a personal metasearch environment, by comparing two selection methods in a working metasearch tool installed on participants' personal computers.

Participants preferred Kullback-Leibler divergence to vGlOSS, in line with earlier experiments, but the effect was smaller than seen with evaluation based on the test collection. This suggests that test collections may overestimate the impact of differences in performance. The embedded comparisons technique appears to be of value, and indications are that it could scale to large numbers of users.

9.1 Experiment

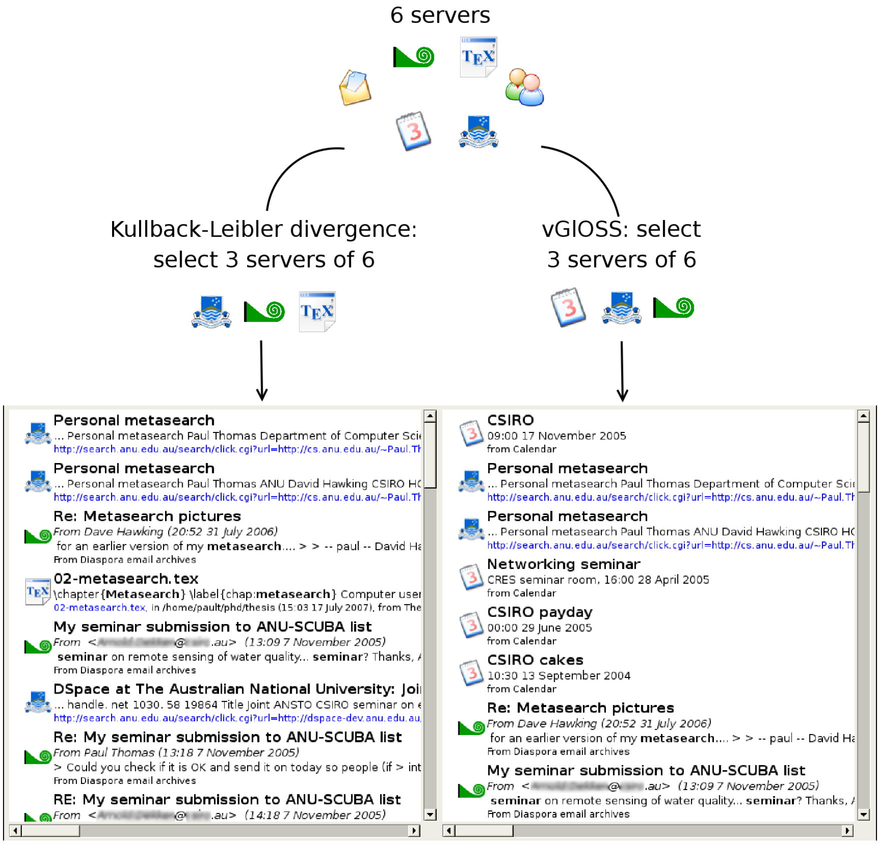
The questions this experiment asked were:

1. Does the embedded comparisons technique work in a less controlled environment than that of Chapter 8, despite the complications of a working personal metasearch tool?

2. If so, do users' preferences for server selection bear out the findings of Chapter 7, which were based on simulated information needs?

Seven users were recruited from amongst those participants in the survey of Section 3.3 who had expressed interest in further involvement. These users were selected to demonstrate a range of possible collections and servers, as well as for their willingness to use prototype software; although a convenience sample, they are likely to resemble early adopters of any personal metasearch tool.

Each participant was given a copy of the personal information search (PIS) prototype software described in Section 3.4. The software was configured to present two


6 servers
Kullback-Leibler divergence: select 3 servers of 6
vGlOSS: select 3 servers of 6

Personal metasearch
... Personal metasearch Paul Thomas Department of Computer Scie
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul.Th

Personal metasearch
... Personal metasearch Paul Thomas ANU David Hawking CSIRO HC
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul.Th

Re: Metasearch pictures
From Dave Hawking (20:52 31 July 2006)
for an earlier version of my **metasearch**.... > > -- paul -- David Ha
From Diaspora email archives

02-metasearch.tex
\chapter{**Metasearch**} \label{chap:**metasearch**} Computer user
02-metasearch.tex, in /home/pault/phd/thesis (15:03 17 July 2007), from The

My seminar submission to ANU-SCUBA list
From <[redacted].au> (13:09 7 November 2005)
seminar on remote sensing of water quality... **seminar**? Thanks, A
From Diaspora email archives

DSpace at The Australian National University: Joi
... handle. net 1030. 58 19864 Title Joint ANSTO CSIRO seminar on e
http://search.anu.edu.au/search/click.cgi?url=http://dspace-dev.anu.edu.au

Re: My seminar submission to ANU-SCUBA list
From Paul Thomas (13:18 7 November 2005)
> Could you check if it is OK and send it on today so people (if > int
From Diaspora email archives

RE: My seminar submission to ANU-SCUBA list
From <[redacted].au> (14:18 7 November 2005)

CSIRO
09:00 17 November 2005
from Calendar

Personal metasearch
... Personal metasearch Paul Thomas Department of Computer Scie
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul.Th

Personal metasearch
... Personal metasearch Paul Thomas ANU David Hawking CSIRO HC
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul.Th

Networking seminar
CRES seminar room, 16:00 28 April 2005
from Calendar

CSIRO payday
00:00 29 June 2005
from Calendar

CSIRO cakes
10:30 13 September 2004
from Calendar

Re: Metasearch pictures
From Dave Hawking (20:52 31 July 2006)
for an earlier version of my **metasearch**.... > > -- paul -- David Ha
From Diaspora email archives

My seminar submission to ANU-SCUBA list
From <[redacted].au> (13:09 7 November 2005)
seminar on remote sensing of water quality... **seminar**? Thanks, A
From Diaspora email archives

Figure 9.1: Server selection for each panel in PIS. This illustrates a hypothetical participant using PIS to integrate six servers; other participants may use different collections or a different number of collections. The left-to-right ordering was randomised for each query.

panels, in the manner of Section 8.2. Both panels used the SM-TSS algorithm [Rasolofo et al. 2003] to merge and re-sort results from different servers, but servers were selected in two ways: one panel, chosen at random, scored servers using Kullback-Leibler divergence (Section 7.1.6), which did well in the experiments in Chapter 7, and the other used vGlOSS (Section 7.1.1), which was not as promising. In both cases, term frequencies were estimated from unigram language models and the top-scoring 50% of servers were selected (subject to a minimum of two). Figure 9.1 illustrates the process.

Although participants were not asked what collections they included, and PIS did not record this information, informal feedback indicated that a variety of collections and servers were used including email, in various formats; local files, also in various formats; Wikipedia and other reference Web sites; intranets; wikis; the public Web; LDAP directories; and local databases.

PIS is a hybrid metasearcher and is able to index some collections locally if appropriate. In these cases, models were built from local indexes and were entirely accurate; otherwise models were built using a minimum of 300 documents selected using the multiple queries sampler of Section 4.3 and collection sizes were estimated using multiple capture-recapture (Section 5.1.2), slightly modified to allow for non-uniform sample sizes. These were the best-performing techniques for sampling and size estimation, respectively, in earlier experiments.

PIS logged a small set of data for each user query: the fact that a query had been issued, the number of servers in use, the ordering of the panels for this query, and the panel and rank of any result opened. It was also configured to ask for additional feedback in the two situations described in Section 8.2.1: first, if no document was selected (to distinguish brilliant success from abject failure) and second, in about 50% of cases, after a document was selected (to confirm that it was in fact useful). To minimise work for participants, they were not explicitly asked which panel was preferred. Relying on implicit indicators limited the amount of data available, but demonstrates a plausible "bare-bones" implementation of embedded comparisons.

The experiment ran for 64 days, although not all participants started at the same time. The instruments for this experiment are described in Appendix D, and the PIS tool is described further in Appendix E.

9.2 Results

Results recorded by PIS were automatically sent to a central server for analysis. If additional feedback indicated that a selected result was not helpful, any associated clickthrough data was removed and was not considered as evidence of a preference for either selection algorithm. Further, since PIS enforced a minimum of two servers for any query, clickthrough data was also removed for any query issued when only one or two servers were configured. In total, 273 queries were recorded from seven users; of these, 98 queries had usable clickthrough data with one to four clicks each.

Experiments reported in Chapter 8 suggested four indicators of overall preference: the panel which received the first click, the panel which received the last click, the panel which received the most clicks, and the panel which received the highest-ranked click. The last of these appears to be the most reliable of these indicators (see Section 8.3.4), and was used here. Ties, for example where the top result from each panel was clicked, were broken in favour of the panel which received the first click. (This happened with 11 queries, and there was no appreciable difference if ties were broken in favour of the other panel.)

Results are summarised in Table 9.1. In the experiments of Chapter 7, which used artificial information needs and collections, Kullback-Leibler divergence consistently outperformed vGlOSS across a range of conditions. Kullback-Leibler divergence is significantly preferred by two of the six participants in the current experiment (one--sided binomial sign test, $\alpha = 0.05$, hypothesised ratio $\pi_1 = 0.5$), and vGlOSS by none. This is broadly consistent with the earlier result, but the effect is not as strong as might have been expected. This may be explained by differences between the two styles of evaluation: for example, set-based judgements in this experiment substitute for document-based judgements in earlier experiments, and judgments in this exper-

			Preferred	
User	Servers	Queries	KL divergence	vGlOSS
A	12	86	10⋆	2
B	4	97	38⋆	19
C	8	64	11	5
D	5	18	3	6
E	2	—	—	—
F	3	2	0	2
G	3	2	1	1

Table 9.1: Inferred preferences for server selection. "Servers" is the highest number of servers configured by each user; "queries" reports the total number of queries issued, discounting any with one or two servers. ⋆ indicates a significant preference (one-sided binomial sign test, $\alpha = 0.05$, $\pi_1 = 0.5$).

iment take into account relevant aspects of each user's context. It is also possible that both techniques are good enough, or poor enough, in a real setting that relative differences are less important; or that "information needs" in previous experiments are not like the queries used here. Since this experiment includes real information needs, real users, and real data, we must conclude that previous experiments were able to detect the direction of the difference between Kullback-Leibler divergence and vGlOSS but were unable to predict the true magnitude of this difference. Further, large differences in traditional metrics such as $\mathcal{R}_n$ may be needed before users observe a difference in quality, a conclusion consistent with earlier work by Allan et al. [2005] and Turpin and Scholer [2006].

There were no significant effects due to elapsed time (so users did not appear to change preferences over time), or due to user (so users did not vary significantly), with the exception of user "D" who has a higher than normal preference for vGlOSS. Contrary to previous experience, there were significantly more clicks on results from the left-hand than the right-hand panel (binomial sign test, $p \ll 0.01$), but since panels were assigned at random this is not a concern.

The other three indicators suggested in Section 8.3 could also be used to infer final preference. All are summarised in Table 9.2. With the exception of user "F", who only recorded two queries with associated clickthrough data, there is no appreciable difference using any of the alternative indicators.

9.3 Observations

The participation rate in this experiment was high: of those people approached as potential users, only three who expressed interest pulled out before submitting results. One of those users withdrew due to a bug in PIS which made the tool unusable in a particular configuration, and which could not be fixed in time. The other two withdrawing participants did not give reasons.

	First click		Last click		Most clicks	
User	KL	vGlOSS	KL	vGlOSS	KL	vGlOSS
A	10*	2	10*	2	10*	2
B	38*	19	37*	20	37*	20
C	10	6	11	5	10	6
D	3	6	3	6	3	6
E	—	—	—	—	—	—
F	0	2	1	1	2	0
G	1	1	1	1	1	1

Table 9.2: Inferred preferences for server selection, using alternative indicators. $\star$ indicates a significant preference (one-sided binomial sign test, $\alpha = 0.05$, $\pi_1 = 0.5$).

The high take-up rate likely indicates that personal metasearch, and the PIS prototype in particular, does seem useful to most people; and that the tool, while only a research prototype, was sufficiently robust for day-to-day use.

This experiment included only a small number of participants as each was selected for using a range of collections, including collections of different sizes and data types. It would have been possible, for example, to install PIS across student machines or across an entire workgroup and thus aquire more users, although this would be at the expense of including a much more homogenous set of users and collections. Future work may consider this possibility. Scaling the embedded comparisons technique to a larger group of users seems feasible: some effort was required helping participants install the software, and further effort was required to debug the software and issue new versions as bugs were exposed, but to support the participants in the present experiment little effort was needed overall. As more participants were added, with collections and usage patterns similar to existing users, experimenter effort would likely be relatively small.

9.4 Conclusions

Results from this experiment were broadly in line with those from earlier experiments, reported in Chapter 7, which used a fixed test collection and artificial "information needs"; however the participants in this experiment expressed a much weaker preference for server selection via Kullback-Leibler divergence than would be expected from earlier results. This suggests that the test collection approach, while useful, is likely to be somewhat inaccurate in predicting user preferences. The inaccuracy may be due to differences in judgements (result sets against documents); user context; both methods being good enough, or poor enough, that quality differences are small; a mismatch between the collections and queries used earlier and those used by test participants; or some combination of the above. This in turn suggests that evaluations based on test collections may be somewhat limited, and in particular that large differences on standard metrics may be needed before users appreciate a difference in quality.

The embedded comparisons technique, while requiring significantly more experimenter effort than testbed techniques, appears feasible even with prototype metasearch tools operating over a variety of collections and with minimal intervention. Although this experiment used a small number of participants, indications are that the technique could scale to larger uses without unreasonable effort on the part of experimenters.

Chapter 10

Summary and conclusions

Personal metasearch, introduced in this thesis, is a novel application of metasearch techniques which is motivated by technical concerns and by ease of use, and which a survey of potential users has confirmed could be of great benefit.

Several models have been suggested and examined for retrieval in such an application. Models which are commonly assumed, including a central index and exhaustive metasearch, are not appropriate, and this thesis has argued for a hybrid metasearch model. This in turn suggests the environment metasearch tools must work in, and can inform evaluations of candidate techniques.

Past research has described methods for sampling documents, estimating collection size, summarising collection contents, and selecting appropriate servers. Much of this past work, however, has considered these in artificial test environments; this thesis has re-examined many methods in a realistic metasearch environment and with a potential real-world application in mind. From this it has been possible to suggest which methods may be appropriate for a personal tool, and which are not. New techniques have also been introduced for sampling and for evaluation; the sampling technique is of use to metasearch tools in general, as well as to personal metasearch, and the evaluation technique is useful in a wide variety of retrieval settings including Web and interactive retrieval.

10.1 Summary of findings

This thesis has introduced and examined personal metasearch, a new application of metasearch techniques to provide a single search tool operating over all of a user's digital sources. Such a tool is desirable: compared to the status quo, with a separate tool for each source, a unified tool reduces cognitive load and time spent and avoids certain types of error.

Personal metasearch can be distinguished from alternative search technologies. Desktop search, which is increasingly popular, also provides a single search tool for a variety of data, and personal information management (PIM) tools provide a single environment for managing workflow. In most cases, however, these tools only offer an index of local or networked files and cannot provide search facilities over other commonly-used collections such as corporate databases, intranets, subscription services, or the public Web. The central index model adopted by these tools may also lose any optimisations made possible by specialised tools, such as controlled vocabularies, thesaurus expansions, or specialised ranking methods.

Personal metasearch can also be distinguished from other metasearch applications. In most work to date, metasearch tools have operated over public collections and roughly homogeneous document types: for example, applications have considered the public Web or newswire collections. As an application aimed at individuals, personal metasearch can consider very different collections, including private or restricted collections such as email archives, calendars, or databases. This range of data, and range of collection characteristics, has not otherwise been studied and offers an alternative way to consider personalisation of search tools.

Several models are possible for search over multiple collections, and of those examined here the most feasible appears to be a hybrid metasearch model where some collections are indexed locally and others are available via servers. The desirability and practicality of a personal metasearch tool, working on such a model, has been confirmed by a survey of search users.

A general description of the metasearch process (Section 2.2.2) provides some insight into what is needed for a practical tool. This thesis has focussed on two subproblems: *server characterisation* and *server selection*. Server characterisation in turn involves at least three components: a reliable technique for sampling documents; a method for estimating the size of each server's holdings; and a method for establishing the subject matter and/or language of each.

Several metasearch methods rely on being able to generate an unbiased sample of documents from a collection, without the cooperation of a server. Experiments in this thesis have demonstrated that all of five sampling methods applicable to personal metasearch are subject to some degree to "query bias", a bias towards a particular subset of high-ranking documents, and in particular towards longer documents. This bias results in poorer-quality size estimates, poorer language models, and also leads to poorer server selection. A new sampling method, "multiple queries", performs as well as the best established alternatives but with much reduced computational and network cost.

A second series of experiments in server characterisation has considered techniques for estimating the size of a collection. A number of methods have been proposed, and eight have been considered and tested for their performance in personal metasearch. Although no technique performed well in all cases, broad patterns exist and provide some indications which techniques will be useful in a working tool.

The third characterisation problem considered here was summarising the subject matter and language of collections. Experiments with unigram language models, which estimate the relative frequency of terms in a collection as a guide to its contents, have shown that the quality of a model depends upon the quality of document samples used as input. Models built with the multiple queries sampler are largely indistinguishable, on three measures, from those built with true random samples; models built with the popular query-based sampler are of poorer quality. Models improve as more documents are included, but improvements eventually cease being significant. The point at which this happens is again determined by the sampling technique in use.

With characterisations of each server, it is possible to carry out a process of *server selection* for each query. The problem of server selection has been well studied, and a large number of techniques have been described, but it has not before been considered with the variety of collection sizes and types characteristic of personal metasearch. Experiments with twelve selection methods have demonstrated that several perform well on average, but the majority are biased towards large collections and perform poorly when smaller collections contain the most relevant documents. There is a strong correlation between measures of model quality and selection performance, indicating the importance of language models and hence size estimates and unbiased document samples.

Finally, with algorithms for server characterisation and selection, and with some assumptions regarding server discovery, query translation, and result merging it is possible to build a simple personal metasearch tool. Research questions remain: how can we evaluate such a tool, and how can we compare one algorithm with another? This thesis has examined several well-used evaluation techniques, but none appear suitable for evaluating tools which work over dynamic and private collections without imposing too great a burden on participants or experimenters. Consequently a new technique has been introduced, based on direct comparisons of whole result sets in the normal search process; this technique allows algorithms to be compared with real information needs, dynamic or private collections, and with regard to a user's context at the time a query is issued. It is lightweight and imposes little overhead on test users or researchers. The technique has been validated in a series of experiments, and demonstrated with a case study comparing techniques for server selection. A case study has confirmed that the evaluation method is useful: it has been possible to compare metasearch techniques *in situ* with little effort, and results of the comparison are broadly but not exactly in agreement with those from a "canned" test collection.

10.2 Building a personal metasearch tool

It is possible, from the experiments reported here, to draw some conclusions on how a working personal metasearch tool might be built.

All sampling techniques tested were poor over the .GOV collection, which has around 1.2 million documents, and it is reasonable to assume they would be similarly poor with larger collections such as Dialog or the public Web. However, collections of this scale are likely to be shared between users and characteristics could be hard-coded (see for example Section E.2.3, which describes models for Wikipedia and for the Web in the PIS prototype). Across smaller collections, likely more characteristic of private data, recent sampling methods including the novel multiple queries method perform well. The multiple queries sampler, in particular, appears appropriate for personal metasearch: it produces samples of a similar quality to the pool-based and random walk samplers, is faster than either, and does not need access to document text. Size estimates built with samples from this method are largely indistinguishable from those built from truly random samples.

Techniques for size estimation rely for the most part on unbiased samples of documents, and biased samples lead to systematic underestimates of size. Given samples from the multiple queries technique, however, the multiple capture-recapture and capture history techniques perform relatively well across a variety of collections and do not need acces to document text. This suggests that either of these techniques could be suitable to use in a personal metasearch tool.

The quality of document samples is again important when estimating language models; so too is the number of documents included. Across different samplers and quality measures, models improve as more documents are added; however, quality improvement does eventually tail off. On the Kullback-Leibler divergence measure, models built with the multiple queries sampler improve until around 400 documents are used, and there are no significant improvements overall past this point. A model built from around 400 documents is practical for a working tool, and this is only slightly more than the 300 commonly used in earlier work.

Experiments with server selection have demonstrated that, to be viable, a technique should not be overly biased by collection size. Several algorithms tested are swayed by large collections. This is likely to be a real problem in personal metasearch, where collections may vary in size by as much as seven orders of magnitude. The Kullback-Leibler divergence and CORI algorithms however appear robust to variation in collection size, and to quality of language models, and are promising for personal metasearch applications.

The multiple queries, multiple capture-recapture, and Kullback-Leibler divergence algorithms have been used to build PIS, a prototype personal metasearch tool. Feedback from test users has been positive, suggesting both that personal metasearch tools could be of real benefit and that the algorithms used in the prototype perform reasonably well in real applications.

10.3 Future work

Work to date suggests future avenues for research in server characterisation and selection.

Techniques for sampling are still effected by query bias, which leads to poor performance in later characterisation and selection. A method which is able to produce representative samples, while making few demands of servers, would be most useful.

Different-sized collections require different-sized document samples for size estimation or for language modelling. Experiments in Chapter 6 here have suggested sample sizes which are appropriate on average, but in many cases a different size may be preferable. Following Baillie et al. [2006a], a technique for choosing an appropriate size — either ahead of time or while sampling — seems worthwhile. There is little work reported on this at present, and none using the wide range of collections of personal metasearch.

Server selection techniques for personal metasearch must work over a large range of collection sizes. As demonstrated in Chapter 7, many existing techniques are biased toward larger collections; those which are not are still performing worse than should be possible. Further work on server selection, and in particular in selection across such a range of collections, would be worthwhile. Algorithms such as returned utility maximisation (RUM) [Si and Callan 2005] and relevant document distribution (RDD) [Voorhees et al. 1994] have not been considered to date, since they require training data in the form of per-document relevance judgements. Additional feedback such as that suggested in Section 8.2.1 could be used to acquire this data, and the algorithms trained on the fly; this possibility is worth further consideration.

This thesis has assumed that servers will not cooperate with a metasearch tool. It seems unlikely that separate providers will cooperate in the manner of STARTS [Gravano et al. 1997a] or Pharos [Dolin et al. 1996]. However it is possible that, for example, an enterprise search tool could cooperate with a metasearch tool for staff inside that enterprise; or that a metasearch tool provided by a whole-of-Web search company could have access to accurate Web statistics. It has not yet been established what sort of benefit would be possible in this case, and whether cooperation would be worthwhile; this is worth considering.

Besides server characterisation and selection, work is needed in related areas if we are to build working tools for personal metasearch. The model of metasearch in Section 2.2.2 suggests several avenues for future work: server discovery, query translation, and result merging have not been considered in the personal metasearch context to date. Result merging has been well-studied in other environments, but not with the heterogenous data likely in a personal application; server discovery and query translation have not yet been considered in depth. Further research on result presentation and user interfaces for metasearch would also be rewarded, as would research towards models and techniques for personalisation.

Most importantly, further work is needed to understand the possible scope and utility of personal metasearch; to characterise the collections likely to be used; and to understand likely uses. This would inform the design of metasearch techniques, and would provide useful background information for evaluations.

10.4 Overall conclusion

Personal metasearch, a novel application of metasearch techniques over all collections a user has access to, is desirable for technical reasons as well as for ease of use. Work remains in some aspects of the metasearch process; but research reported in this thesis has established that personal metasearch is technically feasible, and that a useful, working, tool can be built.

Appendix A

Notation and terminology

This thesis uses the following notation and terminology. Cited material uses the original notation and terminology as far as possible, but has been re-written where necessary for consistency.

A.1 Notation

Collection-, query-, or document-specific terms such as c, $\mathcal{D}$, df, N, s, and $\mathcal{T}$ may carry a subscript to identify a particular collection or document; thus for example N is the size of a particular collection under discussion, and N_i is the size of collection i.

$\emptyset$	The empty set	
α	Rate of type I errors in hypothesis tests; or Laplace smoothing constant for Kullback-Leibler divergence	(p. 82)
B	Burn-in time for random walk sampler	(p. 34)
b	Base score for CORI server selection	(p. 97)
$\mathcal{C}$	All collections known to a metasearch system	(p. 92)
c	A single collection	(p. 92)
$\mathrm{cf}(t)$	Collection frequency: the number of collections which include term t. Equal to $\lvert\{c \in \mathcal{C} : \mathrm{df}_c(t) > 0\}\rvert$	(p. 98)
ctf ratio	Collection term frequency ratio: the proportion of term occurrences accounted for in a model	(p. 79)
$\mathcal{D}$	All documents in some collection	(p. 30)
$\mathcal{D}_{\mathcal{P}}$	All documents from $\mathcal{D}$ covered by a query pool $\mathcal{P}$: equal to $\bigcup_{q\in\mathcal{P}} \mathrm{RES}(q)$	(p. 57)
$\mathrm{D}_{KL}(a\|b)$	Kullback-Leibler divergence between any two distributions a and b	(p. 82)
d	A single document	(p. 30)
$\mathrm{df}(t)$	Document frequency: the number of documents in a collection which include a term t	
$\mathrm{df}(q)$	Document frequency: the number of documents in a collection which match a query q. Equal to $\lvert\mathrm{RES}(q)\rvert$	(p. 55)

(Continued over)

(Continued from previous page)

$\mathrm{df}_m(q)$	Document frequency, from a model m: the number of documents in the model which match a query q	(p. 55)
$\mathrm{E}(X)$	Expected value of some random variable X	
I	Size (inverse document frequency) component for CORI server selection	(p. 97)
k	Result limit: maximum number of documents returned by a server	(p. 30)
λ	Mixing parameter in Kullback-Leibler divergence server selection	(p. 102)
L	Length of a random walk	(p. 35)
l	Minimum similarity for a document to be "interesting" in vGlOSS	(p. 94)
$\textsc{match}_{\mathcal{P}}(d)$	Queries in $\mathcal{P}$ for which document d is a match: equal to $\{q \in \mathcal{P} : d \in \textsc{res}(q)\}$	(p. 33)
M	Number of documents newly seen (marked) in a single sample	(p. 54)
m	Model for some single collection	
m_g	Global model; model for all collections	(p. 102)
$\textsc{merit}_c(q)$	Merit of a collection c; measure of its use with regard to a query q	(p. 115)
N	Number of documents in a collection: equal to $\lvert\mathcal{D}\rvert$	(p. 51)
o	Overlap between a sample and previously-seen documents	(p. 52)
π_1	Hypothesised success rate for binomial sign tests	
$\mathcal{P}$	Query pool	(p. 33)
$\mathcal{P}+$	Subset of queries from pool $\mathcal{P}$ which neither over- nor underflow with regard to a given server: equal to $\{q \in \mathcal{P} : 0 < \lvert\textsc{res}(q)\rvert < k\}$	(p. 33)
$\Pr(x)$	Probability of some event x	
p_1	Probability of success: parameter of a geometric distribution	(p. 38)
$\mathcal{Q}$	Second query pool	(p. 58)
q	A single query	(p. 30)
$\hat{R}$	Mixing parameter for Ponte and Croft's unigram language models	(p. 77)
$\mathcal{R}_n$	Proportion of merit captured by highest-ranked n servers	(p. 115)
r	Result limit for the query-based sampler: number of top-ranked documents downloaded and added to the model	(p. 32)
r_s	Spearman's rank correlation coefficient	(p. 80)
$\mathrm{rdf}(t)$	Normalised (relative) document frequency of term t	(p. 92)
$\textsc{rel}(q)$	Relevant documents for query q	(p. 104)

(Continued over)

(Continued from previous page)

$\mathrm{RES}(q)$	Result set: documents returned for query q by a given server	(p. 30)
$\mathrm{rtf}(t)$	Normalised (relative) term frequency of term t	(p. 92)
$\mathcal{S}$	Set of servers known to a metasearch system	(p. 9)
$s_c(q)$	Score of collection c for query q; assigned by a server selection algorithm	(p. 92)
s_q	Number of successful queries used per run of the multiple queries sampler	(p. 39)
$\mathrm{SIM}_c(q, i)$	Similarity to a query q of documents in c containing the i most common query terms	(p. 95)
$\mathcal{T}$	Unique terms in a collection	(p. 79)
T	Term-specific (term frequency) component for CORI server selection	(p. 97)
t	An individual term	
$\mathrm{tf}(t)$	Term frequency of t: total number of occurences of t in the collection	(p. 79)
$\mathrm{VDENSITY}(d)$	Validity density: proportion of queries covering a document d which neither over- nor underflow. Equal to $\lvert\mathrm{MATCH}_{\mathcal{P}+}(d)\rvert / \lvert\mathrm{MATCH}_{\mathcal{P}}(d)\rvert$	(p. 38)
$\mathrm{VDENSITY}(\mathcal{P})$	Validity density: proportion of queries from a pool $\mathcal{P}$ which neither over- nor underflow. Equal to $\lvert\mathcal{P}+\rvert / \lvert\mathcal{P}\rvert$	(p. 38)
$w_c(t)$	Weight of a term t in collection c	(p. 94)
$w_q(t)$	Weight of a term t in query q	(p. 95)

A.2 Terminology

abject failure Descriptive of a result set which is poor enough that no results appear reasonable, and which therefore will have no clickthrough data recorded.

brilliant success Descriptive of a result set which is good enough to answer a query immediately (for example, in document summaries), and which therefore will have no clickthrough data recorded.

collection A set of documents kept at the same repository: for example all pages in a web site, all email in an archive, or all records in a database.

cooperative A "cooperative" server includes features useful for metasearch tools as well as a standard query interface. For example, a cooperative server may report collection size or term statistics. An "uncooperative" server exposes only a query interface.

cover A query "covers" a document if the document matches the query; it covers a set of documents if all documents in the set match. A query pool "covers" all those documents covered by at least one query in the pool.

distributed information retrieval Used interchangably with "metasearch".

document The unit of retrieval: for example a web page, email message, or database record.

federated search Used interchangably with "metasearch".

metasearch Search over several independent document collections, each of which may be accessible through its own server.

overflow A query q "overflows" if the size of the result set is constrained by some maximum k, and not by the number of matching documents (so $|\textsc{res}(q)| \geq k$).

query bounce Re-running a query, for example in a graphical interface by clicking a control twice.

server A local or networked application which provides search capabilities for a collection.

underflow A query q "underflows" if it produces no results (so $|\textsc{res}(q)| = 0$).

web, Web A collection of documents generally available via HTTP. The capitalised "Web" is used for the publicly-available world-wide Web.

Appendix B

Instruments for searcher survey

Participants in the survey described in Section 3.3 were recruited by email to mailing lists and professional forums.[1] Interested parties were directed to a Web site where they were asked whether they consented to participate (email addresses and telephone numbers have been removed):

> *This survey is being carried out as part of my PhD research at the Department of Computer Science in the Australian National University. The main purpose of the survey is to obtain an insight into what sorts of computerised information resources people use, and how people work with them. This should help researchers design future search software which better meets people's needs.*
>
> *Completing this survey will take about 10–20 minutes and all questions are optional. You will be asked some initial questions about yourself and your experience with computers, then asked specific questions about the sort of computerised information resources you use from day to day. We will record your answers and analyse the results to guide our research.*
>
> ***Privacy statement***
>
> ***Security of the website***
>
> *Users should be aware that the World Wide Web is an insecure public network that gives rise to a potential risk that a user's transactions are being viewed, intercepted or modified by third parties or that data which the user downloads may contain computer viruses or other defects.*
>
> ***Security of the data***
>
> *The data will be kept secure on a password-protected computer for the length of the project, and may be kept for the length of my PhD research. We will not publish any information which could link you to the survey or to any particular response. As the web can be an insecure medium you may choose to complete this survey by fax or post; please contact us directly.*

[1] The experiments described here were approved by the Australian National University Human Research Ethics Committee as protocol 2006/162.

> ***Purpose of data collection***
>
> *This information is being sought for a research project entitled "Understanding digital information sources". The researcher is Paul Thomas, in the Department of Computer Science at the Australian National University. The information you provide will only be used for the purpose for which you have provided it. It will not be disclosed without your consent.*
>
> *If you have any questions, comments, or complaints please contact the researcher, Paul Thomas, at — or on — (x— on the ANU campus). You may also contact my supervisor, Tom Gedeon, at — or on —. If you have any questions regarding your rights as a research participant, please contact the Australian National University's Human Research Ethics Committee at — or on —.*
>
> *Thank you for your participation.*

Consenting participants were then given a survey in three parts. The first part asked:

> *In this survey we are interested in any computer-based information sources you use in your job. Some examples are:*
>
> - *The web as a whole (e.g. a search with Google, Yahoo, etc.)*
> - *Individual websites*
> - *Corporate databases*
> - *Online catalogues, for example library holdings or journals*
> - *Subscription services like share quotes or company analyses*
> - *Dictionaries or thesauri*
> - *Personal email or mailing lists*
> - *Calendars and diaries*
> - *Bulletin boards*
> - *Files on your local computer or in a shared folder*
> - *Corporate intranets or extranets*
> - *Resources made available by customers or suppliers*
>
> *All the questions in this survey are optional, but the more information you are able to give the more help it will be to us. To move on to the next part of the survey, click on "Continue".*
>
> 1. *What is your occupation?*
> 2. *Could you please describe one or more information/knowledge intensive tasks that you perform?*
> 3. *What computer-based information sources do you use in the course of your job? Please list as many as you can think of.*

  - *My own Word or Excel documents (or similar)*
  - *Documents shared amongst my section*
  - *Documents shared amongst my organisation*
  - *A local intranet*
  - *The web*
  - *Archived email*
  - *Others (please specify)*

4. *When do you tend to turn to a computerised source, in preference to other ways to get information?*
5. *Are there circumstances that make you likely to give up altogether on a search for information?*

The second part of the survey, which could be repeated any number of times, asked for details on one or more of the sources identified earlier.

> *Now we would like to know about each individual source you listed above. Here we're asking for details of any one source; when you have answered the questions for this source, please choose either "Continue and describe another source" if you are willing to describe another of the sources you use, or "Continue to the next part of the survey" to move on. We would be grateful if you were able to describe several sources.*
>
> *To jog your memory, the sources you mentioned earlier were: ...*
>
> *You can answer for any one of these.*

1. *What is this source? Please briefly name or describe it.*
2. *How often do you use this source?*
   - *Lots (use daily)*
   - *Some (use weekly)*
   - *Little (use less than weekly)*
   - *Rather not say*
3. *What sort of things are you looking for when you use this source? Can anything found here also be found elsewhere?*
4. *Do you or your organisation pay an ongoing subscription, and/or per-use fees, to access this source?*
5. *Does this source offer a search facility of any kind?*
6. *How often are you able to find what you're looking for?*
7. *Are there any times when you're especially likely to find what you're looking for, or when you're especially likely to give up?*

The final part of the survey had some general questions.

> *Thank you for your responses so far. Some final questions:*
>
> 1. *Do you think it would be useful to be able to search all or some of the sources you've identified via a single search interface? Any detail you can provide would be most helpful.*
> 2. *Do you have any other comments on how you work with these computerised sources, or how you'd like to if it were possible?*
> 3. *If you would like to know more about this project, or are willing to be contacted for further information, how can we contact you? Please note that this is entirely optional.*
>
> *That's the end of the survey. Thanks for your time. If you are interested in finding out more about this survey or the project it's part of, more information is available.*

The underlined text linked to a page describing the overall personal metasearch project.

Participants' sessions were tracked by assigning a random identifier as they entered the first part of the survey; this identifier was maintained in a hidden HTML form field [Raggett et al. 1999].

Appendix C

Instruments for evaluation experiments

Participants in the experiments described in Chapter 8 were recruited by email to public mailing lists and by posters on noticeboards at the Australian National University.[1] Interested parties were directed to a Web site where they were asked whether they consented to participate (email addresses and telephone numbers have been removed):

> *This experiment is being carried out as part of my PhD research at the Department of Computer Science in the Australian National University. The main purpose of the experiment is to obtain an insight into what sorts of search results people find most useful, and to learn more about ways to evaluate search engines for future work. This should help researchers design future search software which better meets people's needs.*
>
> *Participation in this project can take as long or as short a time as you like. You will be asked some initial questions about yourself and your experience with computers, then asked to carry out your normal web searches using software we provide. We will record your interactions with our software and analyse the results to guide our research.*
>
> *Results from this project may be published in a research forum, and used in my PhD thesis. No personal information will be published except in aggregate form (such as averages or totals). We will not publish any information which could link you to the experiment or to and particular search or web page, and any information you provide will only be used for the purpose for which you have provided it. All information will be protected to the greatest extent allowed by law, and data will be kept secure on a password-protected computer during and after the project.*
>
> *Your participation in this research is entirely voluntary. You are welcome to withdraw at any time, even after finishing the search tasks, and there will be no penalty whatsoever.*

[1] The experiments described here were approved by the Australian National University Human Research Ethics Committee as protocol 2005/326.

> *If you have any questions, comments, or complaints please contact the researcher, Paul Thomas, at — or on — (x— on the ANU campus). You may also contact my supervisor, Tom Gedeon, at — or on —. If you have any questions regarding your rights as a research participant, please contact the Australian National University's Human Research Ethics Committee at — or on —.*
>
> *Thank you for your participation.*

Consenting participants were then presented with instructions appropriate to the current experiment (Sections C.2–C.3), and asked to choose a user name. This was used to distinguish users in the logs, but not to identify any individual.

> *Thank you for being part of this experiment. To help us distinguish each participant, please enter a user name. This can be any name you like, and need not be your real name or anything else identifiable. We will not record who chooses which name, so you can remain entirely anonymous; however, we would appreciate it if you used the same name each time you participate in this experiment.*
>
> *If at a later time you decide to withdraw from the experiment, or you would prefer that we delete some of the data we have collected from your searches, please let us know your user name and we can update our records. This is the only situation in which we need to know the name you have chosen.*

An HTTP cookie [Kristol and Montulli 2000] was used to record the chosen name.

C.1 Demographic questions

Before their first search, each user in each experiment was asked a number of demographic questions. Each had a "rather not say" option. Participants were asked:

1. Their sex;
2. Their age;
3. Their level of education: school, first degree, postgraduate degree, or other tertiary qualification;
4. Their experience with computers: daily use, occasional use, some past use, or no prior use;
5. The number of years they had been using computers;
6. Their experience with the Web;
7. The number of years they had been using the Web;
8. Their experience with Web search engines; and
9. The number of years they had been using Web search engines.

Users were then presented instructions and a search interface appropriate to the particular experiment they were recruited for.

C.2 First experiment

The first experiment (Section 8.3.1) used topics selected ahead of time, from popular queries submitted to Google. Volunteers were given the following instructions:

> *Thank you for participating in this experiment. There are three parts; the whole process should take no more than an hour of your time. The software you are given will provide details as you go.*
>
> *First, you will be asked to choose a user name. This can be any name you like, and need not be your real name or anything else identifiable. We will not record who chooses which name, so you can remain entirely anonymous.*
>
> *Next, you will be asked some questions about yourself and your experience with computers and web search engines. If you would rather not answer any of the questions, feel free to select the "rather not say" option.*
>
> *Finally, you will be given some search topics. For each topic, please type one or more words into the search box, the same way you would for your usual web search, and click "Search" or press enter. Where a normal web search only gives you one set of results, however, our software gives you two; please take a look at the pages in each set of results and indicate which set, if either, you think is better. You should feel free to use different search words, or more than one search per topic, if you think this will provide better web pages.*
>
> *You shouldn't need to spend more than five minutes on any topic.*
>
> *For your list of topics, please see overleaf. If you have any questions, please don't hesitate to let the researchers know: you can also try emailing — or telephoning — (x— on the ANU campus).*

Topics were in ten sets of ten. Set #1 is illustrative:

1. 50 Cent
2. Australian Idol
3. Chad Michael Murray (an actor)
4. Dublin bus
5. Holiday
6. Japanese restaurant
7. Lost
8. NZdating
9. Rockstar INXS
10. Trading Post

C.3 Later experiments

Participants in later experiments (Sections 8.3.2–8.3.4) were encouraged to use the tool for naturally-occurring topics. In these experiments the following instructions were issued before the first search:

> *That's all the background information we need. This now works like a normal web search engine: you can type your query in the box below, and click "Search" (or just press enter) to get results from the web. Each time, you will be given two sets of results and we'll ask which (if any) is better. If you are able to answer this question, it'll be of great help, but it's not compulsory.*

The last two sentences were not included when explicit judging was removed in experiment four (Section 8.3.4).

Appendix D

Instruments for server selection evaluation

Participants in the experiments described in Chapter 9 were recruited by email from previous participants who had expressed interest in followup work.[1] Interested parties were directed to a Web site where they were asked whether they consented to participate (email addresses and telephone numbers have been removed):

> *This experiment is being carried out as part of my PhD research at the Department of Computer Science in the Australian National University. The main purpose of the experiment is to obtain an insight into how best to search several different data sources (such as the web, email, and calendars) at once. This should help researchers design future search software which better meets people's needs.*
>
> *Participation in this experiment can be as long or as short as you like. We invite you to use our pilot search software instead of your normal search tools; after a few minutes to set it up it should be able to replace a number of more specialised tools for searching files, email, the web, calendars, individual websites, etc.*
>
> ***What information are we collecting?***
>
> *We will record your interactions with our software and analyse the results to guide our research, but* we will not record any personal or private data and the software will not divulge, even to us, the searches you make or the data it is searching. *If you prefer, however, you can of course choose not to use our software for sensitive data or searches. You can also check from the software itself what information is being recorded.*
>
> *Results from this experiment may be published in a research forum, and used in my PhD thesis. Information will only be published in aggregate form (such as averages or totals), and no personal information will be collected or published. We will not publish any information which could link you to the experiment or to any particular search or set of data, and any information you provide will only be used for the purpose for which you have provided it. All information will be protected to*

[1] The experiments described here were approved by the Australian National University Human Research Ethics Committee as protocol 2007/163.

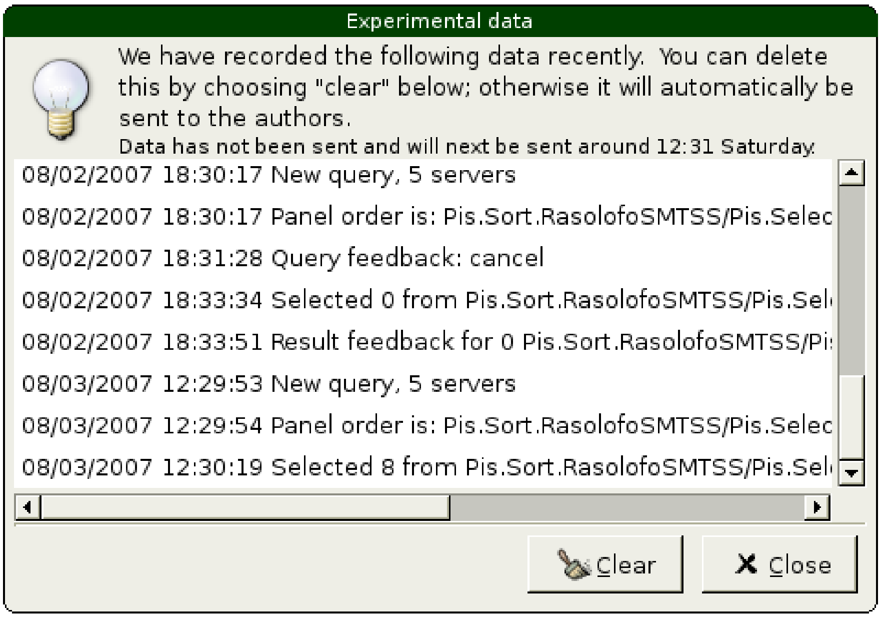
> *the greatest extent allowed by law, and will be kept secure on a password-protected computer during and after the project.*
>
> ***What is this software?***
>
> *Our "personal information search" software provides a search tool which can find information in your files, email archives, local web sites, calendars, address books, on the world-wide web, and in other sources using a single program. It can search all these sources simultaneously, and can automatically choose where to look each time.*
>
> *Depending on what you choose to search, the software may write one or more "indexes" to your local hard drive. It may also scan your email, calendars, etc., to provide its search function. Every care has been taken to ensure it will not damage your data, and versions of this software have been trialled by the author for several months before being made available to you. Of course, if you have any concerns we recommend you maintain an up-to-date backup and run a virus checker.*
>
> *Please note that* this is experimental software, rather than a commercial product. *Although we have taken all reasonable care when writing and testing the software, unfortunately* we are not able to provide any warranty *that it will work as advertised or that it will not result in some adverse impact in your particular environment, nor any compensation.*
>
> *Please also note that if you configure the software to search commercial databases to which you are subscribed and which charge per minute, per search, or per document retrieved, you may incur costs when the software searches these databases on your behalf. Feel free to contact the researcher for more information.*
>
> ***What are my rights?***
>
> *Your participation in this research is entirely voluntary. You are welcome to withdraw at any time, and there will be no penalty whatsoever.*
>
> *If you have any questions, comments, or complaints please contact the researcher, Paul Thomas, at — or on —. You may also contact my supervisor, Tom Gedeon, at — or on —. If you have any questions regarding your rights as a research participant, please contact the Australian National University's Human Research Ethics Committee at — or on —.*
>
> *Thank you for your participation.*

Consenting participants were then offered a copy of the PIS software described in Section 3.4 and Appendix E. The software was configured with two panels to allow the evaluation: one panel used Kullback-Leibler divergence (Section 7.1.6) for server selection, the other used vGlOSS (Section 7.1.1). Both panels used SM-TSS [Rasolofo et al. 2003] to merge results from multiple sources. PIS was also configured to ask for feedback as described in Section 9.1. Users were able to add or remove any collections desired, but could not modify or remove either panel.

PIS logged a small set of data for each user query: the fact that a query had been issued, the ordering of the panels for this query, the panel and rank of any result

```
*** user Thu Aug 9 18:43:24 2007
08/09/2007 15:30:02 New query, 10 servers
08/09/2007 15:30:02 Panel order is:  Pis.Sort.RasolofoSMTSS/-
Pis.Select.VGlOSSSelect Pis.Sort.RasolofoSMTSS/-
Pis.Select.KLSelect
08/09/2007 15:31:06 Selected 5 from Pis.Sort.RasolofoSMTSS/-
Pis.Select.KLSelect
08/09/2007 15:31:20 Result feedback for 5
Pis.Sort.RasolofoSMTSS/Pis.Select.KLSelect:  helpful
```

Figure D.1: Typical data logged for a single query in PIS. "User" is a system-assigned unique identifier.



We have recorded the following data recently. You can delete this by choosing "clear" below; otherwise it will automatically be sent to the authors.

Data has not been sent and will next be sent around 12:31 Saturday.

```
08/02/2007 18:30:17 New query, 5 servers
08/02/2007 18:30:17 Panel order is: Pis.Sort.RasolofoSMTSS/Pis.Selec
08/02/2007 18:31:28 Query feedback: cancel
08/02/2007 18:33:34 Selected 0 from Pis.Sort.RasolofoSMTSS/Pis.Sel
08/02/2007 18:33:51 Result feedback for 0 Pis.Sort.RasolofoSMTSS/Pi
08/03/2007 12:29:53 New query, 5 servers
08/03/2007 12:29:54 Panel order is: Pis.Sort.RasolofoSMTSS/Pis.Selec
08/03/2007 12:30:19 Selected 8 from Pis.Sort.RasolofoSMTSS/Pis.Sel
```

Figure D.2: Reviewing logged data in PIS. Data was automatically sent every 48 hours, unless deleted by the participant.

opened, and any additional feedback. Figure D.1 has an example. Users were given the opportunity to review the accumulated data before it was sent, and to delete it, as illustrated in Figure D.2; otherwise it was automatically sent to a Web-based recorder at roughly 48-hour intervals.

D.1 Sorting with SM-TSS

PIS used the SM-TSS algorithm of Rasolofo et al. [2003] to re-sort and merge results from different collections. The algorithm scores according to the number of words in common between a document's title and the query, normalised by the length of each; it is summarised in Figure D.3.

```
SM-TSS_score ($d$, $q$):
  if ($d$ has a title):
        $score \leftarrow$ text_score(title of $d$, $q$)
  else if ($d$ has a summary):
        $score \leftarrow$ text_score(summary of $d$, $q$)
  else:
        $score \leftarrow$ rank score($d$)
  return $score$

Text_score ($text$, $q$):
  $common \leftarrow |text \cap q|$
  $norm \leftarrow \sqrt{|text|^2 + |q|^2}$
  $score \leftarrow 100,000 \times common/norm$
  return $score$

Rank_score ($d$):
  $score \leftarrow 1000-$ rank of $d$
  return $score$
```

Figure D.3: The SM-TSS algorithm [Rasolofo et al. 2003], as used in PIS.

Appendix E

Software

PIS, a *p*ersonal *i*nformation *s*earcher, is a working personal metasearch tool which has been used in the experiments of Chapter 9. It operates with the hybrid model described in Section 2.1.5 on p. 10, acting as a front-end to servers where available and doing its own indexing otherwise. It implements each of the stages in the metasearch process of Section 2.2.2, with the exception of server discovery, and includes implementations of several algorithms for server characterisation, server selection, and result merging.

Figure E.1 illustrates PIS searching a number of collections and returning a single ranked list, including pieces of email, entries in a library catalogue, BibTeX records, and results from Project MUSE.[1]

PIS represents around 15,000 lines of C# code, plus 42,000 in included third-party libraries, and has been tested on Linux (with the Mono .NET interpreter[2]) and on Windows.

E.1 Interface

As a hybrid metasearcher, PIS offers a single interface to any number of independent search engines and to any number of collections indexed by PIS itself (Figure E.1).

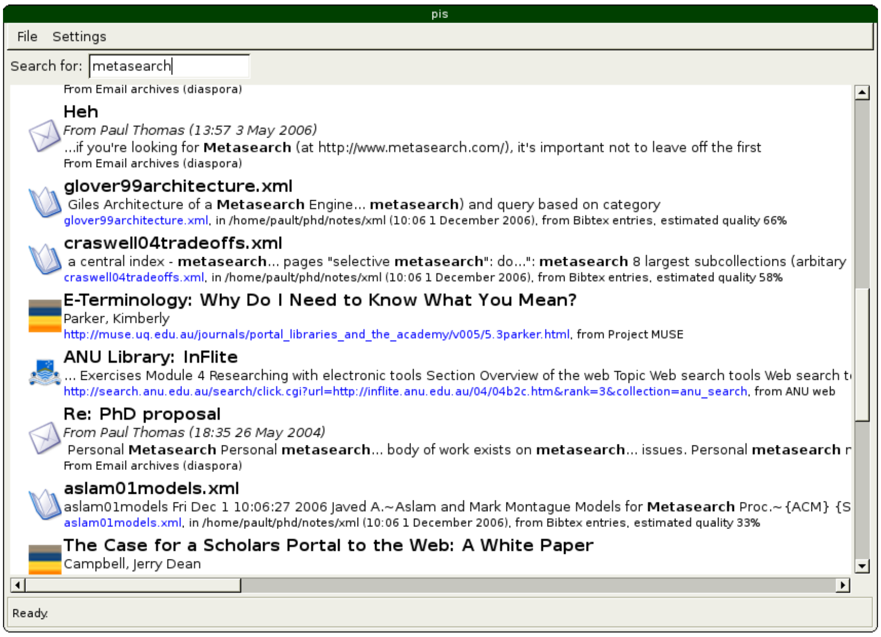
To support embedded comparisons of the type discussed in Chapters 8 and 9, PIS can display results in two or more independent panels, each with its own algorithm for server selection and result sorting and merging (Figure E.2). It also implements the additional feedback options described in Section 8.2.1, which allows experimenters to distinguish brilliant success from abject failure and provides confirmation of implicit judgments. These extra feedback features are illustrated in Figure E.3.

E.2 Modules

PIS uses plug-in modules to provide key metasearch features. The present set of modules are described below.

[1] `http://muse.jhu.edu/`

[2] `http://www.mono-project.org/`



metasearch

From Email archives (diaspora)

Heh
From Paul Thomas (13:57 3 May 2006)
...if you're looking for **Metasearch** (at http://www.metasearch.com/), it's important not to leave off the first
From Email archives (diaspora)

glover99architecture.xml
Giles Architecture of a **Metasearch** Engine... **metasearch**) and query based on category
glover99architecture.xml, in /home/pault/phd/notes/xml (10:06 1 December 2006), from Bibtex entries, estimated quality 66%

craswell04tradeoffs.xml
a central index - **metasearch**... pages "selective **metasearch**": do...": **metasearch** 8 largest subcollections (arbitary
craswell04tradeoffs.xml, in /home/pault/phd/notes/xml (10:06 1 December 2006), from Bibtex entries, estimated quality 58%

E-Terminology: Why Do I Need to Know What You Mean?
Parker, Kimberly
http://muse.uq.edu.au/journals/portal_libraries_and_the_academy/v005/5.3parker.html, from Project MUSE

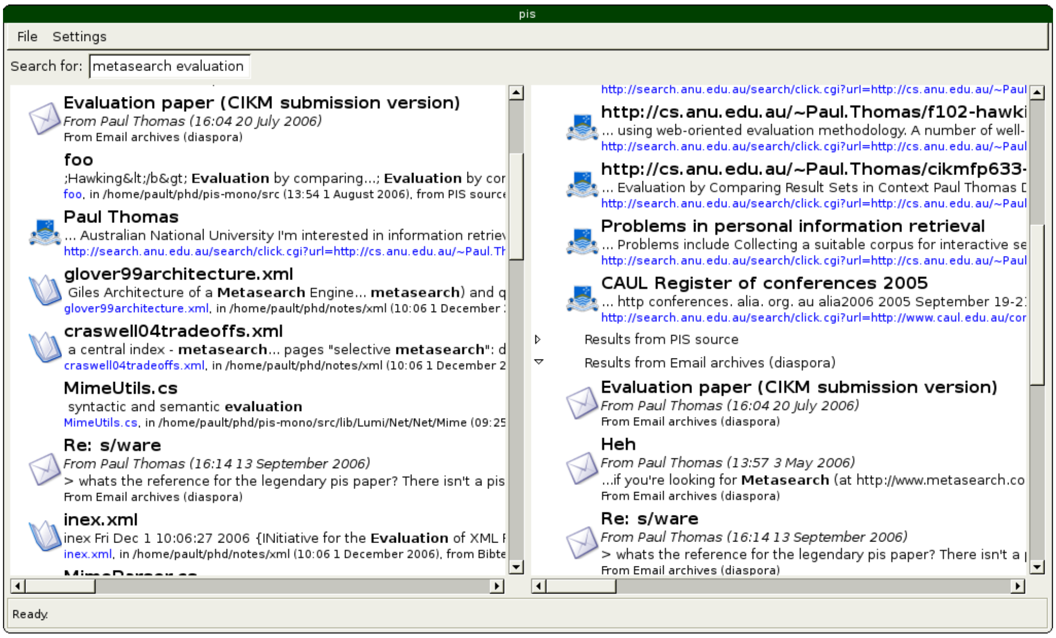
ANU Library: InFlite
... Exercises Module 4 Researching with electronic tools Section Overview of the web Topic Web search tools Web search t
http://search.anu.edu.au/search/click.cgi?url=http://inflite.anu.edu.au/04/04b2c.htm&rank=3&collection=anu_search, from ANU web

Re: PhD proposal
From Paul Thomas (18:35 26 May 2004)
Personal **Metasearch** Personal **metasearch**... body of work exists on **metasearch**... issues. Personal **metasearch** r
From Email archives (diaspora)

aslam01models.xml
aslam01models Fri Dec 1 10:06:27 2006 Javed A.~Aslam and Mark Montague Models for **Metasearch** Proc.~{ACM} {S
aslam01models.xml, in /home/pault/phd/notes/xml (10:06 1 December 2006), from Bibtex entries, estimated quality 33%

The Case for a Scholars Portal to the Web: A White Paper
Campbell, Jerry Dean

Figure E.1: The PIS software.



metasearch evaluation

Evaluation paper (CIKM submission version)
From Paul Thomas (16:04 20 July 2006)
From Email archives (diaspora)

foo
;Hawking</b> **Evaluation** by comparing...; **Evaluation** by cor
foo, in /home/pault/phd/pis-mono/src (13:54 1 August 2006), from PIS source

Paul Thomas
... Australian National University I'm interested in information retriev
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul.Th

glover99architecture.xml
Giles Architecture of a **Metasearch** Engine... **metasearch**) and q
glover99architecture.xml, in /home/pault/phd/notes/xml (10:06 1 December :

craswell04tradeoffs.xml
a central index - **metasearch**... pages "selective **metasearch**": d
craswell04tradeoffs.xml, in /home/pault/phd/notes/xml (10:06 1 December 2

MimeUtils.cs
syntactic and semantic **evaluation**
MimeUtils.cs, in /home/pault/phd/pis-mono/src/lib/Lumi/Net/Net/Mime (09:25

Re: s/ware
From Paul Thomas (16:14 13 September 2006)
> whats the reference for the legendary pis paper? There isn't a pis
From Email archives (diaspora)

inex.xml
inex Fri Dec 1 10:06:27 2006 {INitiative for the **Evaluation** of XML F
inex.xml, in /home/pault/phd/notes/xml (10:06 1 December 2006), from Bibte

http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul

http://cs.anu.edu.au/~Paul.Thomas/f102-hawki
... using web-oriented evaluation methodology. A number of well-
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul

http://cs.anu.edu.au/~Paul.Thomas/cikmfp633-
... Evaluation by Comparing Result Sets in Context Paul Thomas D
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul

Problems in personal information retrieval
... Problems include Collecting a suitable corpus for interactive se
http://search.anu.edu.au/search/click.cgi?url=http://cs.anu.edu.au/~Paul

CAUL Register of conferences 2005
... http conferences. alia. org. au alia2006 2005 September 19-21
http://search.anu.edu.au/search/click.cgi?url=http://www.caul.edu.au/cor

▷ Results from PIS source

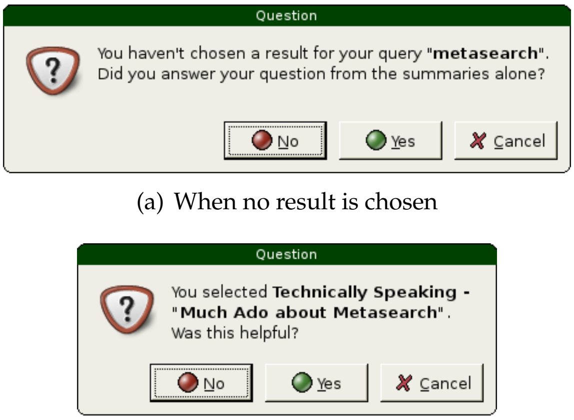
▽ Results from Email archives (diaspora)

Evaluation paper (CIKM submission version)
From Paul Thomas (16:04 20 July 2006)
From Email archives (diaspora)

Heh
From Paul Thomas (13:57 3 May 2006)
...if you're looking for **Metasearch** (at http://www.metasearch.co
From Email archives (diaspora)

Re: s/ware
From Paul Thomas (16:14 13 September 2006)
> whats the reference for the legendary pis paper? There isn't a
From Email archives (diaspora)

Figure E.2: PIS with two panels, for evaluations in the manner of Chapters 8 and 9.



Question

You haven't chosen a result for your query "**metasearch**". Did you answer your question from the summaries alone?

(a) When no result is chosen

Question

You selected **Technically Speaking - "Much Ado about Metasearch"**. Was this helpful?

(b) When a result has been chosen

Figure E.3: Extra feedback from PIS.

E.2.1 Filters

Filters are responsible for parsing documents of various formats (typically from local files, but possibly from a networked resource) and returning indexable content such as text or title. At present PIS has filters for plain text documents; HTML [Raggett et al. 1999]; XML [Bray et al. 2006]; PDF;[3] several programming languages; and images.

E.2.2 Sampling and size estimation

At present there is only one sampling plug-in, which implements the multiple queries algorithm (Section 4.3) with result cut-off $k = 10,000$ and query terms drawn from a set of common English words. This was the most promising sampler in experiments in this thesis (Section 4.5).

Similarly, only the only size estimation technique used at present is multiple capture-recapture [Shokouhi et al. 2006], using 300 to 1000 samples of up to 10 documents each. The algorithm has been modified slightly to allow non-uniform sample sizes, to request more documents if no samples overlap, and to offer a best guess at collection size in the event that none of the eventual samples overlap.

E.2.3 Models

Modelling plug-ins provide techniques for working with unigram language models of the kind described in Section 6.1.2. Separate plug-ins implement simple models read from a plain-text file; models, useful for collections indexed by PIS itself, which get term occurrence data from a local index; a special model for Wikipedia, which is based on all article titles in an August 2007 snapshot; and a special model for the world-wide Web, which is based on term frequencies from a number of large-scale

[3] http://www.adobe.com/devnet/pdf/pdf_reference.html

Web crawls including the .GOV collection from the TREC Web Track and a crawl of the `.au` top-level domain [Ackland et al. 2007].

E.2.4 Server selection

Four modules perform server selection. Two provide baseline algorithms: a trivial "select all" technique and random scoring. A third implements Kullback-Leibler divergence [Si et al. 2002; Xu and Croft 1999], which was one of the most promising techniques examined in Chapter 7; it uses smoothing with a background model with $\lambda = 0.5$, and estimates term frequencies from a model. The fourth implementation is of vGlOSS [Gravano and García-Molina 1995], using the $\text{SUM}(0)$ scoring formula and with weights as described in Section 7.3.1.

E.2.5 Search

A number of modules provide search capabilities. PIS assumes a hybrid model of the kind discussed in Section 2.1.5, so modules can either communicate with a search engine or provide their own indexing and searching. Modules provide search for:

- *Addressbooks*, in standard (vCard) format,[4] and for the KDE[5] addressbook in particular;
- *Calendars*, in standard iCalendar format [Dawson and Stenerson 1998] and the KDE calendar in particular;
- Local *email*, in maildir,[6] mbox [Hall 2005], nnml, or babyl format;
- Remote *email*, via IMAP [Crispin 2003];
- *LDAP directories* [Zeilenga 2006];
- The *Web*, via Google[7] and Yahoo!;[8]
- *Local files*, in any format for which there is a filter (Section E.2.1);
- *Databases* available via PostgreSQL;[9]
- *Bibliographic* and *reference sources*, including Wikipedia,[10] the library of the Australian National University,[11] Project MUSE, and WorldCat;[12]

[4] `http://www.imc.org/pdi/vcard-21.txt`

[5] `http://www.kde.org/`

[6] `http://www.qmail.org/man/man5/maildir.html`

[7] `http://code.google.com/`

[8] `http://developer.yahoo.com/search/`

[9] `http://www.postgresql.org/`

[10] `http://www.wikipedia.org/`

[11] `http://library.anu.edu.au/`

[12] `http://www.worldcat.org/`

- Any service with a web interface. Built-in examples are del.icio.us[13] and the public websites of the Australian National University[14] and the CSIRO.[15]
- The output of any external search software, in an XML-based interchange format.

E.2.6 Sorters

Finally, five different algorithms are available for sorting and merging the results from the selected search engines. Trivial algorithms provide first-in-first-out "sorting" and random "sorting"; PIS also includes an algorithm which sorts results first by collection then by server-assigned score, and an implementation of Rasolofo et al.'s SM-TSS algorithm [2003]. A final algorithm removes any results from a specified collection before passing the remainder to another sorter.

[13] `http://del.icio.us/`

[14] `http://www.anu.edu.au/`

[15] `http://www.csiro.au/`

Bibliography

ABBACI, F., SAVOY, J., AND BEIGBEDER, M. 2002. A method for collection selection in heterogenous contexts. In *Proc. IEEE Conference on Information Technology* (2002), pp. 529–535. (pp. 10, 91)

ACKLAND, R., SPINK, A., AND BAILEY, P. 2007. Characteristics of .au websites: An analysis of large-scale web crawl data from 2005. In *Proc. 13th Australasian World Wide Web Conference (AusWeb07)* (2007). (p. 188)

ADAR, E., KARGER, D., AND STEIN, L. A. 1999. Haystack: Per-user information environments. In *Proc. CIKM* (1999), pp. 413–422. (p. 20)

AGICHTEIN, E., BRILL, E., AND DUMAIS, S. 2006. Improving web search ranking by incorporating user behaviour information. In *Proc. ACM SIGIR* (2006), pp. 19–26. (p. 9)

ALLAN, J., CARTERETTE, B., AND LEWIS, J. 2005. When will information retrieval be "good enough"? In *Proc. ACM SIGIR* (2005), pp. 433–440. (pp. 139, 160)

ANAGNOSTOPOULOS, A., BRODER, A. Z., AND CARMEL, D. 2005. Sampling search-engine results. In *Proc. WWW* (2005), pp. 245–256. (p. 56)

ANSI/NISO. 2003. Z39.50-2003 information retrieval: Application service definition and protocol specification. (p. 11)

AVRAHAMI, T. T., YAU, L., SI, L., AND CALLAN, J. 2006. The FedLemur project: Federated search in the real world. *JASIST 57*, 3, 347–358. (pp. 9, 10, 16, 56)

AZZOPARDI, L., BAILLIE, M., AND CRESTANI, F. 2006. Adaptive query-based sampling for distributed IR. In *Proc. ACM SIGIR* (2006), pp. 605–606. Poster. (p. 16)

BAEZA-YATES, R. AND CASTILLO, C. 2007. Crawling the infinite web. *Journal of Web Engineering 6*, 1, 49–72. (p. 8)

BAILEY, N. T. J. 1951. On estimating the size of mobile populations from recapture data. *Biometrika 38*, 3/4, 293–306. (p. 53)

BAILEY, P., THOMAS, P., AND HAWKING, D. 2007. Does brandname influence perceived search result quality? Yahoo!, Google, and WebKumara. Submitted to *Australasian Document Computing Symposium*. (p. 154)

BAILLIE, M., AZZOPARDI, L., AND CRESTANI, F. 2006a. Adaptive query-based sampling of distributed collections. In *Proc. SPIRE*, Number 4209 in Lecture Notes in Computer Science (2006), pp. 316–328. (pp. 32, 40, 48, 79, 124, 166)

BAILLIE, M., AZZOPARDI, L., AND CRESTANI, F. 2006b. Towards better measures: Evaluation of estimated resource description quality for distributed IR. In *Proc. First*

International Conference on Scalable Information Systems (2006). (pp. 78, 79, 80, 82, 89)

BAILLIE, M., CRESTANI, F., AND LANDONI, M. 2006. PENG: Integrated search of digital news archives. In *Proc. ACM SIGIR* (2006), pp. 607–608. Poster. (p. 16)

BALMIN, A., HRISTIDIS, V., AND PAPAKONSTANTINOU, Y. 2004. ObjectRank: Authority-based keyword search in databases. In *Proc. VLDB* (2004), pp. 564–575. (p. 140)

BAR-YOSSEF, Z., BERG, A., CHIEN, S., FACKCHAROENPHOL, J., AND WEITZ, D. 2000. Approximating aggregate queries about web pages via random walks. In *Proc. VLDB* (2000), pp. 535–544. (pp. 31, 36, 40)

BAR-YOSSEF, Z. AND GUREVICH, M. 2006. Random sampling from a search engine's index. In *Proc. WWW* (2006), pp. 267–376. (pp. 30, 31, 33, 34, 35, 38, 40, 60)

BARBOSA, L. AND FREIRE, J. 2005. Searching for hidden-web databases. In *Proc. Eighth International Workshop on the Web and Databases (WebDB)* (2005), pp. 1–6. (p. 12)

BARREAU, D. AND NARDI, B. A. 1995. Finding and reminding: File organization from the desktop. *ACM SIGCHI Bulletin 27*, 3. (pp. 5, 135)

BAUMGARTEN, C. 1999. A probabilistic solution to the selection and fusion problem in distributed information retrieval. In *Proc. ACM SIGIR* (1999), pp. 246–253. (pp. 9, 13, 91)

BEIGI, M., BENITEZ, A. B., AND CHANG, S.-F. 1998. MetaSEEk: A content-based meta-search engine for images. In *Proc. 1998 SPIE Conference on Storage and Retrieval for Image and Video Databases VI*, Volume 3312 of *Proceedings of SPIE* (1998), pp. 118–128. (pp. 15, 16)

BEITZEL, S. M., JENSEN, E. C., CHOWDHURY, A., GROSSMAN, D., AND FRIEDER, O. 2004. Hourly analysis of a very large topically categorized web query log. In *Proc. ACM SIGIR* (2004), pp. 321–328. (p. 113)

BELLOTTI, V. AND THORNTON, J. 2006. Managing activities with TV-ACTA: TaskVista and Activity-Centered Task Assistant. In *Proc. SIGIR workshop on Personal Information Management* (2006), pp. 8–11. (p. 20)

BENITEZ, A. B., BEIGI, M., AND CHANG, S.-F. 1998. Using relevance feedback in content-based image metasearch. *IEEE Computer* 2, 4, 59–69. (p. 16)

BERNSTEIN, Y., SHOKOUHI, M., AND ZOBEL, J. 2006. Compact features for detection of near-duplicates in distributed retrieval. In *Proc. SPIRE*, Number 4209 in Lecture Notes in Computer Science (2006), pp. 110–121. (p. 14)

BHARAT, K. AND BRODER, A. 1998. A technique for measuring the relative size and overlap of public web search engines. In *Proc. WWW* (1998), pp. 379–388. (pp. 12, 29, 30, 40, 59)

BORLUND, P. 2003. The IIR evaluation model: A framework for evaluation of interactive information retrieval systems. *Information Research 8*, 3. (p. 138)

BORLUND, P. AND INGWERSEN, P. 1998. Measures of relative relevance and ranked half-life. In *Proc. ACM SIGIR* (1998), pp. 324–331. (p. 138)

BOWMAN, C. M., DANZIG, P. B., HARDY, D. R., MANBER, U., AND SCHWARTZ, M. F. 1994. Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS-732-94, University of Colorado at Boulder Department of Computer Science. (p. 78)

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C. M., MALER, E., AND YERGEAU, F. 2006. Extensible markup language (XML) 1.0. `http://www.w3.org/TR/2006/REC-xml-20060816/`. (p. 187)

BRIN, S. AND PAGE, L. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proc. WWW* (1998), pp. 107–117. (p. 36)

BRODER, A., FONTURA, M., JOSIFIVSKI, V., KUMAR, R., MOTWANI, R., NABAR, S., PANIGRAHY, R., TOMKINS, A., AND XU, Y. 2006. Estimating corpus size via queries. In *Proc. CIKM* (2006), pp. 594–603. (pp. 29, 56, 58, 62, 71)

BUCKLEY, C. AND VOORHEES, E. M. 2000. Evaluating evaluation measure stability. In *Proc. ACM SIGIR* (2000), pp. 33–40. (pp. 132, 135)

BUCKLEY, C. AND VOORHEES, E. M. 2004. Retrieval evaluation with incomplete information. In *Proc. ACM SIGIR* (2004), pp. 25–32. (pp. 135, 136)

BUCKLEY, C. AND VOORHEES, E. M. 2005. Retrieval system evaluation. In E. M. VOORHEES AND D. K. HARMAN Eds., *TREC: Experiment and Evaluation in Information Retrieval*, pp. 53–78. MIT Press. (p. 132)

CALLAN, J. AND CONNELL, M. 2001. Query-based sampling of text databases. *ACM Trans. Info. Systems 19*, 2, 97–130. (pp. 12, 32, 62, 79, 80)

CALLAN, J., CONNELL, M., AND DU, A. 1999. Automatic discovery of language models for text databases. In *Proc. ACM SIGMOD* (1999), pp. 479–490. (pp. 12, 16, 32, 40, 42, 47, 53, 54, 56, 61, 68, 78, 79, 85, 115, 124)

CALLAN, J. P., CROFT, W. B., AND BROGLIO, J. 1995. TREC and TIPSTER experiments with INQUERY. *Information Processing and Management 31*, 3, 327–343. (p. 97)

CALLAN, J. P., CROFT, W. B., AND HARDING, S. M. 1992. The INQUERY retrieval system. In *Proc. Third International Conference on Database and Expert Systems Applications* (1992), pp. 78–83. (p. 77)

CALLAN, J. P., LU, Z., AND CROFT, W. B. 1995. Searching distributed collections with inference networks. In *Proc. ACM SIGIR* (1995), pp. 21–28. (pp. 9, 13, 14, 16, 51, 77, 78, 91, 97, 98, 108, 115)

CARTERETTE, B., ALLAN, J., AND SITARAMAN, R. 2006. Minimal test collections for information retrieval. In *Proc. ACM SIGIR* (2006), pp. 268–275. (p. 136)

CASTILLO, C., MARIN, M., RODRIGUEZ, A., AND BAEZA-YATES, R. 2004. Scheduling algorithms for web crawling. In *Proc. WebMedia/LA-WEB* (2004), pp. 10–17. (p. 8)

CHANG, K. C.-C., GARCIA-MOLINA, H., AND PAEPCKE, A. 1996. Boolean query mapping across heterogenous information sources. *IEEE Transactions on Knowledge and Data Engineering 8*, 4, 515–521. (p. 13)

CHO, J. AND GARCIA-MOLINA, H. 2000. The evolution of the web and implications for an incremental crawler. In *Proc. VLDB* (2000), pp. 200–209. (p. 9)

CHO, J. AND GARCIA-MOLINA, H. 2003. Estimating frequency of change. *ACM Trans. Internet Technology 3*, 3, 256–290. (p. 9)

CHOKSHI, B., DYER, J. W., HERNANDEZ, T., AND KHAMBHAMPATI, S. 2006. Relevance and overlap aware text collection selection. Technical Report ASU CSE TR 06-019, Department of Computer Science and Engineering, Arizona State University. (p. 111)

CLAYPOOL, M., LE, P., WASEDA, M., AND BROWN, D. 2001. Implicit interest indicators. In *Proc. Intelligent User Interfaces* (2001), pp. 33–40. (p. 141)

CLEVERDON, C. W. 1967. The Cranfield tests on index language devices. In K. SPÄRK JONES AND P. WILLETT Eds., *Readings in Information Retrieval*, Series in Multimedia Information and Systems, pp. 47–59. San Francisco, CA, USA: Morgan Kaufmann. (p. 132)

CLEVERDON, C. W. AND MILLS, J. 1963. The testing of index language devices. In K. SPÄRK JONES AND P. WILLETT Eds., *Readings in Information Retrieval*, Series in Multimedia Information and Systems, pp. 98–110. San Francisco, CA, USA: Morgan Kaufmann. (pp. 132, 133)

COHEN, W. W. 1995. Fast effective rules induction. In *Proc. Twelfth International Conference on Machine Learning* (1995), pp. 115–123. (p. 75)

COHEN, W. W. 1996. Learning trees and rules with set-valued features. In *Proc. Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference*, Volume 1 (1996), pp. 709–716. (p. 75)

CONRAD, J. G., GUO, X. S., AND SCHRIBER, C. P. 2003. Online duplicate document detection: Signature reliability in a dynamic retrieval environment. In *Proc. CIKM* (2003), pp. 443–452. (p. 14)

COOPER, W. S. 1973. On selecting a measure of retrieval effectiveness. *JASIS 24*, 2, 87–100. (pp. 133, 140)

COPE, J., CRASWELL, N., AND HAWKING, D. 2003. Automated discovery of search interfaces on the web. In *Proc. 14th Australasian Database Conference* (2003), pp. 181–189. (pp. 11, 12)

CORMACK, G. V., PALMER, C. R., AND CLARKE, C. L. A. 1998. Efficient construction of large test collections. In *Proc. ACM SIGIR* (1998), pp. 282–289. (p. 136)

COX, C. 2006. An analysis of the impact of federated search products on library instruction using the ACRL standards. *portal: Libraries and the Academy 6*, 3, 253–267. (p. 26)

CRASWELL, N., BAILEY, P., AND HAWKING, D. 2000. Server selection on the world wide web. In *Proc. ACM International Conference on Digital Libraries* (2000), pp. 37–46. (pp. 12, 91, 96, 97, 98, 101, 108, 110, 112, 115, 117)

CRASWELL, N., CRIMMINS, F., HAWKING, D., AND MOFFAT, A. 2004. Performance and cost tradeoffs in web search. In *Proc. Australasian Database Conference* (2004), pp. 161–170. (pp. 5, 8, 9, 10)

CRASWELL, N., HAWKING, D., AND THISTLEWAITE, P. 1999. Merging results from isolated search engines. In *Proc. Australasian Database Conference* (1999), pp. 189–200. (p. 51)

CRISPIN, M. 2003. Internet message access protocol — version 4rev1. RFC 3501. (p. 188)

CUTRELL, E., ROBBINS, D., DUMAIS, S., AND SARIN, R. 2006. Fast, flexible filtering with Phlat — personal search and organisation made easy. In *Proc. CHI* (2006), pp. 261–270. (p. 20)

DAWSON, F. AND STENERSON, D. 1998. Internet calendaring and scheduling core object specification (icalendar). RFC 2445. (p. 188)

DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., LANDAUER, T. K., AND HARSHMAN, R. 1990. Indexing by latent semantic analysis. *JASIS 41*, 6, 391–407. (p. 74)

DOLIN, R., AGRAWAL, D., DILLON, L., AND EL ABBADI, A. 1996. Pharos: A scalable distributed architecture for locating heterogeneous information sources. Technical Report TRCS96-05, Department of Computer Science, University of California at Santa Barbara. (pp. 11, 13, 25, 74, 167)

DOLIN, R., AGRAWAL, D., AND EL ABBADI, A. 1999. Scalable collection summarization and selection. In *Proc. ACM International Conference on Digital Libraries* (1999), pp. 49–58. (p. 74)

DOLIN, R., AGRAWAL, D., EL ABBADI, A., AND DILLON, L. 1997. Pharos: A scalable distributed architecture for locating heterogeneous information sources. In *Proc. CIKM* (1997), pp. 348–355. (pp. 74, 111)

DREILINGER, D. AND HOWE, A. E. 1997. Experiences with selecting search engines using metasearch. *ACM Trans. Info. Systems 15*, 3, 195–222. (pp. 15, 110)

D'SOUZA, D., THOM, J. A., AND ZOBEL, J. 2004a. Collection selection for managed distributed document databases. *Information Processing and Management 40*, 3, 527–546. (pp. 40, 98, 121)

D'SOUZA, D., ZOBEL, J., AND THOM, J. A. 2004b. Is CORI effective for collection selection? An exploration of parameters, queries, and data. In *Proc. Australasian Document Computing Symposium* (2004). (p. 98)

DUDA, R. O. AND HART, P. E. 1975. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, USA. (p. 75)

DUMAIS, S., CUTRELL, E., CADIZ, J. J., JANCKE, G., SARIN, R., AND ROBBINS, D. C. 2003. Stuff I've Seen: A system for personal information retrieval and re-use. In *Proc. ACM SIGIR* (2003), pp. 72–79. (pp. 20, 21, 141)

EDWARDS, J., MCCURLEY, K., AND TOMLIN, J. 2001. An adaptive model for optimizing performance of an incremental web crawler. In *Proc. WWW* (2001), pp. 106–113. (p. 8)

FETTERLY, D., MANASSE, M., NAJORK, M., AND WIENER, J. 2003. A large-scale study of the evolution of web pages. In *Proc. WWW* (2003), pp. 669–678. (p. 9)

FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. 1999. Hypertext transfer protocol — HTTP/1.1. RFC 2616. (p. 5)

FOX, S. 2003. Evaluating implicit measures to improve the search experience. Talk presented at *SIGIR Workshop on Implicit Measures of User Interests and Preferences*. (p. 137)

FOX, S., KARNAWAT, K., MYDLAND, M., DUMAIS, S., AND WHITE, T. 2005. Evaluating implicit measures to improve web search. *ACM Trans. Info. Systems 23*, 2, 147–168. (p. 142)

FRENCH, J. C., POWELL, A. L., AND CALLAN, J. 1999. Effective and efficient automatic database selection. Technical Report CS-99-08, Department of Computer Science, University of Virginia. (p. 98)

FRENCH, J. C., POWELL, A. L., CALLAN, J., VILES, C. L., EMMITT, T., PREY, K. J., AND MOU, Y. 1999. Comparing the performance of database selection algorithms. In *Proc. ACM SIGIR* (1999), pp. 238–245. (pp. 10, 91, 96, 97, 98, 111, 116, 120, 121)

FRENCH, J. C., POWELL, A. L., VILES, C. L., EMMITT, T., AND PREY, K. J. 1998. Evaluating database selection techniques: A testbed and experiment. In *Proc. ACM SIGIR* (1998), pp. 121–129. (pp. 10, 13, 91, 97, 111, 116)

FREYNE, J., SMYTH, B., COYLE, M., BALFE, E., AND BRIGGS, P. 2004. Further experiments on collaborative ranking in community-based web search. *Artificial Intelligence Review 21*, 3–4, 229–252. (p. 9)

FUHR, N. 1999. A decision-theoretic approach to database selection in networked IR. *ACM Trans. Info. Systems 17*, 3, 229–249. (pp. 108, 109, 144)

GAUCH, S., WANG, G., AND GOMEZ, M. 1996. ProFusion: Intelligent fusion from multiple, distributed search engines. *Journal of Universal Computer Science 2*, 9, 637–649. (pp. 14, 75, 109, 111)

GLOVER, E. J., LAWRENCE, S., BIRMINGHAM, W. P., AND GILES, C. L. 1999. Architecture of a metasearch engine that supports user information needs. In *Proc. CIKM* (1999), pp. 210–216. (pp. 12, 15, 22, 25, 110)

GOLDBERG, J. L. 1995. CDM: An approach to learning in text categorization. In *Proc. Seventh International Conference on Tools with Artificial Intelligence* (1995), pp. 258–265. (p. 100)

GRAVANO, L., CHANG, K., GARCÍA-MOLINA, H., LAGOZE, C., AND PAEPCKE, A. 1997a. STARTS: Stanford protocol proposal for internet retrieval and search. In *Proc. ACM SIGMOD* (1997), pp. 207–218. (pp. 11, 77, 167)

GRAVANO, L., CHANG, K., GARCÍA-MOLINA, H., LAGOZE, C., AND PAEPCKE, A. 1997b. STARTS: Stanford protocol proposal for internet retrieval and search. `http://infolab.stanford.edu/~gravano/starts.html`. Protocol specification. (p. 11)

GRAVANO, L. AND GARCÍA-MOLINA, H. 1995. Generalizing GlOSS to vector-space databases and broker hierarchies. In *Proc. VLDB* (1995), pp. 78–89. (pp. 92, 94, 95, 96, 97, 104, 115, 116, 188)

GRAVANO, L., GARCÍA-MOLINA, H., AND TOMASIC, A. 1994. The effectiveness of GlOSS for the text database discovery problem. In *Proc. ACM SIGMOD* (1994), pp. 126–137. (pp. 13, 91, 92, 94, 116)

GRAVANO, L., GARCÍA-MOLINA, H., AND TOMASIC, A. 1999. GlOSS: Text-source discovery over the internet. *ACM Trans. Database Systems 24*, 2, 229–264. (pp. 13, 77, 78, 91, 92, 94, 97, 116, 118)

GRAVANO, L. AND IPEIROTIS, P. G. 2003. QProber: A system for automatic classification of hidden-web databases. *ACM Trans. Info. Systems 21*, 1, 1–41. (pp. 12, 13, 75)

GULLI, A. AND SIGNORINI, A. 2005. The indexable web is more than 11.5 billion pages. In *Proc. WWW* (2005), pp. 902–903. Poster. (pp. 31, 40, 60)

HALL, E. 2005. The application/mbox media type. RFC 4155. (p. 188)

HANCOCK-BEAULIEU, M. 1990. Evaluating the impact of an online library catalogue on subject searching behaviour at the catalogue and at the shelves. *Journal of Documentation 46*, 318–338. (p. 141)

HANSEN, P. AND JÄRVELIN, K. 2000. The information seeking and retrieval process at the Swedish Patent and Registration Office. In *Proc. ACM SIGIR Workshop on Patent Retrieval* (2000). (p. 141)

HARMAN, D. 1998. Towards interactive query expansion. In *Proc. ACM SIGIR* (1998), pp. 321–331. (p. 137)

HARMAN, D. K. 2005. The TREC test collections. In E. M. VOORHEES AND D. K. HARMAN Eds., *TREC: Experiment and Evaluation in Information Retrieval*, pp. 21–52. MIT Press. (pp. 40, 113, 135)

HAWKING, D., BAILEY, P., AND CRASWELL, N. 2000. Efficient and flexible search using text and metadata. Technical Report 2000/83, CSIRO Mathematical and Information Sciences. `http://es.csiro.au/pubs/hawking_tr00b.pdf`. (pp. 39, 61, 113, 118)

HAWKING, D. AND CRASWELL, N. 2005. Very large scale retrieval and web search. In E. M. VOORHEES AND D. K. HARMAN Eds., *TREC: Experiment and Evaluation in Information Retrieval*, pp. 199–232. MIT Press. (pp. 132, 136)

HAWKING, D., CRASWELL, N., BAILEY, P., AND GRIFFITHS, K. 2001. Measuring search engine quality. *Information Retrieval 4*, 1, 33–59. (p. 14)

HAWKING, D., PARIS, C., WILKINSON, R., AND WU, M. 2005. Context in enterprise search and delivery. In *Proc. IRiX Workshop, ACM SIGIR* (2005), pp. 14–16. (p. 142)

HAWKING, D. AND THISTLEWAITE, P. 1999. Methods for information server selection. *ACM Trans. Info. Systems 17*, 1, 40–76. (pp. 10, 91, 107, 116)

HAWKING, D. AND THOMAS, P. 2005. Server selection methods in hybrid portal search. In *Proc. ACM SIGIR* (2005), pp. 75–82. (pp. 10, 32, 41, 56, 79, 91, 98, 103, 106, 111, 112, 115, 124, 127)

HENZINGER, M. R., HEYDON, A., MITZENMACHER, M., AND NAJORK, M. 2000. On near-uniform URL sampling. In *Proc. WWW* (2000), pp. 295–308. (pp. 35, 40, 60)

HERNANDEZ, T. AND KAMBHAMPATI, S. 2005. Improving text collection selection with coverage and overlap statistics. In *Proc. WWW* (2005), pp. 1128–1129. Poster. (pp. 12, 111)

HERSH, W. AND OVER, P. 2001. TREC-9 interactive track report. In *Proc. TREC* (2001), pp. 41–50. (p. 138)

HERSH, W., PENTECOST, J., AND HICKAM, D. 1996. A task-oriented approach to information retrieval evaluation. *JASIS 47*, 1, 50–56. (p. 140)

HERSH, W., TURPIN, A., PRICE, S., CHAN, B., KRAEMER, D., SACHEREK, L., AND OLSON, D. 2000. Do batch and user evaluations give the same results? In *Proc. ACM SIGIR* (2000), pp. 17–24. (p. 139)

HOWE, A. E. AND DREILINGER, D. 1997. SavvySearch: A meta-search engine that learns which search engines to query. *AI Magazine 18*, 2, 19–25. (pp. 15, 110)

HYUSEIN, B. AND CARTHY, J. 2004. An advanced server ranking algorithm for distributed information retrieval systems on the internet. In *Proc. International Symposium on Computer and Information Sciences*, Volume 3280 of *Lecture Notes in Computer Science* (2004), pp. 837–844. Springer-Verlag. (p. 111)

IPEIROTIS, P. G. AND GRAVANO, L. 2002. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proc. VLDB* (2002), pp. 394–406. (pp. 78, 110)

IPEIROTIS, P. G. AND GRAVANO, L. 2004. When one sample is not enough: Improving text database selection using shrinkage. In *Proc. ACM SIGMOD* (2004), pp. 767–778. (pp. 77, 83)

IPEIROTIS, P. G., GRAVANO, L., AND SAHAMI, M. 2001. Probe, count, and classify: Categorising hidden-web databases. In *Proc. ACM SIGMOD* (2001), pp. 67–78. (pp. 12, 75, 76)

Jansen, B. J., Spink, A., and Saracevic, T. 2000. Real life, real users, and real needs: A study and analysis of user queries on the web. *Information Processing and Management 36*, 2, 207–227. (p. 113)

Järvelin, K. and Kekäläinen, J. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proc. ACM SIGIR* (2000), pp. 41–48. (p. 132)

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proc. Tenth European Conference on Machine Learning* (1998), pp. 137–142. (p. 75)

Joachims, T. 2002a. Evaluating retrieval performance using clickthrough data. In *Proc. SIGIR Workshop on Mathematical/Formal Methods in IR* (2002). (pp. 138, 144)

Joachims, T. 2002b. Optimizing search engines using clickthrough data. In *Proc. ACM Conference on Knowledge Discovery and Data Mining* (2002), pp. 133–142. (p. 9)

Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Proc. ACM SIGIR* (2005), pp. 154–161. (pp. 106, 135, 137)

Kantor, B. and Lapsley, P. 1986. Network news transfer protocol: A proposed standard for the stream-based transmission of news. RFC 977. (p. 74)

Kelly, D. and Belkin, N. J. 2004. Display time as implicit feedback: Understanding task effects. In *Proc. ACM SIGIR* (2004), pp. 377–384. (p. 141)

Koutrika, G. and Ioannidis, Y. 2005. A unified user profile framework for query disambiguation and personalisation. In *Proc. Workshop on New Technology for Personalized Information Access* (2005), pp. 44–45. (p. 22)

Kristol, D. and Montulli, L. 2000. HTTP state management mechanism. RFC 2965. (p. 178)

Krumpholz, A. H. and Hawking, D. 2006. InexBib — retrieving XML documents based on external evidence. In *Proc. Australasian Document Computing Symposium* (2006), pp. 72–79. (p. 154)

Kullback, S. 1959. *Information Theory and Statistics*. John Wiley & Sons, New York, NY, USA. (p. 83)

Kullback, S. and Leibler, R. A. 1951. On information and sufficiency. *Annals of Mathematical Statistics 22*, 1, 79–86. (pp. 82, 83)

Lancaster, F. W. 1969. *MEDLARS: Report on the Evaluationo of its Operating Efficiency*. Series in Multimedia Information and Systems. Morgan Kaufmann, San Francisco, CA, USA. (p. 133)

Large, A. and Beheshti, J. 1997. OPACs: A research review. *Library & Information Science Research 19*, 2, 111–133. (p. 141)

Lawrence, S. and Giles, C. L. 1998. Searching the World Wide Web. *Science 280*, 98–100. (p. 60)

LAWRENCE, S. AND GILES, C. L. 1999. Accessibility of information on the web. *Nature 400*, 107–109. (p. 60)

LEMPEL, R. AND MORAN, S. 2002. Optimizing result prefetching in web search engines with segmented indicies. In *Proc. VLDB* (2002), pp. 370–381. (p. 9)

LEVY, A. Y., RAJARAMAN, A., AND ORDILLE, J. J. 1996. Querying heterogenous information sources using source descriptions. In *Proc. VLDB* (1996), pp. 251–262. (p. 13)

LIMBU, D. K., CONNOR, A., AND MACDONELL, S. 2006. Contextual relevance feedback in web information retrieval. In *Proc. IIiX Symposium on Information Interaction in Context* (2006), pp. 235–244. (p. 22)

LIU, K.-L., MENG, W., QUI, J., YU, C., RAGHAVAN, V., WU, Z., LU, Y., HE, H., AND ZHAO, H. 2007. AllInOneNews: Development and evaluation of a large-scale news metasearch engine. In *Proc. ACM SIGMOD* (2007), pp. 1017–1028. (p. 16)

LIU, K.-L., SANTOSO, A., YU, C., MENG, W., AND ZHANG, C. 2001. Discovering the representative of a search engine. In *Proc. CIKM* (2001), pp. 577–579. Poster. (pp. 12, 29, 52, 53, 60, 61, 62, 76, 124)

LOPATOVSKA, I. 2006. Measuring experienced utility of information retrieval systems: Experimental approach. In *Proc. ASIST*, Volume 42 (2006). (p. 141)

LU, Z., CALLAN, J. P., AND CROFT, W. B. 1996. Measures in collection ranking evaluation. Technical Report 96-39, Computer Science Department, University of Massachusetts. (pp. 115, 116)

MAGENNIS, M. AND VAN RIJSBERGEN, C. J. 1997. The potential and actual effectiveness of interactive query expansion. In *Proc. ACM SIGIR* (1997), pp. 324–332. (p. 137)

MANMATHA, R., RATH, T., AND FENG, F. 2001. Modeling score distributions for combining the outputs of search engines. In *Proc. ACM SIGIR* (2001), pp. 267–275. (p. 106)

MCGOWAN, J. P., KUSHMERICK, N., AND SMYTH, B. 2002. Who do you want to be today? Web personae for personalised information access. In *Proc. Adaptive Hypermedia and Adaptive Web-Based Systems*, Number 2347 in Lecture Notes in Computer Science (2002), pp. 514–517. (p. 22)

MENG, W., WANG, W., SUN, H., AND YU, C. 2002. Concept hierarchy based text database categorization. *Knowledge and Information Systems 4*, 2, 132–150. (pp. 75, 76, 111)

MENG, W., WU, Z., YU, C., AND LI, Z. 2001. A highly scalable and effective method for metasearch. *ACM Trans. Info. Systems 19*, 3, 310–335. (p. 103)

MOFFAT, A. AND ZOBEL, J. 1994. Information retrieval systems for large document collections. In *Proc. TREC* (1994), pp. 85–94. (p. 99)

NARDI, B. AND BARREAU, D. 1997. “Finding and reminding” revisited: Appropriate metaphors for file organization on the desktop. *ACM SIGIR Bulletin 29*, 1. (p. 135)

NORDLIE, R. 1999. “User revealment” — a comparison of initial queries and ensuing question development in online searching and in human reference interactions. In *Proc. ACM SIGIR* (1999), pp. 11–18. (p. 141)

NOTTELMANN, H. AND FUHR, N. 2003. Evaluating different methods of estimating retrieval quality for resource selection. In *Proc. ACM SIGIR* (2003), pp. 290–297. (pp. 56, 109, 124)

NOTTELMANN, H. AND FUHR, N. 2004. Combining CORI and the decision-theoretic approach for advanced resource selection. In *Proc. ECIR* (2004), pp. 138–153. (pp. 98, 109)

PERKOWITZ, M., DOORENBOS, R. B., ETZIONI, O., AND WELD, D. S. 1997. Learning to understand information on the internet: An example-based approach. *Journal of Intelligent Information Systems 8*, 133–153. (p. 10)

PETERS, C., BRASCHLER, M., GONZALO, J., AND KLUCK, M. Eds. 2001. *Second Workshop of the Cross-Language Evaluation Forum, CLEF 2001*, Volume 2406 of *Lecture Notes in Computer Science* (2001). (p. 132)

PITKOW, J., SCHÜTZE, H., CASS, T., COOLEY, R., TURNBULL, D., EDMONDS, A., ADAR, E., AND BREUEL, T. 2002. Personalized search. *Comm. ACM 45*, 9, 50–55. (p. 22)

PONTE, J. M. AND CROFT, W. B. 1998. A language modeling approach to information retrieval. In *Proc. ACM SIGIR* (1998), pp. 275–281. (pp. 76, 77, 101)

POWELL, A. L. AND FRENCH, J. C. 2003. Comparing the performance of collection selection algorithms. *ACM Trans. Info. Systems 21*, 4, 412–456. (pp. 10, 91, 92, 96, 97, 98, 101, 111, 116)

POWELL, A. L., FRENCH, J. C., CALLAN, J., CONNELL, M., AND VILES, C. L. 2000. The impact of database selection on distributed searching. In *Proc. ACM SIGIR* (2000), pp. 232–239. (pp. 10, 41, 91, 98, 115, 117)

POWELL, J. AND FOX, E. A. 1998. Multilingual federated searching across heterogenous collections. *D-Lib Magazine 4*, 9. (p. 78)

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in FORTRAN* (2nd ed.). Cambridge University Press, Cambridge, United Kingdom. (p. 80)

RAGGETT, D., HORS, A. L., AND JACOBS, I. 1999. HTML 4.01 specification. `http://www.w3.org/TR/1999/REC-html401-19991224/`. (pp. 5, 176, 187)

RAGHAVAN, S. AND GARCIA-MOLINA, H. 2001. Crawling the hidden web. In *Proc. VLDB* (2001), pp. 129–138. (p. 10)

RASOLOFO, Y., ABBACI, F., AND SAVOY, J. 2001. Approaches to collection selection and results merging for distributed information retrieval. In *Proc. CIKM* (2001), pp. 191–198. (pp. 10, 108, 112, 115, 117)

RASOLOFO, Y., HAWKING, D., AND SAVOY, J. 2003. Result merging strategies for a current news metasearcher. *Information Processing and Management 39*, 4, 581–609. (pp. 14, 158, 182, 183, 184, 189)

RICKER, W. E. 1975. *Computation and Interpretation of Biological Statistics of Fish Populations*. Number 191 in Bulletins of the Fisheries Research Board of Canada. Department of Fisheries and the Environment, Ottawa, Canada. (pp. 52, 54)

ROBERTSON, S. E., WALKER, S., JONES, S., HANCOCK-BEAULIEU, M. M., AND GATFORD, M. 1994. Okapi at TREC-3. In *Proc. TREC* (1994), pp. 109–126. (p. 118)

RU, Y. AND HOROWITZ, E. 2005. Indexing the invisible web: A survey. *Online Information Review 29*, 3, 249–265. (p. 76)

RUSMEVICHIENTONG, P., PENNOCK, D. M., LAWRENCE, S., AND GILES, C. L. 2001. Methods for sampling pages uniformly from the world wide web. In *Proc. AAAI Fall Symposium on Using Uncertainty Within Computation* (2001), pp. 121–128. (pp. 35, 37, 40)

RUTHVEN, I. 2003. Re-examining the potential effectiveness of interactive query expansion. In *Proc. ACM SIGIR* (2003), pp. 213–220. (p. 137)

SAKAI, T. 2004. New performance measures based on multigrade relevance: Their application to question answering. In *Proc. NTCIR* (2004). (p. 132)

SALTON, G. AND MCGILL, M. J. 1983. The SMART and SIRE experimental retrieval systems. In K. SPÄRK JONES AND P. WILLETT Eds., *Readings in Information Retrieval*, Series in Multimedia Information and Systems. San Francisco, CA, USA: Morgan Kaufmann. (pp. 32, 42, 101)

SANDERSON, M. AND JOHO, H. 2004. Forming test collections with no system pooling. In *Proc. ACM SIGIR* (2004), pp. 33–40. (p. 136)

SANDERSON, M. AND ZOBEL, J. 2005. Information retrieval system evaluation: Effort, sensitivity, and reliability. In *Proc. ACM SIGIR* (2005), pp. 162–169. (p. 135)

SCHUMACHER, F. X. AND ESCHMEYER, R. W. 1943. The estimate of fish population in lakes or ponds. *Journal of the Tennessee Academy of Science 18*, 228–249. (p. 54)

SELBERG, E. AND ETZIONI, O. 1995. Multi-service search and comparison using the MetaCrawler. In *Proc. WWW* (1995). (pp. 12, 14)

SHELDON, M. A., DUDA, A., WEISS, R., O'TOOLE, J., AND GIFFORD, D. K. 1994. Content routing for distributed information servers. In *Proc. Int. Conf. on Extending Database Technology* (1994), pp. 109–122. (pp. 74, 111)

SHEN, X., TAN, B., AND ZHAI, C. 2005. Implicit user modeling for personalized search. In *Proc. CIKM* (2005), pp. 824–831. (pp. 21, 22, 140, 145)

SHESKIN, D. J. 2004. *Handbook of Parametric and Nonparametric Statistical Procedures* (3rd ed.). Chapman & Hall/CRC, Boca Raton, Florida, USA. (pp. 42, 55, 80, 81, 89, 125)

SHOKOUHI, M. 2007. Central-rank-based collection selection in uncooperative distributed information retrieval. In *Proc. ECIR* (2007), pp. 160–172. (pp. 91, 106, 107, 116, 118, 119)

SHOKOUHI, M., BAILLIE, M., AND AZZOPARDI, L. 2007. Updating collection representations for federated search. In *Proc. ACM SIGIR* (2007), pp. 511–518. (pp. 76, 78, 79, 115)

SHOKOUHI, M., SCHOLER, F., AND ZOBEL, J. 2006. Sample sizes for query probing in uncooperative distributed information retrieval. In *APWeb 2006*, Volume 3841 of *Lecture Notes in Computer Science* (2006), pp. 63–75. Springer. (pp. 32, 40, 62)

SHOKOUHI, M., ZOBEL, J., SCHOLER, F., AND TAHAGHOGHI, S. M. M. 2006. Capturing collection size for distributed non-cooperative retrieval. In *Proc. ACM SIGIR* (2006), pp. 316–323. (pp. 12, 29, 32, 53, 54, 56, 60, 62, 65, 66, 69, 70, 115, 124, 187)

SI, L. AND CALLAN, J. 2003a. The effect of database size distribution on resource selection algorithms. In *Proc. ACM SIGIR* (2003), pp. 31–42. (pp. 51, 98, 102, 103, 106, 115, 116, 117, 118, 124, 127)

SI, L. AND CALLAN, J. 2003b. Relevant document distribution estimation method for resource selection. In *Proc. ACM SIGIR* (2003), pp. 298–305. (pp. 12, 13, 29, 32, 41, 51, 55, 56, 62, 79, 91, 98, 104, 105, 106, 115, 116, 118, 124)

SI, L. AND CALLAN, J. 2003c. A semisupervised learning method to merge search engine results. *ACM Trans. Info. Systems 21*, 4, 457–491. (p. 16)

SI, L. AND CALLAN, J. 2004. Unified utility maximization framework for result selection. In *Proc. CIKM* (2004), pp. 32–41. (p. 110)

SI, L. AND CALLAN, J. 2005. Modeling search engine effectiveness for federated search. In *Proc. ACM SIGIR* (2005), pp. 83–90. (pp. 110, 167)

SI, L., JIN, R., CALLAN, J., AND OGILVIE, P. 2002. A language modeling framework for resource selection and results merging. In *Proc. CIKM* (2002), pp. 391–397. (pp. 13, 14, 77, 78, 101, 102, 115, 118, 124, 188)

SILVERSTEIN, C., HENZINGER, M., MARAIS, H., AND MORICZ, M. 1999. Analysis of a very large web search engine query log. *SIGIR Forum 33*, 1, 6–12. (pp. 26, 113)

SOBOROFF, I. 2006. Dynamic test collections: Measuring search effectiveness on the live web. In *Proc. ACM SIGIR* (2006), pp. 276–283. (p. 132)

SOBOROFF, I., NICHOLAS, C., AND CAHAN, P. 2001. Ranking retrieval systems without relevance judgements. In *Proc. ACM SIGIR* (2001), pp. 66–73. (p. 135)

SOBOROFF, I. AND ROBERTSON, S. 2003. Building a filtering test collection for TREC 2002. In *Proc. ACM SIGIR* (2003), pp. 243–250. (p. 136)

SPÄRCK JONES, K. AND VAN RIJSBERGEN, C. J. 1976. Information retrieval test collections. *Journal of Documentation 32*, 1, 59–75. (p. 132)

Spink, A., Ozmutlu, S., Ozmutlu, H. C., and Jansen, B. J. 2002. U.S. versus European web searching trends. *SIGIR Forum 36*, 2, 32–38. (p. 113)

Sugiyama, K., Hatano, K., and Yoshikawa, M. 2004. Adaptive web search based on user profile constructed without any effort from users. In *Proc. WWW* (2004), pp. 675–684. (p. 22)

Teevan, J., Alvarado, C., Ackerman, M. S., and Karger, D. R. 2004. The perfect search engine is not enough: A study of orienteering behaviour in directed search. In *Proc. Conf. Human Factors in Computing Systems* (2004), pp. 415–422. (pp. 5, 134, 135)

Teevan, J., Dumais, S. T., and Horvitz, E. 2005a. Beyond the commons: Investigating the value of personalizing web search. In *Proc. Workshop on New Technology for Personalized Information Access* (2005), pp. 84–92. (p. 22)

Teevan, J., Dumais, S. T., and Horvitz, E. 2005b. Personalizing search via automated analysis of interests and activities. In *Proc. ACM SIGIR* (2005), pp. 449–456. (p. 21)

Thomas, P. and Hawking, D. 2006. Evaluation by comparing result sets in context. In *Proc. CIKM* (2006), pp. 94–101. (p. 131)

Thomas, P. and Hawking, D. 2007. Evaluating sampling methods for uncooperative collections. In *Proc. ACM SIGIR* (2007), pp. 503–510. (p. 29)

Thomas, P. and Rowlands, T. 2007. Estimating the value of automatic disambiguation. In *Proc. ACM SIGIR* (2007), pp. 719–720. Poster. (p. 22)

Turpin, A. and Hersh, W. 2001. Why batch and user evaluations do not give the same results. In *Proc. ACM SIGIR* (2001), pp. 225–231. (p. 139)

Turpin, A. and Scholer, F. 2006. User performance versus precision measures for simple search tasks. In *Proc. ACM SIGIR* (2006), pp. 11–18. (pp. 139, 160)

Turtle, H. R. and Croft, W. B. 1991. Evaluation of an inference network-based retrieval model. *ACM Trans. Info. Systems 9*, 3, 187–222. (p. 97)

van Rijsbergen, C. J. 1979. *Information Retrieval* (2nd ed.). Butterworths, London, United Kingdom. (p. 132)

Vaughan, L. 2004. New measurements for search engine evaluation proposed and tested. *Information Processing and Management 40*, 4, 677–691. (p. 147)

Voorhees, E. 1999. The TREC-8 question answering track. In *Proc. TREC* (1999), pp. 77–92. (p. 132)

Voorhees, E. and Tong, R. 1997. Multiple search engines in database merging. In *Proc. ACM International Conference on Digital Libraries* (1997), pp. 93–102. (p. 13)

Voorhees, E. M. 1995. Siemens TREC-4 report: Further experiments with database merging. In *Proc. TREC* (1995), pp. 121–130. (p. 99)

Voorhees, E. M. and Buckley, C. 2002. The effect of topic set size on retrieval experiment error. In *Proc. ACM SIGIR* (2002), pp. 316–323. (p. 135)

VOORHEES, E. M. AND GAROFOLO, J. S. 2005. Retrieving noisy text. In E. M. VOORHEES AND D. K. HARMAN Eds., *TREC: Experiment and Evaluation in Information Retrieval*, pp. 183–197. MIT Press. (p. 136)

VOORHEES, E. M., GUPTA, N. K., AND JOHNSON-LAIRD, B. 1994. The collection fusion problem. In *Proc. TREC* (1994), pp. 95–104. (pp. 13, 109, 144, 167)

VOORHEES, E. M. AND HARMAN, D. K. Eds. 2005. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press. (p. 132)

WEBBER, W. AND MOFFAT, A. 2005. In search of reliable retrieval experiments. In *Proc. Australasian Document Computing Symposium* (2005). (p. 133)

WHITE, R. W., RUTHVEN, I., AND JOSE, J. M. 2005. A study of factors affecting the utility of implicit relevance feedback. In *Proc. ACM SIGIR* (2005), pp. 35–42. (p. 140)

WHITE, R. W., RUTHVEN, I., JOSE, J. M., AND VAN RIJSBERGEN, C. J. 2005. Evaluating implicit feedback models using searcher simulations. *ACM Trans. Info. Systems 23*, 3, 325–361. (p. 137)

WILLIAMS, H. E. AND ZOBEL, J. 2005. Searchable words on the web. *Int'l Journal of Digital Libraries 5*, 2, 99–105. (p. 98)

WOLFRAM, D. 1992. Applying informetric characteristics of databases to IR system file design, part I: Informetric models. *Information Processing and Management 28*, 1, 121–133. (p. 82)

WU, M., FULLER, M., AND WILKINSON, R. 2001. Searcher performance in question answering. In *Proc. ACM SIGIR* (2001), pp. 375–381. (p. 140)

WU, M., MURESAN, G., MCLEAN, A., TANG, M.-C. M., WILKINSON, R., LI, Y., LEE, H.-J., AND BELKIN, N. J. 2004. Human versus machine in the topic distillation task. In *Proc. ACM SIGIR* (2004), pp. 385–392. (p. 139)

XU, J. AND CROFT, W. B. 1999. Cluster-based language models for distributed retrieval. In *Proc. ACM SIGIR* (1999), pp. 254–261. (pp. 10, 77, 78, 82, 83, 91, 101, 102, 107, 113, 115, 117, 188)

YANG, H. AND ZHANG, M. 2006. Two-stage statistical language models for text database selection. *Information Retrieval 9*, 1, 5–31. (p. 110)

YANG, Y. AND CHUTE, C. G. 1994. An example-based mapping method for text categorization and retrieval. *ACM Trans. Info. Systems 12*, 3, 252–277. (p. 76)

YILMAZ, E. AND ASLAM, J. A. 2006. Estimating average precision with incomplete and imperfect judgements. In *Proc. CIKM* (2006), pp. 102–111. (p. 137)

YU, C., MENG, W., LIU, K.-L., WU, W., AND RISHE, N. 1999. Efficient and effective metasearch for a large number of text databases. In *Proc. CIKM* (1999), pp. 217–224. (pp. 103, 104)

YUWONO, B. AND LEE, D. L. 1997. Server ranking for distributed text retrieval systems on the internet. In *Proc. 5th Int. Conf. on Database Systems for Advanced Applications* (1997), pp. 41–49. (pp. 13, 77, 78, 97, 100, 101, 116)

ZEILENGA, K. 2006. Lightweight directory access protocol (LDAP): Technical specification road map. RFC 4510. (p. 188)

ZHANG, Y. AND MOFFAT, A. 2006. Some observations on user search behaviour. In *Proc. Australasian Document Computing Symposium* (2006). (p. 113)

ZIPF, G. K. 1949. *Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley, Reading, MA, USA. (p. 82)

ZOBEL, J. 1997. Collection selection via lexicon inspection. In *Proc. Australasian Document Computing Symposium* (1997). (pp. 99, 115)

ZOBEL, J. 1998. How reliable are the results of large-scale information retrieval experiments? In *Proc. ACM SIGIR* (1998), pp. 307–314. (pp. 135, 136)