Federated Search of Text Search Engines in Uncooperative Environments

Luo Si

Language Technology Institute
School of Computer Science
Carnegie Mellon University
lsi@cs.cmu.edu

Thesis Committee:
Jamie Callan (Carnegie Mellon University, Chair)
Jaime Carbonell (Carnegie Mellon University)
Yiming Yang (Carnegie Mellon University)
Luis Gravano (Columbia University)

2006

ACKNOWLEDGEMENTS

I would never have been able to finish my dissertation without the guidance of my committee members, help from friends, and support from my family.

My first debt of gratitude must go to my advisor, Jamie Callan, for taking me on as a student and for leading me through many aspects of my academic career. Jamie is not only a wonderful advisor but also a great mentor. He told me different ways to approach research problems. He showed me how to collaborate with other researchers and share success. He guided me how to write conference and journal papers, had confidence in me when I doubted myself, and brought out the best in me. Jamie has given me tremendous help on research and life in general. This experience has been extremely valuable to me and I will treasure it for the rest of my life.

I would like to express my sincere gratitude towards the other members of my committee, Yiming Yang, Jamie Carbonell and Luis Gravano, for their efforts and constructive suggestions. Yiming has been very supportive for both my work and my life since I came to Carnegie Mellon University. Her insightful comments and suggestions helped a lot to improve the quality of this dissertation. Jamie has kindly pointed out different ways to solve research problems in my dissertation, which results in many helpful discussions. I am grateful of Luis for his vision and expertise, which helps me to think more deeply and get a better understanding of the research problems.

A large part of my achievements as a graduate student and researcher comes from the collaborations with my co-authors. I especially would like to thank Rong Jin, ChengXiang Zhai, Tapas Kanungo and Jimmy Huang. Through all these interactions, I have been exposed to so many great research problems and have learned many useful techniques.

I benefited so much from my study at Language Technology Institute in School of Computer Science, Carnegie Mellon University. It has been such a great experience to interact with faculty members, friends and fellow students. I would like to acknowledge Adele Weitz, Brooke Hyatt, Stacey Young, Roni Rosenfeld, Eric Xing, Alan Black, Eric Nyberg, Jie Yang, Yi Zhang, Paul Ogilvie, Kevyn Collins-Thompson, Jie Lu, Thi Nhu Truong, Grace Hui Yang, Pucktada Treeratpituk, Mark Hoy, Jian Zhang, Fan Li, Shinjae Yoo, Yue Pan, Qin Jin, Chun Jin, Guy Lebanon, Kathrin Probst, Fei Huang, Rong Zhang, Wen Wu, Yan Liu, Rong Yan, Gu Lingyun, Jiazhi Ou, Joy Zhang, Bing Zhao, Jun Yang, Han Bejamin, Brian Langner, Wei-hao Lin, Christian

Monson, Yi Chang, Ningning Hu, Datong Chen, Haiyan Wang, Jiang Gao, Kai Huang, Minglong Shao, Hong Yan. Especially, I would like to thank Paul Ogilvie and Jie Lu for proof reading this manuscript.

My deep gratitude is due to my parents for their continuous guidance, encouragement, and support during my life. Without their love, any achievement in my life would have been simply unimaginable. Finally, I want to express my special thanks to my dear wife Danni. Thank you for sharing the happiness and hardness through the years. Thank you to be in my life.

TABLE OF CONTENTS

Chapter 1: Introduction

Chapter 2: Modeling Federated Search Problems

Chapter 3: Adding Size to Resource Representations

Chapter 4: Resource Selection

Chapter 5: Results Merging

Chapter 6: Results Merging for Multilingual Ranked Lists

Chapter 7: Unified Utility Maximization Framework

Chapter 8: Modeling Search Engine Retrieval Effectiveness

Chapter 9: Conclusion and Future Work

Reference:

Abstract

Conventional search engines like Google provide access to Web information that can be acquired easily by crawling hyperlinks. However, a large amount of information cannot be copied arbitrarily by conventional search engines. This type of hidden information, which is very valuable, can only be accessed via an alternative search model other than the centralized retrieval model used by conventional search engines.

Federated search provides access to the hidden information by providing a single interface that connects to multiple source-specific search engines. There are three main research problems in federated search. First, information about the contents of each individual information source must be acquired (*resource representation*). Second, given a query, a set of sources must be selected to do the search (*resource selection*). Third, the results retrieved from selected sources may be merged into a single list before it is presented to the end user (*results merging*).

This dissertation addresses these main research problems within federated search. New algorithms are proposed for effectively and efficiently estimating information source sizes, estimating distributions of relevant documents across information sources for a given query, and merging document rankings returned by selected sources. Furthermore, a unified utility maximization framework is proposed to combine the range of individual solutions together to construct effective systems for different federated search applications. The framework can incorporate information such as search engine retrieval effectiveness, which is an important issue for real world federated search applications. Empirical studies in a wide range of research environments and a real world prototype system under different operating conditions demonstrate the effectiveness of the new algorithms.

This new research, supported by a more theoretical foundation, better empirical results, and more realistic simulation of real world applications, substantially improves the state-of-the-art of federated search. It serves as a bridge for moving federated search from an interesting research topic to a practical tool for real world applications.

Chapter 1: Introduction

This chapter motivates the research problems of federated search. It first describes hidden Web contents, in which a large amount of valuable information resides but cannot be searched by conventional search engines. It then proposes federated search as the search solution for this type of hidden information. Furthermore, several federated search applications are presented to address different types of information needs. This chapter also briefly introduces the goal and the contribution of this dissertation.

1.1 Hidden Web contents

A large amount of information has been accumulated on the Web. It was estimated that the Web has grown to contain about 74 million sites [Netcraft, 2005][1]. The explosive growth of the Web demands effective search solutions to find relevant information for Web users. Conventional search engines like Google or MSN have provided effective search solutions for some type of information on the Web by coping Web pages into a single centralized database, indexing the contents, and making them searchable. This type of information that can be copied by conventional search engines is called "*visible Web*" contents.

However, a large chunk of the Web is not accessible by conventional search engines. Many information sources contain information that cannot be copied into a centralized database. This type of information is called "*hidden Web*" contents (also called "invisible" or "deep" Web contents).

Hidden Web contents cover a wide range of topics. Some specific examples are: the database of the US Patent and Trademark Office[2] (USPTO), which contains full text of millions of approved or pending patents; the National Science Foundation's award database[3], which provides descriptions for funded scientific research projects; and the U.S. Government Printing Office[4] (GPO) portal, which connects to a large number of government agency databases. The contents of hidden Web are in English as well as many other languages

[1] http://news.netcraft.com/archives/2005/12/index.html

[2] http://www.uspto.gov/patft/index.html

[3] http://www.nsf.gov/awardsearch/

[4] http://www.gpoaccess.gov/databases.html

(e.g., the German Patent and Trademark Office[5] provides the search functionality of patents in several European languages). Furthermore, the contents of hidden Web are also in very diverse formats. Many hidden information sources contain documents in unstructured natural language (i.e., text) while many other hidden information sources provide information in semi-structured or structured formats. The National Institutes of Health's GeneBank[6] contains DNA sequence information for genes in thousands of species and Array Express in European Bioinformatics Institutes[7] is a public repository that provides a large amount of biological experimental data such as the microarray data.

The exact size of hidden Web contents is still unknown. However, previous studies have consistently reported that the size of hidden Web is larger (if not much larger) than that of visible Web (2-50 times larger in [Sherman, 2001]; 500 times larger [Bergman, 2001]). Moreover, the growth rate of the hidden Web contents is similar or faster than that of the visible Web contents [Bergman, 2001].

The information sources that contain hidden Web contents are called "*hidden information sources*". There exist different types of hidden information sources: i) many hidden information sources only allow the access of their contents via the source-specific search interfaces due to intellectual property protection; ii) some information sources allow their contents to be copied by conventional search engines, but the information is updated very frequently and it is difficult for conventional search engines to crawl the updated information immediately; and iii) the access of the contents within some hidden information sources is subject to fee or subscription. A previous study [Bergman, 2001] has shown that the third type of information sources that require fee or subscription accounts for only about three percent of the whole hidden Web.

Hidden Web information sources have to maintain source-specific search engines, which is associated with a large amount of human efforts. This often implies that the sources are not tiny, hence they are more likely to be created and maintained by professionals. Therefore, one might conclude that the contents within these information sources are often very valuable.

In addition to the Web, hidden information also exists among distributed information sources within large international organizations or medium-sized organizations, where different parties may control the sources or different sources serve different needs. It is generally difficult to afford the large amount of communication and maintenance costs to use a big centralized database within these types of environments.

[5] http://depatisnet.dpma.de/DepatisNet

[6] http://www.ncbi.nih.gov/Genbank/

[7] http://www.ebi.ac.uk/arrayexpress/

There are multiple types of hidden information, such as unstructured text data with text search engines and semi-structured or structured data with structured search engines. The focus of this dissertation is the hidden information that can be accessed through text search engines.

1.2 Federated search

For visible Web contents, the solutions of conventional search engines are very effective. They use the *centralized retrieval model* (i.e., ad-hoc retrieval) to copy the crawlable information into a single centralized database, index the contents and rank the documents in the database for user queries. This method works well when information sources expose their contents for Web crawlers. However, this is not true for hidden Web contents, where the information can only be accessed via source-specific search engines.

One key distinction between the centralized retrieval model and the federated search is that the searching process in federated search is conducted by the search engines of individual information sources, which reflects the distributed location and control of information among hidden information sources. The mechanism of federated search is more complex than the centralized retrieval model. It is commonly viewed as consisting of five subproblems, described as follows:

Resource discovery: A resource discovery algorithm [Cope, Craswell and Hawking, 2003] automatically identifies information sources that contain information hidden from conventional search engines.

Wrapper Induction: Wrapper induction [Gazen and Minton, 2005; He et al., 2006] provides a standard API of querying the search interfaces and extracting data from the returned result pages from each available hidden source.

The above two subproblems are important for federated search applications. However, they are not the focus of this dissertation. The research in this dissertation extensively studies the three main subproblems below.

Resource representation: It is very important to learn the subject areas as well as other key statistics of hidden information sources. There are different types of resource representations: content descriptions of hidden information sources by the words and their occurrences [Gravano et al., 1994; Callan, Lu & Croft, 1995; Voorhees et al., 1995; Viles & French, 1995; Ipeirotis & Gravano, 2004], information source size estimates (i.e., the number of documents) [Liu et al., 2001], search engine retrieval effectiveness profiles [Si

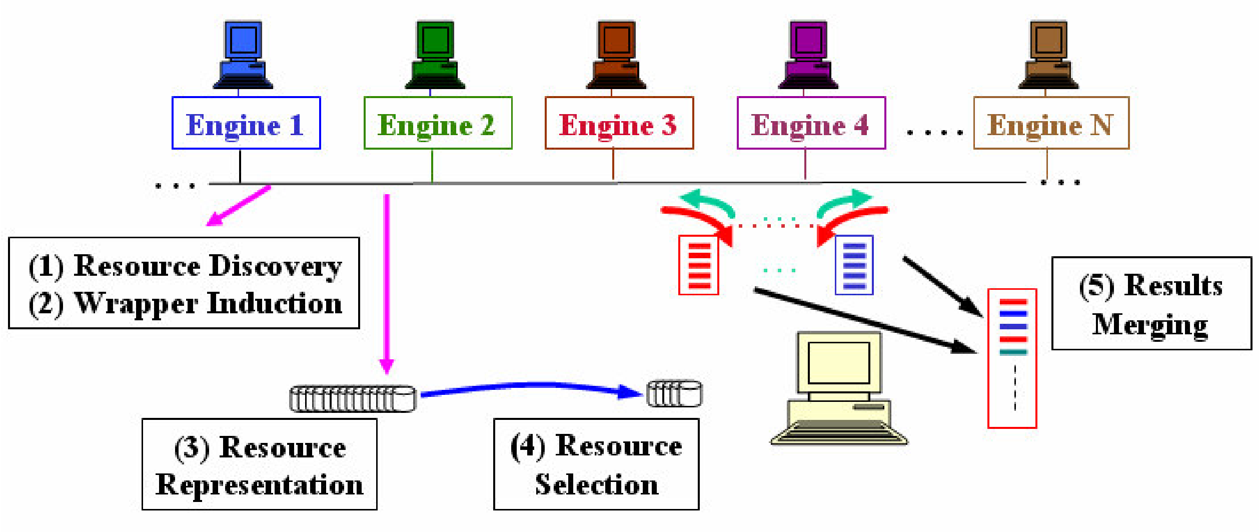

Figure 1.1: The main components of a federated search system.

& Callan, 2005a], search engine response time [Hosanagar, 2005], and so on. Acquiring accurate resource representations effectively and efficiently is very important for different federated search applications.

Resource selection: Given an information need as a text query, it is generally impractical to search all available information sources due to the high communication and computational costs. Resource selection algorithms choose a small set of information sources that are most appropriate for a user query [Gravano et al., 1994; Voorhees et al., 1995; Callan, Lu & Croft, 1995; Viles & French, 1995; Fuhr, 1996; Fuhr, 1999; Craswell, Hawking & Thistlewaite, 1999; French et al., 1999; Callan, 2000; Ipeirotis & Gravano, 2002; Si & Callan, 2003a; Nottelmann & Fuhr, 2003b; Si & Callan, 2004b].

Results merging: The user query can be forwarded to search the selected information sources. It is generally undesirable to present many individual ranked lists (e.g., more than 5) from selected information sources separately. A natural solution is to merge the ranked lists into a single list before presenting it to the end user [Voorhees et al., 1995; Kirsch, 1997; Craswell, Hawking & Thistlewaite, 1999; Si & Callan, 2002a; Si & Callan, 2003b].

A diagram of a federated search system that contains the components described above is shown in Figure 1.1.

There are various solutions for the three main subproblems of federated search as resource representation, resource selection and results merging, depending upon the degree of cooperation that can be assumed among the hidden information sources. In the environment of local networks within small companies, the information sources may choose the same type of retrieval algorithm and cooperate closely to provide their corpus statistics. This type of *cooperative environments* enables simplified and more effective solutions for federated search subproblems such as resource description and results merging [Callan, 2000; Gravano et al., 1997; Si et al., 2002b]. On the other hand, in wide area networks within large organizations or on the Web, it may not be known which type of retrieval algorithm an information source uses, and it is unlikely that different sources will cooperate except in the most rudimentary manner. Even if they are willing to cooperate in these environments it is often difficult to detect whether the information sources provide correct information. These characteristics of *uncooperative environments* demand more sophisticated federated search solutions.

Most prior research focused on cooperative environments that assume different types of cooperation from information sources [Voorhees et al., 1995; Viles & French, 1995; Callan, 2000; Gravano et al., 1997; Kirsch, 1997]. A recent trend is to study the more complex situation of uncooperative environments for wide area networks or the Web [Si & Callan, 2002a; Ipeirotis & Gravano, 2002; Si & Callan, 2003a; Si & Callan, 2003b; Bergholz & Chidlovskii, 2004], which is the focus of this dissertation.

1.3 Applications of federated search

For visible Web contents, a previous study [Baeze-Yates & Ribeiro-Neto, 1999] has shown that users may prefer different search applications when they have different types of information needs. This is also true for federated search since there exist various federated search applications to satisfy different types of information needs, which may require different federated search applications.

The CompletePlanet portal[8] provides *structure guided browsing* of thousands of hidden information sources. It enables users to explore a wide range of hidden information sources that they are interested in. This browsing model works well when users have broad information needs. However, when users' information needs can be easily expressed as text queries and when users want to directly find relevant information, other choices such as the information source recommendation application or the federated document retrieval application are more appropriate.

Information source recommendation (e.g., the CompletePlanet portal and the IncyWincy invisible Web

[8] www.completeplanet.com

search engine[9]) goes a step further than the browsing approach by recommending most relevant information sources to information needs expressed as text queries. This type of application is very useful if users want to browse the selected information sources by themselves instead of asking the system to retrieve relevant documents automatically. It is also a more appropriate choice when user interaction is required to choose from multiple search configurations for specific information sources. An information source recommendation system is composed of two components, namely resource representation and resource selection.

A more complex federated search solution is *federated document retrieval*. It selects relevant information sources for user queries, as does the information source recommendation system. Furthermore, user queries are forwarded to search the selected information sources and finally the returned individual ranked lists are merged into a single list to present to the users. Therefore, federated document retrieval provides a more complete search solution by combining all the three components of federated search: resource representation, resource selection and results merging. It is a more complicated solution than information source recommendation. Systems like Metalib[10] have been developed within cooperative environments, but very little has been pursued for uncooperative environments.

1.4 The goal and contribution of this dissertation

Federated search has been a popular research topic for more than a decade and there has been considerable prior research. Most of the prior work is concentrated on the cooperative environments and relatively little can be applied in uncooperative environments. The research in this dissertation proposes new algorithms to address the three main subproblems of federated search in uncooperative environments.

Most past federated search research dealt with individual subproblems separately, but the field starts to realize that good solutions of different components optimize different criteria (e.g., high recall for information source recommendation; high precision for federated document retrieval). The inconsistency limits the effectiveness of good integrated solutions for federated search applications. Based on this observation, this dissertation proposes a unified probabilistic framework to integrate effective solutions for different subproblems together.

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

[10] http://dali.cdlib.org:8080/metasearch/nsdl/search.cgi

This section first describes new research in the three subproblems respectively and then discusses the unified framework.

For resource representation, previous research mainly focused on how to learn and describe the topics covered by each available hidden information source. This was accomplished by acquiring corpus statistics of information sources such as the vocabulary or term frequencies [Callan & Connell, 2001; Gravano et al., 1997]. Many of prior work only estimated the relative frequencies of different terms in the vocabulary, which is effective when all information sources are of similar sizes. However, within federated search environments that contain information sources of skewed source sizes, the absolute term frequencies per source or inverse document frequencies of different terms across available sources are very important for applications such as information source recommendation and federated document retrieval. To acquire this type of information, it is necessary to estimate the size of each information source (i.e., the number of documents).

It is easy to obtain information source size estimates in cooperative environments. But information source size estimation in uncooperative environments is a major unsolved problem until now. Previous research [Liu et al., 2001] required a huge amount of communication costs to estimate information source sizes especially for large information sources. In this dissertation, a more efficient Sample-Resample algorithm is proposed to utilize sampled documents from query-based sampling and estimate the information source size [Si & Callan, 2003a].

Besides content topics and information source sizes, there are other important properties of information sources, such as search engine retrieval effectiveness (i.e., the ability to retrieve relevant documents from individual sources for user queries) or information authority, which are very important for real world federated search applications. This dissertation proposes a new method to address the problem of estimating search engine retrieval effectiveness. The acquired information can be used to select information sources that are effective in returning the relevant documents that they contain.

For resource selection, most prior research represented each information source as a virtual big document [Yuwono & Lee, 1997; Callan, 2000; Craswell, 2000; French et al., 1999; Xu & Croft, 1999; Si et al., 2002b]. The similarities between user queries and big documents were calculated by using slight variations of well-known document retrieval algorithms to make the resource selection decision. We call this method as the "*big document*" approach. It works well when available information sources are of the similar sizes. However, the "big document" approach does not distinguish individual documents within available information sources so it often has a strong bias against either small hidden information sources or large information sources [Craswell, 2000; Si & Callan, 2003a; Si & Callan, 2003c] (more detailed information can be found in Chapter 4).

This dissertation describes a new resource selection algorithm that models information sources as document repositories instead of big documents. It explicitly estimates the distribution of relevant documents across available information sources for the information source recommendation application. It makes full use of the information source size estimates and the content descriptions acquired from the resource representation component [Si & Callan, 2003a]. This approach is not only more theoretically solid but also provides accurate empirical results within different types of federated search environments.

Results merging is the final step for a federated document retrieval system. It merges the individual ranked lists from selected information sources into a single final ranked list. This is a difficult task especially in uncooperative environments as different hidden information sources may use different retrieval algorithms or have different corpus statistics. Previous results merging methods either require cooperation from available sources [Viles and French, 1995; Xu and Callan, 1998; Kirsch, 1997], or approximate comparable scores with heuristic methods [Voorhees, 1995; Callan, Lu & Croft, 1995]. However, these methods either require cooperation that is not valid in uncooperative environments or are not very effective.

This dissertation proposes a Semi-Supervised Learning (SSL) results merging algorithm. This method uses sampled documents to create source-independent scores for a few representative documents from each information source. These documents serve as training data to build linear models that transform source-specific scores to corresponding source-independent scores. The linear models are applied on all the returned documents to calculate the comparable source-independent scores, and thus the final result list can be obtained with these source-independent scores. When there is not enough training data in the sampled documents, a variant of the SSL algorithm downloads a minimum number of documents on the fly to create additional training data [Si & Callan, 2002a; Si & Callan, 2003b]. The SSL algorithm has been shown to generate rather accurate final document rankings with a small cost.

It was common in prior research to view the three main subproblems of federated search in isolation from each other. The relationship between the subproblems has not been well studied, which is a serious problem to build integrated solutions for different federated search applications. For example, the resource selection algorithms for the information source recommendation application are generally designed and evaluated by how well they select information sources that contain as many relevant documents as possible (i.e., *high-recall* goal). However, prior research pointed out that a good resource selection algorithm for an information source recommendation system may not work well for a federated document retrieval system with the *high-precision* goal (i.e., more relevant documents at the top of final ranked list) [Craswell, 2000; Si & Callan, 2003a].

A central goal of the proposed research is the development of a formal federated search framework that integrates the solutions of different subproblems into an integrated framework. This approach allows a system to explicitly model and compensate for the inconsistencies between different goals. Specifically, when used for information source recommendation, the framework is optimized to maximize the utility as high recall, and when used for federated document retrieval, it is optimized for high precision. This unified utility maximization framework provides a more theoretical and unified foundation of federated search. Thorough empirical studies have shown that this unified framework produces more accurate results for both the information source recommendation application and the federated document retrieval application.

The shift to modeling federated solutions within a single unified framework also supports a broader set of evidence than just relevance, e.g., search engine retrieval effectiveness, information authority and reading difficulty. These factors can be very important for real world applications. For example, if the search engine of a selected information source is of very low quality, it may return very few relevant documents. In this dissertation, a resource selection algorithm is proposed to incorporate the factor of search engine retrieval effectiveness. The new algorithm selects information sources that not only contain many relevant documents but also are effective to return them.

Two main contributions distinguish this dissertation work from previous research. One key contribution of this dissertation shows the successful utilization of a *centralized sample database* (CSDB) containing the documents sampled from all available information sources [Callan, Connell & Du, 1999; Callan, 2000]. Most previous research [Craswell, 2000; Si et al., 2002b] discarded the sampled documents after building the resource descriptions. The research in this dissertation realizes that the centralized sample database is a valuable sample of the universe of documents available in a federated search environment. Its power exists in providing a uniform environment to compare representative documents from different information sources. This dissertation shows that the centralized sample database can be utilized in many problems of federated search and proves its success with empirical studies. Specifically: i) the probabilities of relevance of all the documents among hidden information sources are inferred from the probabilities of relevance of the sampled documents in the centralized sample database; ii) sampled documents from a small set of queries can be ranked by both the source-specific retrieval algorithm for a particular information source and an effective centralized retrieval algorithm; the consistency among the two ranked lists is utilized to measure the search engine retrieval effectiveness; and iii) the documents in the centralized sample database serve as training data to build query-specific and source-specific linear models to transform source-dependent scores into source-independent scores for results merging

Another key contribution of this dissertation is to turn away from the "big document" resource selection approach. The "big document" resource selection algorithms were the previous state-of-the-art methods. This dissertation analyzes and presents empirical results to show the deficiencies of the "big document" approach. Our new approach recognizes that it is necessary to model available information sources as document repositories and to estimate the probabilities of relevance of all the documents across sources for accurate resource selection decision. Some methods have been proposed in this dissertation to estimate the probabilities of relevance of the documents and then make appropriate resource selection decisions.

The research of federated search has attracted great attention for a long time. This dissertation advances the state-of-the-art in the main subproblems of federated search separately and also makes an important step forward to propose a unified framework to integrate the individual effective solutions together. The new framework is theoretically solid and is open to be extended for specific consideration of real operational applications. Detailed empirical studies and analysis in this dissertation have shown that the new solutions are very effective. In order to reflect the diversity of real world applications, the new federated algorithms have been evaluated within a range of research and real-world federated search environments that include a set of monolingual environments, a multilingual environment, environments with effective search engines and environments with both effective and ineffective search engines. All the contributions indicate that the new research in this dissertation is ready to be utilized in real world environments.

1.5 Outline

The dissertation is organized as follows. Chapter 2 describes our choices of federated search environments for evaluating different federated search algorithms, including how to create multiple federated search testbeds with newswire data, Web data and multilingual documents. Chapters 3 to 6 study research problems of individual components within federated search applications. Specifically, Chapter 3 discusses the research problem of resource representation. It introduces several important types of resource representations and also proposes new algorithms for estimating information source sizes. Chapter 4 focuses on the research problem of resource selection. It reviews several previous state-of-the-art algorithms, analyzes the deficiency and proposes new resource selection algorithms that explicitly estimate the distribution of relevant documents for optimizing the high-recall goal. Chapter 5 studies the research problem of results merging. It analyzes the disadvantage of previous results merging algorithms and proposes new results merging algorithms based on estimating regression models for mapping source-specific document scores to comparable document scores. Chapter 6 further investigates how to merge document ranked lists in multiple languages.

A unified utility framework is proposed in Chapter 7 to integrate and adjust effective solutions of main components of federated search in a single framework and optimize various goals of different federated search applications. This is shown to be a better choice than simply combining effective solutions together. The unified framework enables the modeling ability of considering a wide range of evidence other than relevance. Chapter 8 shows how to extend the unified utility model to incorporate search engine retrieval effectiveness, which is a very important issue for real world federated search applications.

Finally, Chapter 9 concludes the dissertation by summarizing the contribution and pointing out several future research directions.

Chapter 2: Modeling Federated Search Problems

This chapter describes our choices of federated search environments for evaluating different federated search algorithms. A real world federated search application is first introduced and the possibilities and limitations of evaluation with this application are discussed. Next, a range of research environments is presented to simulate various operational federated search environments. The research environments include newswire data, Web data, and multilingual data. The experimental methodology of modeling multiple types of search engines of various qualities is also addressed. Finally, this chapter briefly introduces experimental metrics for evaluating different federated search applications.

2.1 The FedStats portal: federated search in the real world

The best candidates to evaluate federated search algorithms are real world applications. The FedStats Web site[11] is an information portal that provides "one stop" search of statistical information published by many federal agencies so that citizens, businesses, and government employees can find useful information without separately visiting Web sites of individual agencies. The existing solution copies and indexes the contents of more than one hundred information sources into a single centralized database. However, the centralized index becomes outdated because the efforts of crawling and updating all the contents very frequently are not affordable (e.g., FedStats crawls each site about every three months). Thus, a federated search solution was requested and this was the main focus of the FedLemur project [Avrahami et al., 2006].

The initial FedLemur system provides federated search solution for 20 agency sites. The list of the Web sites is shown in Figure 2.1. 20 wrappers were manually created to connect to individual Web sites for submitting user queries and fetching retrieved results. From a technical perspective, building wrappers and adding new sites is easy for the 100 agency sites connected by the FedStats portal. In the long run, automatic construction

[11] http://search.fedstats.gov

- **Bureau of Economic Analysis (BEA):** http://www.bea.doc.gov/
- **Bureau of Justice Statistics (BJS):** http://www.ojp.usdoj.gov/bjs/
- **Bureau of Labor Statistics (BLS):** http://www.bls.gov/search/search.asp
- **Bureau of Transportation Statistics (BTS):** http://www.bts.gov/
- **Energy Information Administration (EIA):** http://www.eia.doe.gov/
- **Environmental Protection Agency (EPA):** http://www.epa.gov/epahome/search.html
- **USDA's Economic Research Service (ERS):** http://www.ers.usda.gov/
- **Federal Reserve Board (FRB):** http://search.federalreserve.gov/
- **Housing and Urban Development (HUD):** http://www.huduser.org/
- **Internal Revenue Service (IRS):** http://search.irs.gov/web/advanced-search.htm
- **International Trade Administration (ITA):** http://www.ita.doc.gov/td/industry/otea/
- **USDA's National Agricultural Statistics Service (NASS):** http://www.usda.gov/nass/
- **National Center for Education Statistics (NCES):** http://nces.ed.gov/
- **National Center for Health Statistics (NCHS):** http://www.cdc.gov/nchs/search/search.htm
- **National Institute of Child Health and Development (NICHD):** http://www.nichd.nih.gov/search.cfm
- **NSF Science Resources Statistics (NSF):** http://www.nsf.gov/sbe/srs/search.htm
- **Social Security Administration Office of Policy (SSA):** http://www.ssa.gov/policy/
- **U.S. Census Bureau:** http://www.census.gov/main/www/srchtool.html
- **Childstats:** http://www.childstats.gov/sitesearch.asp

Figure 2.1: 20 agency Web sites that are connected by the FedLemur system.

Table 2.1: 27 queries with relevance judgments used for evaluating results merging algorithms in the FedLemur system.

abortion	federal grants	obesity
bankruptcy	gross domestic product	religion
bmi	hate crimes	suicide
car accidents	health insurance	teen pregnancy
consumer price index	homeless	tourism
consumer spending	immigration	tqm total quality management strategies
cost of living	largest employers	unemployment rate
crime rates	life expectancy	welfare
domestic violence	literacy	women and employment

of wrappers (i.e., wrapper induction) [Gazen and Minton, 2005][He et al., 2006] is a more viable solution, but that is outside the scope of this dissertation.

27 test queries shown in Table 2.1 were utilized in the evaluation. These queries were selected from the query logs of FedStats Web portal by members of the FedStats team. The selection criterion was based on topic breadth and frequency in query logs. For each test query, a resource selection algorithm (CORI, described in Chapter 4) was applied to select 3 or 5 most relevant resources. The selected information sources were searched and 35 or 50 documents were returned from each source. The varying number of sources and retrieved documents was not an ideal experimental methodology, but it reflects evaluation in an operational environment; the government employees doing relevance assessments were busy and only had

time to judge a short list of 105 documents (i.e., 35 documents each from 3 sources). Returned documents from various information sources were sorted randomly and judged by members of the CMU and FedStats teams.

In this dissertation, the focus is to evaluate the accuracy of different results merging algorithms with the FedLemur system by utilizing the set of queries and human relevance judgments described above. This was easy to do in an efficient way. It would also be possible to evaluate resource selection algorithms with the FedLemur system, but this would require crawling all the sites completely and pooling the documents for human relevance judgments, which is not explored in this dissertation.

The FedLemur system represents federated search solutions for information sources within a large organization. A similar example is the West system[12] which connects to thousands of legal, financial and news information sources [Conrad, 2002]. The FedLemur system and the West system share similar characteristics, such as: i) the information is scatted over different information sources due to either the maintenance and policy issues or technique difficulties; ii) the information sources are often created and maintained by different providers and the overlap among their contents tends to be low; iii) the information sources contain a relatively large number of documents, for example, most information sources within the FedLemur system contain from 2,000 to more than one 100,000 documents; iv) the contents of the information sources are often carefully written and edited by professionals, so the quality is high; and v) the information sources often focused on several specified topics (e.g., agency reports within the FedStats system or legal, financial and news documents within the West system). These similarities between the FedLemur and the West systems are common characteristics of federated search solutions in large organizations or large companies as well as of federated search systems for domain-specific hidden Web.

2.2 Simulating real world applications with TREC Data

Evaluation with real world applications is the most desired way to test federated search algorithms. However, it is hard to conduct large scale user studies or to obtain full control of the systems. An alternative method is to simulate the operational environments of federated search systems by utilizing existing large text collections with thorough relevance judgments.

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

Thorough evaluation of federated search algorithms requires a sufficient number of information sources, enough queries, and relevance judgments. It is often expensive to meet all these requirements in the user studies of real world applications. In contrast, existing large text collections such as the TREC data provide the opportunity to simulate real world federated search environments.

TREC (Text REtrieval Conference) [13] [Harman, 1995] is conducted by the National Institute of Standards and Technology (NIST). The goal of this conference is to encourage research in information retrieval for text applications by providing large test collections including large corpora with sufficient queries and relevance judgments [Harman, 1995]. The creation of TREC data provided a good opportunity of conducting federated search experiments to simulate the environments with large numbers of rather diverse information sources, distributed geographically and maintained by many parties. For example, the database merging track of the TREC-4 conference was one of the earliest explorations of federated search [Harman, 1995].

2.2.1 Simulation with TREC news/government data

TREC news/government data covers topics in several areas. The documents in the corpora are written by professionals. These characteristics are similar to those of the FedLemur system or the West system. Therefore, this type of data provides a good opportunity to simulate the federated search environments of large organizations or domain-specific hidden Web.

A common strategy to create federated search testbeds in previous research was to partition TREC news/government corpora by source, date, and/or topic into many smaller information sources with reasonable sizes and homogeneous contents [Lu et al., 1996; Xu & Callan, 1998; French et al., 1999; Hawking & Thistlewaite, 1999; Callan, 2000; Si & Callan, 2002a; Ipeirotis & Gravano, 2004]. This approach has several advantages: i) news/government documents are representative of the contents provided by professionally-written information sources; ii) testbeds are composed of many information sources, each containing thousands of documents on average, which is more realistic than testbeds of only several information sources [Fox et al., 1992; Yuwono & Lee, 1997] or testbeds of small information sources [Craswell, 2000]; and iii) a large body of previous research has reported experiment results on the testbeds of TREC news/government collections [Xu & Callan, 1998; French et al., 1999; Hawking & Thistlewaite, 1999; Callan, 2000; Si & Callan, 2002a], which provided baselines for evaluating the effectiveness of the new algorithms. These characteristics make the TREC news/government data a good candidate to simulate the federated search environments of large organizations or domain specific hidden Web.

[13] http://trec.nist.gov

Table 2.2: Testbed statistics of Trec123_100Col and Trec4_kmeans testbeds.

Name	Number of Sources	Query Count	Size (GB)	Number of Docs			Megabytes (MB)		
				Min	Avg	Max	Min	Avg	Max
Trec123_100Col	100	100	3.2	752	10782	39713	28.1	32	41.8
Trec4_kmeans	100	50	2.0	301	5675	82727	3.9	20	248.6

Table 2.3: Query set statistics for Trec123_100Col and Trec4_kmeans testbeds.

Name	TREC Topic Set	TREC Topic Field	Average Length (Words)
Trec123_100Col	51-150	Title	3.1
Trec4_kmeans	201-250	Description	7.2

Specifically, two commonly-used testbeds organized by different criteria are chosen in this dissertation, as described below.

(i) **Organized by source and date** (**Trec123_100Col**): 100 information sources were created from TREC CDs 1,2,3. They were named by source and publication date. Documents were assigned to information sources based on source and publication date. 100 short queries extracted from the title fields of TREC topics 51-150 were associated with this testbed [French et al., 1999; Callan, 2000; Si & Callan, 2002a].

(ii) **Organized by topic** (**Trec4_kmeans**): 100 information sources were created from TREC 4 data. A k-means clustering algorithm was used to automatically cluster the documents by topic, and then each information source was associated with one cluster. The contents of the information sources are homogenous and the word distributions are skewed. 50 longer queries were created from the description fields of the TREC topics 201-250 [Xu & Croft, 1999; Si & Callan, 2002a].

Summary statistics for these two testbeds are shown in Table 2.2 and the characteristics of their corresponding queries are shown in Table 2.3.

There are several other choices to construct testbeds from the TREC news/government data [Lu et al., 1996; Xu & Callan, 1998; French et al., 1999; Hawking & Thistlewaite, 1999; Callan, 2000; Si & Callan, 2002a; Ipeirotis & Gravano, 2004]. For example, Fox et al. [Fox et al., 1992] used 5 collections from TREC CD 1 and French et al. [French et al., 1998] partitioned TREC CDs 1,2,3 with finer granularity into 236 collections. Compared with those choices, Trec123_100Col is more widely used [French et al., 1999; Callan,

Table 2.4: Statistics for the large databases.

Database	LDB1	LDB2	APall	WSJall	FRall	DOEall
Number of documents (x 1,000):	231.3	199.7	242.9	173.3	45.8	226.1
Size (MB):	665	667	764	533	492	194

2000; Si & Callan, 2002a; Nottelmann & Fuhr, 2003b] and there are many baseline results available for this testbed. Trec4_kmeans is organized by topic, which provides another direction to simulate various federated search environments.

Our experience with the FedLemur project suggested that it is important to evaluate federated search algorithms in environments containing many "small" information sources and a few "very large" sources. However, it can be seen from Table 2.2 that the Trec123_100Col testbed has a relatively uniform information source size distribution and the Trec4_kmeans testbed has modest skewed source size distribution. Furthermore, another interesting direction is to vary the distribution of relevant documents among information sources and study the effectiveness of different algorithms with this configuration. Following these ideas, five more testbeds were created based on the Trec123_100Col testbed. Each of them contains many "small" information sources and two large information sources that are about an order of magnitude larger than other sources [Si & Callan, 2003a].

Trec123_2ldb_60Col ("representative"): The resources in the Trec123_100Col testbed were sorted alphabetically. Every fifth source, starting with the first, was merged into one large source called LDB1. Every fifth source from, starting with the second, was merged into another large source called LDB2. The other 60 sources were left unchanged. This testbed simulates environments with bimodal source size distributions where large sources have about the same densities of relevant documents as the small ones (the two large sources still have more relevant documents due to their large sizes).

Trec123_AP_WSJ_60Col ("relevant"): The 24 Associated Press information sources in the Trec123_100Col testbed were combined into a large APall information source, while sixteen Wall Street Journal collections were collected into a large WSJall source. The other 60 small sources were unchanged. This testbed simulates environments of bimodal source size distributions where large sources have higher densities of relevant documents than small ones.

Trec123_FR_DOE_81Col ("nonrelevant"): The 13 Federal Register information sources in the Trec123_100Col testbed were collapsed into a large FRall information source, while the 6 Department of Energy information sources were merged into a large DOEall information source. The remaining 81 information sources were unchanged. This testbed simulates environments of bimodal source size distributions where large sources have lower densities of relevant documents than small ones.

Trec123_10Col: The representative, relevant and nonrelevant testbeds contain many small sources and two very large sources. For some federated search applications (e.g., information source size estimation), it is important to test their performance in a federated search environment with many large information sources. The Trec123_10Col testbed was created to accomplish this goal. This testbed contains 10 large information sources. Particularly, the information sources in the Trec123_100Col testbed were sorted alphabetically. Every tenth source, starting from the first one, was merged into the first new source. Every tenth source, starting from the second one, was combined into the second new source, and so on [Si & Callan, 2003a].

Trec123_2Col: This testbed was built by merging the 100 sources in Trec123_100Col into two very large information sources in a round robin way. It is utilized to evaluate information source estimation algorithms with very large information sources.

2.2.2 Simulation with TREC Web data

TREC news/government data can be utilized to simulate the federated search environments of large organizations or domain specific hidden Web. However, it is not appropriate to simulate the federated search environments such as the open domain hidden Web. Some specific reasons are: i) the federated search environments of open domain hidden Web generally have more diverse contents or more genres; and ii) the federated search environments of open domain hidden Web tend to have a larger number of hidden information sources.

Web data is a better choice to simulate the open domain hidden Web environments. There exist large collections of TREC Web data acquired by Web crawlers. The TREC Web data provides us a good opportunity to simulate the federated search environments with a large number of information sources such as the open domain hidden Web. The scales of the federated search systems for open domain hidden Web can be much larger than the scales of federated search systems in large organizations or companies. Furthermore, it can be imagined that the contents of information sources in open domain hidden Web environments are also very diverse and contain more genres. This characteristics serve as a guidance to build testbeds that simulate federated search environments of open domain hidden Web.

The TREC Web collection WT10g [Craswell, 2000; Lu & Callan, 2003] contains about 10 gigabytes of documents crawled from 11,485 Web sites. Each Web site can be considered equivalent to an information provider, which makes this testbed a good candidate to simulate large-scale federated search applications.

Table 2.5: Statistics for WT10g testbed.

Name	**Number of Sources**	**Query Count**	**Size (GB)**	**Number of Docs**			**Megabytes (MB)**		
				Min	**Avg**	**Max**	**Min**	**Avg**	**Max**
WT10g	934	100	7.8	300	1169	26505	0.3	8.4	161

Table 2.6: Query set statistics for WT10g testbed.

Name	**TREC Topic Set**	**TREC Topic Field**	**Average Length (Words)**
WT10g	451-550	Title	2.6

However, one weaknesses of this testbed is that many information sources contain very small number of documents (about two thirds sources contain fewer than 1,000 documents). This characteristic is different from those observed from the FedLemur project [Avrahami et al., 2006], where most information sources contain reasonable amount of documents (i.e., more than 1,000) with valuable topic-oriented contents (government agency reports).

In this dissertation, the TREC WT10g collection was divided into many information sources by considering each Web server to be a distinct information source. However, small information sources that contain very few documents were filtered out. Specifically, 934 sources were obtained by dividing WT10g data into 11,485 collections and selecting those that contain more than 300 documents. More detailed information is shown in Table 2.5. TREC Web queries 451-550 were used on this testbed. The summary statistics of these queries can be found in Table 2.6. Those short queries reflect the fact that 85% of the queries posted at Web search engines have 3 or fewer query terms [Jansen et al., 2000].

In order to provide thorough experimental results for information source size estimation algorithms, several testbeds were created from TREC Web data as follows:

WT10g_10%, WT10g_30%, WT10g_50%, WT10g_70%, WT10g_90% and WT10g_100%: The WT10g_10%, WT10g_30%, WT10g_50%, WT10g_70% and WT10g_90% testbeds were created by randomly selecting 10%, 30%, 50%, 70% or 90% of all the documents within the WT10g testbed respectively. The WT10g_100% testbed was created by collapsing all the documents into a single information source.

GOV_10%, GOV_30%, GOV_50%, GOV_70%, GOV_90% and GOV_100%: The .GOV test collection was created in 2002 by crawling more Web sites within the domain of .gov. It contains 1.25 million

Table 2.7: Statistics for CLEF 2005 Multi-8 testbed.

Language	Dutch	English	Finnish	French	German	Italian	Spanish	Swedish
Number of documents:	190,604	169,477	55,344	129,806	294,809	157,558	454,045	142,819
Size (MB):	551	599	139	335	388	369	1,132	369

Table 2.8: Query set statistics for CLEF 2005 Multi-8 testbed.

Name	CLEF Set	CLEF Topic Field	Average Length (Words)
Multi-8 (Train)	141-160	Title+Desc	12.0
Multi-8 (Test)	161-200	Title+Desc	11.7
Multi-8 (All)	141-200	Title+Desc	11.8

documents and has a size of 18.1 GB. These testbeds were created from the .GOV test collection by either randomly selecting some amount of documents or choosing all the documents.

2.3 Simulating multilingual environments with CLEF data

The TREC text collections introduced in this dissertation contain documents in a single language (English). However, in real world hidden Web environments, many information sources are composed of documents in other languages. For example, the German Patent and Trademark Office Web site contains a large number of patents in several European languages; those patents can only be accessed by posing queries in different languages. The task of accessing hidden information sources in multiple languages requires us to develop a multilingual federated search solution. This section discusses how to build a federated search environment with multilingual information sources.

The Cross-Language Evaluation Forum (CLEF)[14] is co-sponsored by the European Commission and several American and European companies. The goal of this forum is to develop an infrastructure for the testing, tuning and evaluation of information retrieval systems operating on European languages mainly in cross-language contexts, and to create test-suites of reusable data that can be employed by system developers for benchmarking purposes. A large number of multilingual documents, queries and human relevance judgments have been accumulated in different evaluation tasks of the CLEF campaign. Particularly, eight

[14] http://www.clef-campaign.org/

information sources were built for eight different languages. The statistics of the eight multilingual sources can be seen in Table 2.7. CLEF 2005 provided relevance judgments of 20 training queries for tuning different algorithms [Nunzio et al., 2005]. All algorithms were formally evaluated on another set of 40 test queries. The summary statistics of these queries are shown in Table 2.8.

This CLEF data provides a good opportunity to simulate multilingual federated search environments. As the CLEF queries are intentionally written to retrieve documents in most languages, the CLEF data is not a good choice for evaluating resource selection tasks. Therefore, a testbed is created from CLEF data for evaluating results merging algorithms within a multilingual federated search environment. However, it is more appropriate for results merging experiments. The choice of creating one database per language is appropriate because results merging is typically conducted over a small set of result lists.

2.4 Simulating multiple types of search engines

There exist multiple types of search engines in uncooperative federated search environments. Three different types of effective retrieval algorithms are used in our experiments: INQUERY [Turtle 1990; Callan, Croft & Harding, 1992], a statistical language model algorithm [Lafferty & Zhai, 2001; Ogilvie & Callan, 2001b], and a vector-space algorithm similar to SMART [Buckley et al., 1995]. These three algorithms are generally considered effective and are widely used in ad-hoc retrieval systems.

The INQUERY algorithm [Turtle 1990; Callan, Croft & Harding, 1992; Broglio et al., 1995] adapts an Okapi term frequency normalization formula [Robertson & Walker, 1994] in a Bayesian inference network model to rank the documents. Formally, the belief of the jth document according to the term q is expressed as follows:

$$T=\frac{tf}{tf+0.5+1.5*doclen_j/avg\_doclen} \qquad (2.1)$$

$$I=\frac{\log\left(\frac{|d|+0.5}{df}\right)}{\log(|d|+1.0)} \qquad (2.2)$$

$$p(\, q|d_j)=b+(1\text{-}b)*T*I \qquad (2.3)$$

where: tf is the number of occurrence of q in this document;

df is the number of documents that contain q;

|d| is the number of documents in the corpus;

doclen_j is the number of words in this document;

avg_doclen is the average document length in the corpus; and

b is the default belief, usually set to 0.4.

The belief P (Q | d_j) is calculated by combining evidence as P(q | d_j) from different query terms. This can be achieved by applying different probabilistic operators. The INQUERY operators cover a wide range of Boolean, proximity and synonym operators. More detailed information can be found in [Turtle, 1990; Broglio et al., 1995; Callan, 2000]. In this dissertation, the belief P (Q | d_j) is calculated by averaging the evidence of individual query words. This score can vary in the range of [0.4 , 1.0], but typically falls within [0.4 , 0.7].

The basic idea of the statistical language model retrieval algorithm [Lafferty & Zhai, 2001; Ogilvie & Callan, 2001b] is to treat each document as a multinomial distribution of the words in the vocabulary. It ranks documents by how likely they can generate a particular query. Formally, the generation probabilities of documents for query Q are calculated as:

$$P(Q|d_j)=\prod_{q\in Q}(\lambda P(q|d_j)+(1\text{-}\lambda)P(q|G)) \qquad (2.4)$$

The Jelinek-Mercer smoothing is used to generate the document language model. It is a linear combination of the maximum likelihood document model (i.e., P (q | d_j)) and a global collection language model (i.e., P (q | G)), which are calculated based on the relatively frequencies of term q in the j^{th} document and in the global corpus respectively. The global corpus is created by combining all the documents together. The coefficient λ controls the influence of each model and is set to 0.5 in our experiments. P (Q | d_j) is usually a very small positive number and the logarithm of this value gives us a final document score, which is often in the range of [-60 , -30].

The vector-space retrieval algorithm in this dissertation uses the SMART “lnc.ltc” weighting scheme [Buckley et al., 1995]. The logarithmic version of term frequency and the cosine normalization is used by both query and document representations. The query representation utilizes logarithmic idf weight and the document representation does not. The similarity between the query representation and the document representation is calculated as:

$$\mathrm{Sim}(Q,d_j)=\frac{\sum \log(tf+1)*\left(\log(qtf+1)\log\frac{|d|}{df}\right)}{\sqrt{\sum \log(tf+1)^2}\sqrt{\left(\log(qtf+1)\log\frac{|d|}{df}\right)^2}} \qquad (2.5)$$

where qtf stands for the term frequency of a specific query term. The document scores of this retrieval algorithm fall into the range of [0.0 , 1.0].

All the above three retrieval algorithms are effective and can be used for simulating effective search engines. However, inaccurate search engines are also common in real world environments. For example, the well-known PubMed[15] system uses an unranked Boolean retrieval algorithm. Ineffective government search engines have also been observed that return unranked or randomly ranked results, or return many documents that do not exist, as in the case of broken links.

In order to simulate the behavior of ineffective search engines in real world applications, three types of ineffective retrieval algorithms are introduced in this work: an INQUERY retrieval algorithm with added random noise to the original retrieval scores, where the random noise ranges from 0 to 0.3 (the original scores range from 0.4 to 1); an extended Boolean retrieval algorithm, which adds up the term frequencies of query terms without considering the idf factor; and a unigram language model with bad linear smoothing parameter λ, which is set to be 0.99 to bias towards the collection language model.

All these retrieval algorithms are implemented with the Lemur toolkit[16] [Ogilvie & Callan, 2001b], and they are usually assigned to the information sources in a round-robin manner.

2.5 Federated search experimental metrics

[15] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi

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

This dissertation studies several federated search applications. The solutions of these applications are evaluated with a variety of experimental metrics, which mainly follow the procedures established by previous research [Liu et al., 2001; French et al., 1999; Callan, 2000].

An information source recommendation system suggests relevant information sources for users and then users will visit the selected sources and browse the contents manually. Users often prefer information sources that contain as many relevant documents as possible. Therefore, the resource selection algorithms of an information source recommendation system are measured by the goal as *high recall* that is analogous to the recall metric for ad-hoc retrieval. Specifically, the information source ranking of a specific resource selection algorithm is compared with that of a desired source ranking, which ranks information sources by the amount of relevant documents that they contain for a user query. The smaller the difference between the ranking of the specific algorithm and the desired ranking is, the more effective is the algorithm [French et al., 1999; Callan, 2000].

For federated document retrieval, the system automatically searches the selected information sources and merges the returned ranked lists from different sources into a single ranked list before presenting it to the end users. Most users only concentrate on the documents at the top of the final ranked lists. Therefore, users often evaluate the effectiveness of a federated document retrieval system by the precision at the top of the final ranked list [Xu & Croft, 1999; Callan, 2000]. This is formally denoted as the *high-precision* goal in this work.

2.6 Summary

It is important to design representative federated search environments to evaluate different federated search algorithms. This chapter describes our choices of federated search environments and briefly introduces some experimental methodology.

The chapter first introduces a real world federated search application, the FedStats portal, which connects to 20 government agencies with uncooperative search engines. This real world application is mainly utilized to evaluate results merging algorithms within the federated search environment. A set of 27 real world queries was selected and human relevance judgments were obtained for the evaluation.

As there are many constraints for doing experiments with real world federated search applications, a set of federated search testbeds was created within research environments to conduct thorough evaluation. Particularly, we use a set of testbeds from the TREC news/government data to simulate domain-specific federated search environments and use another set of testbeds from the TREC Web data to simulate open-domain Web-like federated search environments. These testbeds are associated with different characteristics (e.g., source size distribution, relevant document distribution and writing quality).

This chapter also introduces a set of multilingual federated search environments. These testbeds are important to evaluate federated search algorithms when available information sources contain documents in different languages. Particularly, a set of language-specific information sources was created from CLEF data to simulate multilingual federated search environments.

Finally, several experimental metrics are introduced in this chapter for evaluating different federated search applications. It is pointed out that information source recommendation is evaluated by the number of relevant documents contained in selected sources (i.e., high recall), while federated document retrieval is evaluated by the number of relevant documents ranked at the top of the final ranked lists (i.e., high precision).

Chapter 3: Adding Size to Resource Representations

Acquiring accurate resource descriptions of available information sources is the first step for every federated search system. There are two problems that need to be addressed for acquiring accurate and comprehensive resource descriptions in an efficient way: i) what types of resource descriptions are required in order to accomplish the federated search tasks; and ii) for each type of resource description, how can it be obtained efficiently. Previous research on resource description was mainly focused on representing each information source by a description of its words and the word frequencies. There exist good solutions for discovering word histogram representations from previous research. This chapter introduces information source sizes as another type of resource description. Specifically, this chapter motivates why information size estimation is important for federated search applications and discusses related prior research. Furthermore, a new algorithm is proposed to calculate the source size estimates more efficiently and empirical studies have been conducted to show its effectiveness and efficiency.

3.1 Previous research on resource representation

This section shows the previous research on resource representation from two aspects: resource description constructed by words and their occurrences and the size estimates of information sources.

3.1.1 Representation of Contents

Most prior research represented each information source by a description like term frequencies of the words that occur in the information source, and the word frequencies [Gravano, 1994; Gravano & García-Molina, 1995; Callan, Lu & Croft, 1995] or other statistics derived from the word frequencies such as the term weights [Gravano & García-Molina, 1995]. The description is possible to be extended for other indicative text features such as phrases or proper names. This type of resource description catches the content topics within each information source.

The most desirable scenario to acquire this type of resource description is when every information source shares its corpus statistics in a cooperative manner. The Stanford Protocol Proposal for Internet Retrieval and Search (i.e., STARTS) [Gravano et al., 1997] is a complete protocol of federated search in cooperative environments. It covers many topics from resource description acquisition to results merging. The source metadata acquisition part of the STARTS protocol obtains the information about information sources' contents and other features. More specifically, each information source is required to provide both the content summary containing information such as vocabulary and word frequencies and also other metadata indicating the properties of the information source such as stopwords, document score range and the type of retrieval algorithm.

However, cooperative protocols do not work in uncooperative federated search environments such as large organizations or the Web. In these environments, it is generally difficult to assume that the information sources can cooperate to provide resource representations that are compatible with each other. Furthermore, even if the information sources are willing to share their information, it is not easy to judge whether the information they provide is accurate or not.

An alternative method, which works in uncooperative environments, is the query-based sampling approach. This solution generates and submits single word queries to each information source and downloads some documents in the returned document ranked lists to learn the resource content descriptions [Callan, Connell & Du, 1999; Callan & Connell, 2001]. Specifically, an initial sampling query is selected randomly from a list of English words. Subsequent sampling queries were selected randomly from the resource description being learned for the information source (i.e., a term seen in a document retrieved earlier from that source). The only assumption made by the query-based sampling method [Callan & Connell, 2001] is that all the information sources run queries and return documents. It does not require information sources to provide content information or use a particular type of search engine cooperatively.

Experiments have shown that under a variety of conditions the query-based sampling method can acquire rather accurate content description for each hidden information source by using about 80 queries to download a relatively small number of documents (i.e., 300 documents) [Callan, 2000; Callan & Connell, 2001; Craswell et al., 2000]. More specifically, after obtaining 250 sampled documents, about 80 percent of the term occurrences in an information source can be covered by the words in the sampled data [Callan, 2000]. The Spearman rank correlation coefficient [Press et al., 1992] has also been utilized to measure the similarities between two rankings of the sampled df values and the actual df values. It is shown to be about 0.7 after acquiring 250 sampled documents on several testbeds, which has a maximum value of 1.0. Some variants of the query-based sampling techniques are the focused probing method [Ipeirotis & Gravano, 2002], which utilizes query probes pre-derived from rule-based classifiers of a hierarchy of topics, and the probe queries method [Craswell et al., 2000], which uses multi-term queries chosen from query log.

3.1.2 Representation of source size

Information source size is another important property of a hidden information source. It is easy to acquire source sizes within cooperative federated search environments. However, this is a much more difficult problem within uncooperative environments. Source size estimates are very important for federated search subproblems such as resource selection and results merging. For example, resource selection algorithms need source size estimates to adjust (normalize) the information source selection scores for accommodating the widely varying information source size distributions [Si & Callan, 2003b]. However, this was rarely used within prior research due to the difficulty of acquiring good size estimates efficiently. Information source size can be defined in many different ways such as the size of the vocabulary, the number of word occurrences and the number of documents. In this work, we define information source size to be the number of documents. Other related statistics such as the number of words can be estimated from the number of documents and other statistics obtained from the sampled documents.

Liu and Yu [Liu et al., 2001] proposed a basic Capture-Recapture algorithm to estimate the information source size statistics. This method follows previous work in the statistics community of estimating wild animals population size. Specifically, the algorithm assumes that two independent documents id lists can be obtained from a particular information source (e.g., by running two different queries). Let N denote the actual information source size, A be the event that a document id is included (captured) in the first sample, which contains altogether n_1 document ids, B be the event that a document id is in the second sample, whose size is n_2, and m_2 is the number of document ids that are in both samples. The probabilities of events A and B can be calculated as follows:

$$P(A)=\frac{n_1}{N} \qquad P(B)=\frac{n_2}{N} \tag{3.1}$$

The conditional probability that a document id appears in the second sample given it is observed in the first sample is:

$$P(B|A)=\frac{m_2}{n_1} \tag{3.2}$$

The two samples are assumed to be independent as follows:

$$P(B|A) = P(B) \tag{3.3}$$

Then the size estimate $\widehat{N}$ for this information source can be obtained as:

$$\frac{n_2}{\widehat{N}} = \frac{m_2}{n_1} \quad \Rightarrow \quad \widehat{N} = \frac{n_1 n_2}{m_2} \tag{3.4}$$

Liu and Yu reported that the basic Capture-Recapture method can acquire rather accurate information source size estimates (i.e., the error rate is about 5% for the size estimate of an information source with 300,000 documents) [Liu et al., 2001]. However, their method was not efficient. The basic Capture-Recapture method used a large number of sampled queries (i.e., about 1,000 queries) which required a large set of document ids for each query (i.e., each query retrieved 1,000 document ids) to estimate the size of an information source with 300,000 documents. The empirical studies in Section 3.4 show that the accuracy of the Capture-Recapture method degrades substantially when fewer queries are used or when only smaller ranked lists are available.

3.2 Centralized sample database

Many federated search systems only utilize the sampled documents to obtain resource descriptions and then discard the sampled documents. However, the sampled documents are valuable for other purposes. The sampled documents from all available information sources can be combined into a single searchable database called the *centralized sample database*. The centralized sample database is very important for federated search applications. In federated search environments, it is not possible or practical to copy all the searchable information into a single centralized database (i.e., *centralized complete database*) as conventional search engines do. The centralized sample database is a surrogate that can be used to simulate the behavior of the centralized complete database, as shown in Figure 3.1.

Ogilvie and Callan's work [Ogilvie & Callan, 2001a] was the first research to utilize a centralized sample database. Although their attempt at using a centralized sample database for query expansion was not successful, the centralized sample database may be an important resource for many other problems.

Many research problems in federated search can be generalized as learning transformations from source scores/ranks to scores/ranks in a centralized environment. The centralized sample database can be used to simulate the behavior of the centralized complete database and thus opens the door to a class of

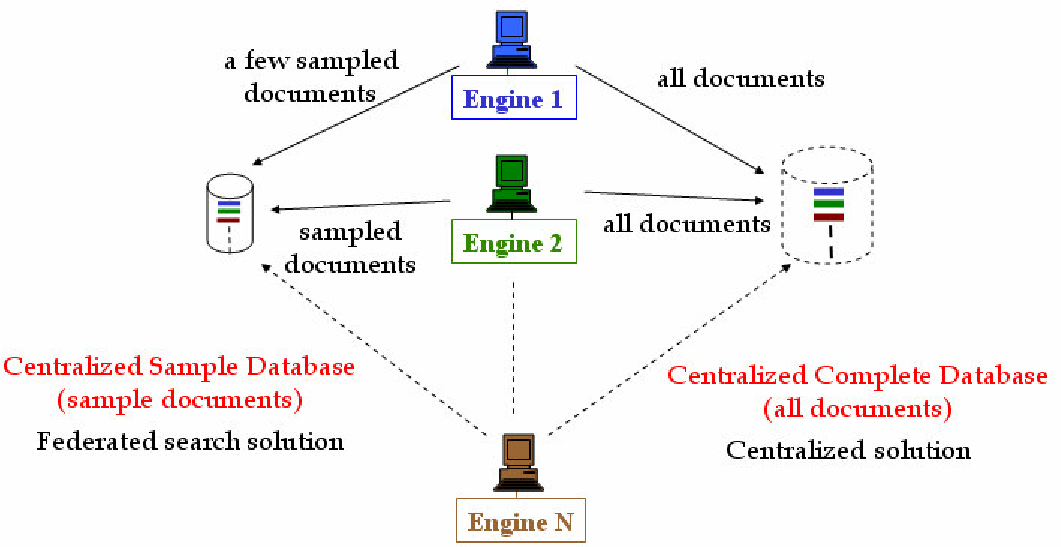

Figure 3.1: The importance of centralized sample database to simulate the characteristics of centralized complete database in a federated search environment.

transformations. There are multiple types of transformations such as score to score, score to rank, rank to score and rank to rank. The research in this dissertation utilizes different types of transformations for different applications. Two examples are briefly described below.

The major research problem of results merging is to make the ranked lists returned from selected information sources comparable and merge them into a single list. The centralized sample database provides an environment in which comparable scores can be calculated for documents from different information sources. The regression based results merging algorithm in Chapter 5 uses *a small set of* documents that have both source-specific document scores and source-independent document scores from the centralized sample database as training data for learning how to transform source-specific scores to source-independent scores. The learned models can be applied to all the returned documents from selected sources to merge the documents into a singe list. Furthermore, when search engines of selected sources do not return document scores (e.g., in FedLemur system), a model is proposed to transform document ranks in source-specific ranked lists to source-independent scores. More detailed information can be found in Section 5.3.5.

Another type of transformation is used in Chapter 8 to model search engine retrieval effectiveness. This method utilizes the sampled document ranked lists for a set of training queries to build models of search engine retrieval effectiveness. The method obtains higher accuracy when the assumption is true: the rank patterns of the ranked lists of the training queries are representative of those of the test queries. Specifically, for a source-specific ranked list of a training query, the documents in the list are downloaded and a new ranked list is created by an effective centralized retrieval algorithm using the centralized sample corpus statistics. A rank transform mapping can be learned between these two lists. The learned rank mappings for a set of ranked lists can be collected to build the search engine effectiveness profile for the particular search engine. Intuitively, if the source-specific search engine always generates ranked lists consistent with those from the centralized retrieval algorithm, the search engine is effective, otherwise it is not.

Besides using the centralized sample database to build different transformation models, the resource selection algorithm described in Chapter 7 utilizes the centralized sample database in another way. This method runs a user's query on the centralized sample database to produce a ranked list that is *a sample* of the ranked lists that would be created if all sources were searched. This sampled document ranking can be scaled up, using information about the size of each available source, to obtain an accurate estimate of how all of the documents (mostly unseen) would be ranked if the (non-existent) centralized complete database was searched. Finally, this estimated ranked list can be used to identify which sources can contribute most of the top-ranked documents.

In sum, the centralized sample database serves as an important surrogate to simulate the behavior of the (unseen) centralized complete database. It enables a set of new techniques for different federated search applications, which are shown in more detail in the rest of this dissertation.

3.3 New information source size estimation methods

The basic Capture-Recapture algorithm was shown to obtain accurate information source size estimates in previous research. However, it is based on an important assumption that a long ranked list (i.e., 1,000 document ids per query) can be acquired by one interaction with each information source. This is often not true in real world applications. When only short ranked lists are available by a single interaction, the basic Capture-Recapture algorithm has excessive communication costs to obtain a long ranked list by sending multiple requests.

Our goal is to develop an effective and much more efficient source size estimation method. In this section, several variants of the Capture-Recapture algorithm are developed to utilize different accessing methods of document ranked lists when the methods are supported. Furthermore, a new source size estimation algorithm based on a different estimation strategy is introduced.

3.3.1 Variants of the Capture-Recapture method

The effectiveness of the Capture-Recapture algorithm can be strongly influenced by what types of methods are provided by the information sources for accessing their document ranked lists. Generally, long document ranked lists cannot be obtained by a single interaction with information sources in real world applications. In contrast, much shorter ranked lists (e.g., 10 or 20 documents) are often available. Assuming 20 document ids can be returned in a single result page with one interaction (information sources may return more or fewer document ids; 20 is used for simplicity), the Capture-Recapture algorithm is allowed to choose document ids from a pool of 20 document ids acquired by a single interaction with an information source. If the ranked lists must be obtained sequentially, the Capture-Recapture algorithm can only access the top 20 document ids for each query with one interaction. On the other hand, some information sources provide the service of directly accessing the ranked list at any specified section. As the documents that appear at the top of a ranked list may have a bias to be ranked highly for many queries, selecting the documents in a wide range of ranked lists makes it possible for the Capture-Recapture algorithm to acquire more random document ids. These two variants of the Capture-Recapture algorithms are called the “Top” approach and the “Direct” approach respectively.

The basic Capture-Recapture algorithm [Liu et al., 2001] only randomly chooses a single document id from a returned result page, while it is possible to utilize all the 20 document ids in the result page and the corresponding variant of the Capture-Recapture algorithm is denoted as the “All” approach in this dissertation.

3.3.2 Sample-Resample method

The Sample-Resample method [Si & Callan, 2003a] uses a different strategy to estimate information source sizes than the Capture-Recapture algorithm. There are several assumptions made by this new information source size estimation algorithm. First, it assumes that the resource representations are created by the query-based sampling algorithm. Then the document frequency information of terms within the sampled documents can be obtained from resource representation. The second assumption of the Sample-Resample method is that each information source provides the information of how many documents in this information source match a single word query (i.e., the document frequency of a term in the complete information source). This type of statistic is often returned together with the retrieved document ranked lists by information sources even in uncooperative environments. For example, both Google and MSN indicate the approximate number of documents matching a single word query.

The new method acquires the information source size estimates with a sample and resample process. The basic procedure of this algorithm can be described as:

- The query-based sampling method is used to build the resource description and the centralized sample database (i.e., the sample process); and
- A single term is randomly selected from the resource description of a specific information source and the term is sent to search the information source as a single word query (i.e., the resample process).

Assume N_{samp} documents from this information source have been sampled and collected in the centralized sample database. Let df_{q_samp} be the number of sampled documents from this source that contain the query term. N denotes the (unknown) information source size and df_q denotes the actual document frequency of the query word in the complete information source. Let A denote the event that a sampled document from the source contains this query term and B denote the event that an arbitrary document in the complete information source is observed to contain this term in the resample step. The probabilities of these two events can be calculated as follows:

$$P(A)=\frac{df_{q\_samp}}{N_{samp}} \qquad P(B)=\frac{df_q}{N} \tag{3.5}$$

If the sampled documents acquired by query-based sampling can be assumed to be a good representation of the complete information source, the above two events should have equal probabilities or formally as P (A) = P (B). Therefore, the size of this information source can be estimated as:

$$\widehat{N}=\frac{df_q * N_{samp}}{df_{q\_samp}} \tag{3.6}$$

In order to reduce the estimation variance, multiple resample queries can be utilized in the Sample-Resample algorithm and the final estimate can be calculated by averaging the individual information source size estimates. For example, if altogether K resample queries are submitted, the final information source size estimate is calculated as follows:

$$\widehat{N}=\frac{1}{K}\sum_{k=1}^{K}\frac{df_{q_k} * N_{samp}}{df_{q_k\_samp}} \tag{3.7}$$

It can be imaged that the variance of the size estimates drops with more and more resample queries. This is studied more carefully with the experiments in Section 3.4.2.

3.4 Evaluation methodology and experimental results

Very little prior research has addressed the evaluation methodology of information source size estimation algorithms. This section first discusses how to define evaluation metrics for fair comparison of source size estimation algorithms by considering their costs. Furthermore, a set of experiments is conducted to study the effectiveness and efficiency of both the Capture-Recapture algorithm and the Sample-Resample algorithm.

3.4.1 Evaluation methodology

Information source size estimation algorithms are associated with different types of costs. To provide accurate evaluations of these algorithms, the accuracies of the algorithms should be compared when they are associated with the same amount of costs. In this work, the costs of different information source size estimation algorithms are measured by the number of interactions that they make with a particular hidden information source. As both the action of acquiring a page of ranked document ids and the action of downloading a document need a single interaction with an information source, these two actions are associated with the same cost.

Both the Capture-Recapture algorithm and the Sample-Resample algorithm submit queries to a hidden information source to collect some information. The Capture-Recapture algorithm sends out a query and extracts document ids from the returned document ranked lists. If it is assumed that 20 document ids can be returned in a single result page, the Capture-Recapture algorithm is allowed to choose the document ids from a pool of 20 document ids by a single interaction with a particular information source. The Sample-Resample algorithm requires submitting several queries to the hidden information source in the resample step to collect document frequencies in the complete information source.

There are two evaluation scenarios to compare the Capture-Recapture algorithm and the Sample-Resample algorithm. In the first scenario, information source size estimation is combined with other components of a federated search system. Both the Capture-Recapture algorithm and the Sample-Resample algorithm can take advantage of the information acquired by the query-based sampling method. Therefore, the Capture-Recapture algorithm can use the document ids in the result pages of sampled queries and the Sample-Resample algorithm can access the downloaded documents to calculate the sampled document frequency statistics. More specifically, let us assume that the query-based sampling method obtains the resource description for each information source by submitting 80 queries and downloading 300 documents. The Sample-Resample algorithm can take advantage of the 300 downloaded documents (i.e., 80 queries are

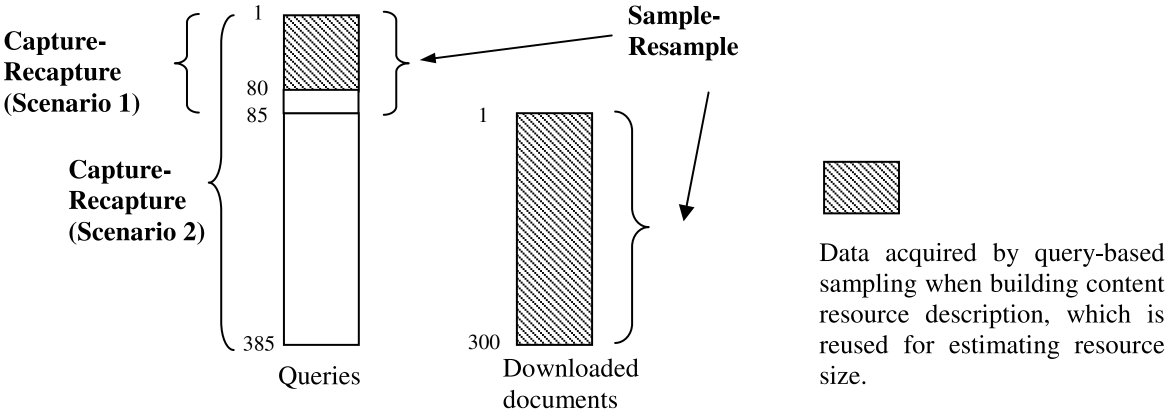
Capture-
Recapture
(Scenario 1)
1
80
85
385
Capture-
Recapture
(Scenario 2)
Queries
Sample-
Resample
1
300
Downloaded
documents
Data acquired by query-based sampling when building content resource description, which is reused for estimating resource size.

Figure 3.2: The data utilized by the Capture-Recapture algorithm and the Sample-Resample algorithm. (the shadow part of the data may be obtained from query-based sampling.)

implicitly used for acquiring these documents) while the Capture-Recapture algorithm can only utilize the 80 pages of ranked document ids from the sample queries. The Sample-Resample method needs to send several extra queries in the resample process. The number of the resample queries is set to 5 in the experiments of this chapter. Therefore, the Capture-Recapture algorithm of this evaluation scenario is only allowed to utilize 85 pages of ranked document ids as shown in Figure 3.2. Note that it is possible to utilize some of the sampling queries from query-based sampling as the resample queries. However, this approach may produce overestimated document frequencies on the sampled documents and thus is not utilized in this work.

In the second scenario, information source size estimation is independent from other federated search components, and the information from query-based sampling cannot be accessed for free and the costs such as downloading documents must be included in the evaluation. Therefore, the cost of the Sample-Resample algorithm is increased and thus the Capture-Recapture algorithm within scenario 2 is allowed to obtain more pages of ranked document ids as shown in Figure 3.2. There is a bias towards the Capture-Recapture algorithm for the second evaluation scenario where it is allowed to send 385 queries to obtain pages of ranked document ids instead of only 85 queries in the first scenario.

Scenario 1 may be a better representation of operational environments. However, in order to conduct a component-level study and to evaluate the effectiveness of the new proposed Sample-Resample algorithm in a stricter manner, the second evaluation scenario is chosen in this work. This choice does not really affect the Sample-Resample method. It gives Capture-Recapture a best-case scenario for the comparison. If the Capture-Recapture cannot win here, it definitely does not win with scenario 1.

In summary, the Capture-Recapture method is allowed to send 385 queries to a hidden information source, where the document ids acquired by the first half of the queries are used as the first sample and the document ids acquired by the second half of the queries are collected in the second sample. Different variants of the Capture-Recapture algorithm utilize different methods of accessing ranked lists. By the same amount of interactions with the information source, the Sample-Resample method uses 80 queries to download 300 documents from the information source in the sample process and submits 5 additional queries in the resample process.

The absolute error ratio (AER) measure is used to evaluate the accuracies of information source size estimation algorithms, which is consistent with prior work [Liu et al., 2001]. Formally, let N_{dbi} denote the actual information source size of the i^{th} information source and $\widehat{N}_{db_i}$ denote the corresponding estimate. Then, the AER is calculated as follows:

$$AER=\frac{\left|\widehat{N}_{db_i}\text{-}N_{db_i}\right|}{N_{db_i}} \qquad (3.8)$$

The mean absolute error ratio (MAER), which is the average value of multiple AER values, is used for evaluation when a set of information source size estimates is evaluated.

The first set of experiments was conducted to compare the effectiveness of the Capture-Recapture algorithm and the Sample-Resample algorithm. Four testbeds were used, namely the Trec123_100Col testbed, the Trec123_10Col testbed, the WT10g_10% testbed and the GOV_10% testbed (more detailed information can be found in Chapter 2). These four testbeds contain information sources of small sizes (i.e., about 10,000 documents on average) and moderately large information sources (i.e., about 100,000 documents on average) respectively. Each information source within the Trec123_100Col and Trec123_10Col testbeds is composed of documents with homogeneous document statistics (i.e., on Trec123_100Col) or moderately heterogeneous document statistics (i.e., on Trec123_10Col). In contrast, the WT10g_10% and GOV_10% testbeds are created from Web data and they are composed of documents with more heterogeneous statistics.

The second set of experiments was conducted to evaluate the Sample-Resample algorithm with larger testbeds such as the Trec123_2Col testbed (i.e., about 500,000 documents per source) and the WT10g_100% and GOV_100% testbeds (i.e., about 1,200,000 documents per source). The information sources in the second set of experiments contain documents with heterogeneous statistics. A variant of the Sample-Resample algorithm, which utilizes multiple word queries in the sample process, is proposed to further improve the accuracy of information source size estimation. The behavior of the algorithm is studied

Table 3.1: Experiment results of the information source size estimation methods on 4 small or moderately large testbeds with mean absolute error ratio (MAER) metric.

Testbed	**Size Per Source (Average)**	**Mean Absolute Error (MAER)**		
		Capture-Recapture TopAll	**Capture-Recapture DirectAll**	**Sample-Resample**
Trec123_100Col	10,782	0.377	**0.182**	0.232
Trec123_10Col	107,820	0.849	0.404	**0.299**
WT10g_10%	121,537	0.915	0.697	**0.556**
GOV_10%	124,775	0.893	0.603	**0.483**

with an extensive set of experiments by varying the information source sizes and the number of resample queries.

When not stated explicitly, the error rates of different algorithms are obtained by averaging the results from 5 different trials.

3.4.2 Experiment results

The experiment results of all the variants of the Capture-Recapture algorithm and the Sample-Resample algorithm are shown in Table 3.1. It can be seen that the extension proposed in Section 3.3.1 for the Capture-Recapture algorithm by using the method of randomly accessing document ranked lists (i.e., "Direct") has substantially improved the accuracy of the Capture-Recapture algorithm. The improvement can be attributed to the reason that the method of randomly accessing the document ranked lists makes the two document id samples more independent.

The Sample-Resample algorithm was more accurate than both the variants of the Capture-Recapture algorithm except for the DirectAll method on the Trec123_100Col testbed. Its advantage over the Capture-Recapture algorithm was larger on the Trec123_10Col, WT10g_10% and GOV_10% testbeds, which have larger information sources. The Capture-Recapture algorithm was only comparable with the Sample-Resample algorithm for small information sources with the support of random access of the ranked lists, which is a very strong assumption and may not be provided by many information sources. Therefore, we tend to draw the conclusion that the Sample-Resample algorithm is more robust than the Capture-Recapture algorithm especially for relatively large information sources. We do not argue that the Sample-Resample algorithm is better than the Capture-Recapture algorithm in all cases. In the environments where the information sources do not provide document frequency information, where the information sources can provide a large amount of document ids (i.e., larger than 20) within a result page in their ranked lists, or where the random access of ranked list can be guaranteed, some variants of Capture-Recapture algorithm may have their advantages.

Careful analysis reveals that both the Capture-Recapture algorithm and the Sample-Resample algorithm tend to underestimate the information source sizes. This can be explained by the assumptions made by these two approaches. The Capture-Recapture algorithm assumes that the two samples of document ids are independent. This is not perfect as some documents that have more words and more diverse contents are more likely to be retrieved for different queries and thus are more likely to appear in both the first sample and the second sample. Also, the early sampled documents may be more likely to be retrieved again. On the other hand, the Sample-Resample algorithm assumes that the set of sampled documents is a good representation of the complete information source. However, the complete information source contains many documents that seldom appear at the top of ranked lists for one-term queries within query-based sampling. These documents are composed of a large proportion of unseen words. The proportion of documents that contain a particular resample query is usually overestimated based on the sampled documents. Thus, a larger denominator in Equation 3.6 produces an underestimated source size estimate.

The first set of experiments was conducted to show the effectiveness of the Sample-Resample algorithm for small or moderately large information sources. For information sources that contain more documents with diverse document statistics, it is more crucial to obtain unbiased sample documents in query-based sampling for accurate source size estimates. In order to investigate the effect of biased sampled documents on information source size estimation, a second set of experiments was conducted to study the behavior of the Sample-Resample algorithm on the Trec123_2Col testbed, the WT10g_100% testbed and the GOV_100% testbed, which contain information sources that have about 500,000 to 1,200,000 documents. Particularly, two variants of the Sample-Resample algorithm were designed to utilize sampled documents acquired by different sampling methods.

One variant of the Sample-Resample estimation algorithm acquires sampled documents by perfectly random sampling during query-based sampling (called Sample-Resample method with random sampling). Although the perfectly random sampling approach is not supported by most information sources in real world applications, the Sample-Resample method with random sampling serves as an optimal baseline algorithm to compare with other variants of the Sample-Resample method with biased sampling approaches.

The original Sample-Resample method acquires sampled documents by sending one-word queries. A natural extension of this sampling approach is to use multi-word sampling queries during query-based sampling,

Table 3.2: Experiment results of the information source size estimation methods on 3 large testbeds using the mean absolute error ratio (MAER) metric.

Testbed	Size Per Source (Average)	Mean Absolute Error		
		Sample-Resample	Sample-Resample (Multi-Word Sampling)	Sample-Resample (Random Sampling)
Trec123_2Col	539,100	0.492	0.170	0.046
WT10g_100%	1,215,370	0.543	0.181	0.037
GOV_100%	1,247,753	0.589	0.090	0.049

which has been studied for building resource content descriptions [Craswell, 2000]. Specifically, a set of words (i.e., 10 in this work) are randomly generated in an independent way and they are combined together as a multi-word query. Multi-word queries are associated with more diverse topics than single-word queries and thus increase the possibilities of producing a more random set of sampled documents. This sampling approach promises to generate sampled documents with less sampling bias. It is called the Sample-Resample method with multi-word sampling in this work. Note that the multi-word sampling approach may not work for information sources with exact match retrieval algorithms. For example, a search engine with a Boolean retrieval algorithm may always treat a multi-word query with the Boolean operator "AND" and may return no documents if the query words never occur together in the documents of the information source. (However, the method works with Boolean "OR" operator.) This is not a problem for the experiments in this dissertation, which utilize best match retrieval algorithms as described in Chapter 2. Similar to the experimental setting of the first set of experiments, different variants of the Sample-Resample method were allowed to sample 300 documents and send 5 resample queries in the second set of experiments.

The experiment results of the three variants of Sample-Resample method, namely the original Sample-Resample method, the Sample-Resample method with multi-word sampling, and the Sample-Resample method with random sampling are shown in Table 3.2. It can be seen that the accuracy of the original Sample-Resample method was worse on the Trec123_2Col testbed than on the Trec123_100Col and Trec123_10Col testbeds. Analysis indicates that as there are a much larger number of documents with heterogeneous characteristics in each of the two very large information sources on the Trec123_2Col testbed, so the top ranked documents in the ranked lists of sampling queries are associated more sampling bias. For example, the documents from the Department of Energy collections are substantially shorter (i.e., about 120 terms on average) than documents from other collections (e.g., on average, documents for Associated Press contain about 460 terms). DOE documents seldom appear at the top of the ranked lists of sampling queries. This sampling bias causes the original Sample-Resample method to ignore the existence of many rarely seen

Table 3.3: Experiment results of the Sample-Resample method with Multi-Word Sampling approach on testbeds of varying the source sizes. The metric is mean absolute error ratio (MAER).

Testbed	**WT10g_10%**	**WT10g_30%**	**WT10g_50%**	**WT10g_70%**	**WT10g_90%**	**WT10g_100%**
Number of Documents	121,537	364,611	607,605	850,759	1,093833	1,215,370
MAER	0.094	0.105	0.153	0.158	0.186	0.181

Testbed	**GOV_10%**	**GOV_30%**	**GOV_50%**	**GOV_70%**	**GOV_90%**	**GOV_100%**
Number of Documents	124,775	374,326	623,877	873,427	1,122978	1,247,753
MAER	0.068	0.072	0.089	0.088	0.079	0.090

documents. On the other hand, the accuracies on the WT10g_100% and GOV_100% testbed did not change much from those on the WT10g_10% and GOV_10% testbed (Table 3.1) as all of them contain documents with heterogeneous statistics.

It can be seen from Table 3.2 that the Sample-Resample method with the multi-word sampling approach generated reasonably good estimates on all the three testbeds. This suggests that the multi-word sampling approach (when it is supported by information sources) does help to reduce the sampling bias caused by one-word sampling approach.

The last column of Table 3.2 shows the accuracy of the Sample-Resample method with perfectly random sampling. It can be seen that the method can acquire very accurate source size estimate. This exactly points out that the key factor to improve the estimation accuracy of Sample-Resample method is to generate representative sample documents with small sampling bias.

The behavior of the Sample-Resample method with multi-word sampling approach is studied by further varying the information source sizes and the number of resample queries. Particularly, a range of information sources were created by selecting 10%-100% documents from WT10g and GOV testbeds. These testbeds contain about 120,000 documents to about 1,200,000 documents. The Sample-Resample method with the multi-word sampling approach was applied on these testbeds and the results are shown in Table 3.3. It can be seen from the results that the mean absolute error rates are always under 20% for all configurations. The absolute error increases as the source sizes grow. However, the growth in error rate is slower when sizes grow from 50% to 100% than from 10% to 50% on both the WT10g and GOV testbeds.

Table 3.4: Experiment results of the Sample-Resample method with Multi-Word Sampling approach on the Trec123_Col2 testbed by varying number of resample queries (1-100). The metric is mean absolute error ratio (MAER). The standard deviation (STD) of MAER is also provided.

Number of Resample Queries	**1**	**5**	**10**	**20**	**50**	**100**
MAER	0.186	0.166	0.142	0.150	0.135	0.153
STD	0.134	0.099	0.117	0.0967	0.085	0.079

Table 3.5: Experiment results of the Sample-Resample method with Multi-Word Sampling approach on the WT10g_100% testbed by varying number of resample queries (1-100). The metric is mean absolute error ratio (MAER). The standard deviation (STD) of MAER is also provided.

Number of Resample Queries	**1**	**5**	**10**	**20**	**50**	**100**
MAER	0.179	0.150	0.178	0.201	0.165	0.172
STD	0.141	0.111	0.110	0.112	0.087	0.091

Table 3.5: Experiment results of the Sample-Resample method with Multi-Word Sampling approach on the GOV_100% testbed by varying number of resample queries (1-100). The metric is mean absolute error ratio (MAER). The standard deviation (STD) of MAER is also provided.

Number of Resample Queries	**1**	**5**	**10**	**20**	**50**	**100**
MAER	0.119	0.096	0.107	0.090	0.077	0.079
STD	0.103	0.066	0.097	0.083	0.088	0.065

Another set of experiments was conducted by varying the number of resample queries used by the Sample-Resample method. The previous empirical studies of the Sample-Resample method in this chapter were conducted by sending 5 resample queries to available sources. It is helpful to explore the impact of the number of resample queries. Specifically, the Sample-Resample method with multi-word sampling approach was applied on the three testbeds TREC123_2Col, WT10g_100%, and GOV_100% by using 1, 5, 10, 20, 50 and 100 resample queries. The results are shown in Tables 3.4 and 3.5 respectively. These results were obtained by averaging 15 trials. It can be seen from the results that the mean absolute error rates of using a few resample queries (i.e., 1 or 5) are similar to those of using many more resample queries (i.e., 50 or 100). However, the standard deviation of the error rates drops substantially as more resample queries are used. Especially, the decrease of the standard deviation is apparent by using five resample queries than by using only one resample query. The decrease of the standard deviation is much slower when more resample queries are used (i.e., more than 10). This supports our previous approach of using 5 resample queries.

3.5 Summary

Previous research has provided good solutions for discovering word histograms from available sources. Recently, the field starts to recognize the importance of information source sizes as another type of resource representation for supporting more accurate resource selection decision, which demands an effective and efficient source size estimation algorithm.

The prior Capture-Recapture method is shown in this chapter to be inefficient for acquiring accurate source size estimates. On the other hand, a new Sample-Resample method is proposed. This method works with query-based sampling method and views the sampled documents as a small set of representative documents from complete information sources. It analyzes document frequency statistics from both the sampled documents and the complete information sources to estimate information source sizes. The Sample-Resample method is shown to generate more accurate size estimates than the Capture-Recapture method. Furthermore, different variants of the Sample-Resample method have been proposed. When multi-word sampling is supported by available information sources, the Sample-Resample method can acquire source size error rates as low as 10%-20% on a range of testbeds with several hundred thousand documents to about 1 million documents.

There are several directions to investigate the behavior of the Sample-Resample method in the future. For example, the accuracy of the Sample-Resample method has been evaluated with information sources that contain up to about 1 million documents, which is an upper bound for the federated search environments evaluated in this dissertation. However, it is helpful to test the Sample-Resample method with even larger sources for real world federated search applications. Another possibility is to propose a new variant of the Sample-Resample method without the requirement of obtaining the document frequency information from complete information sources. This ability is important as information sources in real world applications may not provide this functionality or may provide statistics with errors.

Another interesting issue is that the experiments in this chapter provide more information about bias in query-based sampling, which was also observed in previous research [Craswell, 2000]. The experiments demonstrate the effect of the bias on source size estimation algorithms. The multi-word sampling approach provides a method of addressing the sampling bias, but there may be better solutions, which remains an interesting future research topic.

This chapter also points out the importance of a centralized sample database as a surrogate for understanding and simulating the behavior of centralized complete database, which does not exist in federated search environments. The centralized sample database is very important for resource selection and results merging as described in Chapters 4-8.

For most experiments in this dissertation, a fixed number of documents (e.g., 300) are sampled from each information source. Prior research has indicated that this is adequate in many federated search environments. However, an adaptive query-based sampling approach combined with the Sample-Resample size estimation method might improve the accuracy of resource selection especially for large information sources. For example, an adaptive sampling approach could sample a small percentage (e.g., 1%) documents from each available source. It is an open question whether adaptive sampling would further improve the accuracy of the resource selection and results merging methods described in Chapters 4-8.

Furthermore, the research in this dissertation builds the centralized sample database only once without update. However, in real world applications, the contents of hidden information sources may update or change frequently. Previous research in [Ipeirotis, Ntoulas & Gravano, 2005] provided a good method to model the content changes. One future research topic is to study the strategy of updating the centralized sample database for better performance of federated search applications such as resource selection and results merging.

Chapter 4: Resource Selection

After resource descriptions are acquired, the next task of an information source recommendation system or a federated document retrieval system is to select a small set of information sources to search. This chapter first introduces previous research on resource selection and particularly discusses the deficiency of the "big document" resource selection approach. To address the problem, several extensions are proposed to incorporate information source size estimates with two well-known resource selection algorithms: CORI and KL divergence. Furthermore, a Relevant Document Distribution Estimation (ReDDE) resource selection algorithm is introduced, which explicitly optimizes the high-recall goal of information source recommendation application. Experiment results are shown to evaluate the accuracies of these resource selection algorithms.

4.1 Previous research on resource selection

There is a large body of prior research on resource selection, which includes bGlOSS/gGlOSS/vGlOSS [Gravano et al., 1994; Gravano et al., 1999], query clustering/RDD [Voorhees et al., 1995], decision-theoretic framework (DTF) [Fuhr, 1999; Nottelmann & Fuhr, 2003b], lightweight probes [Hawking & Thistlewaite, 1999], CVV [Yuwono & Lee, 1997], CORI [Callan, Lu & Croft, 1995; Callan, 2000], KL-divergence algorithm [Xu & Croft, 1999], and a hierarchical database sampling and selection algorithm [Ipeirotis & Gravano, 2002; Ipeirotis & Gravano, 2004].

The algorithms of query clustering/RDD, DTF and lightweight probes utilize different types of training data. The query clustering/RDD methods and the DTF method require human relevance judgments while the lightweight probes method obtains necessary statistics in an unsupervised manner. In contrast, most other methods do not require training data and only utilize the information obtained from resource descriptions. The CVV, CORI and KL-divergence algorithms follow the strategy of the "big document" approach by treating information sources as big documents and ranking information sources by their similarity scores with user queries. On the other hand, the bGlOSS/gGlOSS/vGlOSS algorithms turn away from the "big document" approach by considering goodness/utilities of individual documents. Finally, the hierarchical database sampling and selection algorithm builds information source hierarchy and utilizes other base resource selection algorithms (e.g., CORI or KL-divergence) to rank the information sources. Below we provide more detailed information about these algorithms.

The query clustering/RDD [Voorhees et al., 1995] resource selection algorithms rely on a query log that is composed of a set of training queries and relevance judgments. These algorithms use methods like the k nearest neighbor algorithm [Yang, 1999; Duda, Hart & Stork, 2000] to detect similar training queries to a particular user query, and then rank the information sources by the distribution of the relevant documents for these training queries. They may work well when there are enough similar training queries with complete relevance judgment data. However, the main problems with these methods are: i) the required human effort of generating relevance judgment data grows linearly with the number of information sources; and ii) when individual information sources update their contents, it is necessary to generate new training data.

The DTF method [Fuhr, 1999; Nottelmann & Fuhr, 2003b] yields a source selection decision that minimizes a function of overall costs (e.g., retrieval accuracy, query processing cost and communication cost) for the federated document retrieval application. The DTF method is based on a solid decision framework. However, it assumes that all information sources use the same type of retrieval algorithm, which is usually not true in uncooperative environments. The DTF method requires training data of human relevance judgments for the results retrieved from each available information source, which is an excessive amount of human effort if there are many information sources. More detailed analysis of this algorithm can be found in Chapter 7.

The lightweight probes method [Hawking & Thistlewaite, 1999] broadcasts two-word subsets of user queries to all available information sources to obtain query term statistics. These term statistics are used to rank the information sources. This method requires very little amount of prior knowledge about each information source and calculates the information source ranking in an online manner. Therefore, it is better in recognizing content change of information sources. However, it is often associated significant communication costs for sending the query probes, which is more serious in a large federated search system with many information sources. Furthermore, the lightweight probes method assumes a common representation and some cooperation from available sources, which may not be available in uncooperative environments.

Many resource selection algorithms share the property of treating information sources as big documents and calculating similarities between these "big documents" and user queries to make the selection decision. These big document resource selection algorithms include the CVV [Yuwono & Lee, 1997] algorithm, the

CORI [Callan, Lu & Croft, 1995] algorithm and the Kullback-Leibler (KL) divergence algorithm [Xu & Croft, 1999] etc. They choose different representations of the “big documents” and calculate different types of similarity scores. However, these methods do not explicitly consider whether individual documents within an information source are relevant or similar to the query. This causes trouble for optimizing the high-recall goal of information source recommendation application, where the actual goodness of each information source is measured by the amount of relevant documents it contains. For example, in “big document” methods the source selection scores are based on the number of matching words in the information sources. These methods cannot distinguish whether there are many matches in a single long document or few matches in each of many short documents, because the boundaries among documents are not preserved.

The Cue Validity Variance (CVV) resource selection algorithm [Yuwono & Lee, 1997] assigns different weights (CVV) to the words. The words that better discriminate information sources are assigned higher weights than the words that distribute more evenly across the information sources. With these term weights, the CVV algorithm ranks available information sources by the sum of the weighted document frequencies of query words. However, as large information sources often have large document frequency values for query words, it has been indicated in previous research [Craswell, 2000] that the CVV resource selection algorithm tends to favor large information sources and thus information source size normalization needs to be introduced for improving the selection accuracy.

Another two “big document” resource selection methods are CORI and KL-divergence algorithms. More detailed information about these two algorithms is provided here as they are used as the baseline algorithms in this chapter.

The CORI resource selection algorithm [Callan, Lu & Croft, 1995; Callan, 2000] utilizes a Bayesian inference network model with an adapted Okapi term frequency normalization formula [Robertson & Walker, 1994] to rank available information sources. CORI is related to the INQUERY ad-hoc retrieval algorithm. Formally, the belief of the i^th information source associated with the word q, $p(q|db_i)$, is calculated as:

$$T=\frac{df}{df+50+150*cw_i/avg\_cw} \tag{4.1}$$

$$I=\frac{\log\left(\frac{|DB|+0.5}{cf}\right)}{\log(|DB|+1.0)} \tag{4.2}$$

$$p(q|db_i)=b+(1-b)*T*I \tag{4.3}$$

where: df is the number of documents in the i^{th} information source that contain q;

cf is the number of information sources that contain q;

|DB| is the number of information sources to be ranked;

cw_i is the number of words in the i^{th} information source;

avg_cw is the average cw of the information sources to be ranked; and

b is the default belief, usually set to 0.4.

The CORI algorithm ranks information sources by the belief P (Q | db_i), which denotes the probability that query Q is satisfied with the observation of the i^{th} information source. The most common way to calculate the belief P (Q | db_i) is to use the average value of the beliefs of all query words; a set of more complex query operators are also available for handling structured queries [Callan, 2000].

The Kullback-Leibler (KL) divergence resource selection algorithm was proposed by Xu and Croft [Xu & Croft, 1999]. In this method, the content descriptions of all information sources are treated as single big documents and are modeled using multinomial distributions. Similarly, user queries are also modeled as multinomial distributions. The Kullback-Leibler divergence between the distributions of a user query and available information sources is used to rank the information sources. Formally, the KL divergence between the query Q and the i^{th} information source is computed as:

$$KL(Q,db_i)=\sum_{q\in Q}P(q|Q)\log\left\{\frac{P(q|Q)}{\lambda P(q|db_i)+(1-\lambda)P(q|G)}\right\} \tag{4.4}$$

P (q | db_i) is the probability of query term q in the unigram language model (multinomial distribution) of the content description of the i^{th} information source. P (q | Q) and P (q | G) are the probabilities of the query term q in the query language model and a global language model respectively, which are calculated based on the relatively frequencies of term q in user query and the global corpus. The global language model can be obtained by combining resource descriptions of all information sources together and building a single language model. Linear interpolation constant λ smoothes the information source language model with the global language model, which is set to a numerical value between 0 and 1 (e.g., 0.5).

The CORI and KL-divergence resource selection algorithms have been shown in previous studies to be more robust and effective than several alternatives in different experiment environments [French et al., 1999;

Craswell et al., 2000; Xu & Croft, 1999]. They are computationally efficient and can be easily applied in uncooperative environments with query-based sampling method. However, they belong to the big document resource selection approach and do not normalize the sizes of hidden information sources well. Section 4.4 shows experiments that both the CORI and KL-divergence algorithms have a strong bias against large information sources and thus miss a large amount of relevant documents.

Gravano and García-Molina proposed the bGlOSS resource selection algorithm [Gravano et al., 1994] and the gGlOSS/vGlOSS algorithms in [Gravano & García-Molina, 1995; Gravano et al., 1999]. These methods turn away from "big document" approach by considering the goodness of individual documents. This is similar to our new approach and thus these methods are discussed in more detail below.

The bGlOSS algorithm is based on a Boolean retrieval algorithm. It assumes that the distributions of different query terms are independent and estimates the number of documents containing query terms to rank the information sources. The vGlOSS algorithm is a more sophisticated model based on the vector space model. It represents a document in the vector space of m distinct words as $<w_{1,...,}w_m>$ where w_k is the weight (e.g., the idf value) assigned to the k^{th} word in the document. Similarly, a user query is represented in the same space as $<q_{1,...,}q_m>$ where q_k is typically a function of the number of occurrences that the k^{th} word appears in the query. Using these representations, the vGlOSS method calculates the similarity value sim(Q , d) between a query Q and a document d as follows:

$$\text{sim(Q,d)}=\sum_{k=1}^{m} q_k * w_k \tag{4.5}$$

which is a dot product between two vectors of words in documents and words in the query. The vGlOSS method calculates the *goodness* of the information source with respect to the query as:

$$\text{Goodness}(l,Q,db_i)=\sum_{\{d\in db_i\, ,\, sim(Q,d)>l\}} \text{sim(Q,d)} \tag{4.6}$$

where l is a threshold. The process is repeated for all information sources and finally they are ranked by the corresponding goodness scores. When the contents of all the documents in the hidden information sources are accessible, the goodness values represented in Equation 4.6 can be exactly calculated. Therefore, the values as Ideal(l) of available information sources can be obtained. But in most federated search environments, the resource selection algorithms can only observe limited information about the documents in the information sources. In this case, the vGlOSS algorithm tries to approximate the Ideal(l) function by two functions Max(l) and Sum(l) [Gravano & García-Molina, 1995; Gravano et al., 1999]. These two functions calculate the estimated goodness for an information source based on a high correlation scenario and a disjoint scenario of the query word co-occurrences. vGlOSS needs two vectors containing information about the document frequency and the sum of weights of each word from a particular information source to calculate Max(l) and Sum(l). Information sources can provide these statistics in cooperative environments. Otherwise, sampling queries can be sent to learn the information in uncooperative environments.

The vGlOSS method considers individual documents to judge whether they are relevant or not and thus turns away from the "big document" approach. This provides a better opportunity to model the utility of each information source as the amount of relevant documents that it contains. However, two important issues limit its power. First, the two approximations as Max(l) and Sum(l) make too strong assumptions about query word distributions within information sources. Max(l) assumes query words always occur together in the documents while Sum(l) assumes that query words do not occur together. Second, the word weight w_k in Equation 4.5 is source-specific and thus the same document may be judged as relevant in one information source and as irrelevant in another information source due to different corpus statistics. As the utility of a particular information source is measured by the number of relevant documents it contains, the information source-specific weighting scheme may not be appropriate. These two issues can be used to explain why the vGlOSS algorithm is less accurate than algorithms such as CORI or KL-divergence in several empirically studies [French et al., 1999; Craswell, 2000].

The hierarchical database sampling and selection algorithm [Ipeirotis & Gravano, 2002] derives information source descriptions by using focused query probes (i.e., query on specific topics) and builds hierarchical structure for hidden information sources. A more recent shrinkage-based selection algorithm [Ipeirotis & Gravano, 2004] utilizes a similar strategy with refined resource representation. These methods provide a better way to smooth the word distributions in the resource representations of information sources. However, as the resource selection in the above two algorithms is still conducted with base resource selection algorithms such as CORI and bGlOSS, these resource selection algorithms still suffer from the weakness of the base algorithms.

4.2 Incorporating size estimates into resource selection algorithms

The goal of the resource selection algorithms in an information source recommendation system is to select a small number of information sources with the largest number of relevant documents. Therefore, to estimate the number of relevant documents contained in available sources, the sizes of available information sources play an important role in designing effective resource selection algorithms. However, very little research has been conducted to study the effect of information source sizes on resource selection. One reason is the difficulty to acquire information source size estimates effectively and efficiently in uncooperative environments. Chapter 3 presents the Sample-Resample method as a promising solution, which provides an opportunity to better adjust or normalize resource selection algorithms with respect to the information source sizes.

The information source size scale factor is associated with each information source and defined as the ratio of its estimated size and the number of sampled documents from this source as follows:

$$SF_{db_i} = \frac{\hat{N}_{db_i}}{N_{db_i\_samp}} \tag{4.7}$$

where $\hat{N}_{db_i}$ denotes the source size estimate for a particular ith information source and N_{dbi_samp} denotes the number of sampled documents from this information source.

Previous research has shown that the CORI and KL-divergence algorithms are more robust and effective than several other alternatives in different federated search environments [French et al., 1999; Craswell et al., 2000; Xu & Croft, 1999]. They were chosen as the baseline algorithms in this work and new variants of these two algorithms that adjust for information source sizes are proposed in this section.

The information source selection scores for the CORI selection algorithm are calculated as Equations 4.1, 4.2 and 4.3. Callan pointed out that the CORI formula of Equation 4.1 is a variation of the Roberson’s term frequency (tf) [Robertson & Walker, 1994] weight, in which the term frequency is replaced by document frequencies in the sampled documents and the constants are scaled by a factor of 100 to accommodate the large document frequency values [Callan, Lu & Croft, 1995; Callan, 2000]. Equation 4.1 can be generalized and reformulated as follows:

$$T = \frac{df}{df + df\_base + df\_factor * cw_i / avg\_cw} \tag{4.8}$$

where df is the document frequency, df_base and df_factor are two constants, cw_i and avg_cw represent the number of words in the ith information source and the average number of words across available information sources respectively. In uncooperative environments, CORI is combined with the query-based sampling method and the above statistics are calculated on the sampled data. df is the document frequency in the sampled documents; cw_i and avg_cw are calculated based on the sampled documents and the df_base and df_factor constants are set to 50 and 150 by default. This configuration has been shown to be effective on several testbeds with rather uniform source size distributions like Trec123_100Col [Callan, 2000], where the average information source size is about 10,000 and 300 hundreds documents are sampled from each source.

Our basic idea of incorporating the information source size factor into the CORI resource selection algorithm is to simulate the behavior as if the complete resource descriptions are available (i.e., the statistics in Equations 4.1, 4.2 and 4.3 are calculated from all the documents across available information sources). To accomplish that, at least three issues should be addressed in Equation 4.8 [Si & Callan, 2004a].

First, the document frequency represented by df in Equation 4.8 is the document frequency of a specific term q in the sampled documents. To estimate the actual document frequency in the information source, the sampled document frequency should be scaled as follows:

$$\mathrm{df}' = \mathrm{df} * \mathrm{SF}_{\mathrm{db}_i} \tag{4.9}$$

Note that a similar approach is also used to estimate actual document frequency in [Ipeirotis & Gravano, 2002; Ipeirotis & Gravano, 2004].

Second, the cw_i in Equation 4.8 denotes the number of words contained in the sampled documents from the i^{th} information source, while avg_cw represents the average number of words in sampled documents across all the information sources. To incorporate the information source size factor, these two values should also be scaled as:

$$\mathrm{cw}_i' = \mathrm{cw}_i * \mathrm{SF}_{\mathrm{db}_i} \tag{4.10}$$

$$\mathrm{avg\_cw}' = \frac{1}{|\mathrm{DB}|}\sum_i \mathrm{cw}_i * \mathrm{SF}_{\mathrm{db}_i} \tag{4.11}$$

where |DB| represents the number of information sources.

The last issue to be addressed is the two constants as df_base and df_factor. It was indicated in [Callan, 2000] that large df_base and df_factor values should be used to accommodate large values of the document frequencies. However, how to most effectively set the values of df_base and df_factor is still not clear. The values of 0.5 and 1.5 are chosen for ad-hoc document retrieval in the Okapi formula [Robertson & Walker, 1994]. The values of 50 and 150 have been shown to work for the CORI resource selection algorithm with both complete resource descriptions and sampled resource descriptions (i.e., 300 sampled documents for each information source) [Callan, 2000]. We do not try to solve the optimal settings of df_base and df_factor in this work. However, the effects of larger values of df_base and df_factor are investigated. More specifically, these two values are scaled as follows:

$$\text{df\_base}' = 50 * SF_{db_i} \tag{4.12}$$

$$\text{df\_factor}' = 150 * SF_{db_i} \tag{4.13}$$

Finally, all these updated formulae are plugged into Equation 4.8 to calculate a new T value that considers the information source size factor. Furthermore, the information source beliefs P (Q | db_i) are calculated and the information sources can be ranked accordingly.

Two variants of the CORI algorithm are proposed based on the above extensions. CORI/Ext1 algorithm uses the updated document frequency in Equation 4.9 and the updated number of words in information sources in Equations 4.10 and 4.11. The second extension, which is called CORI/Ext2, takes advantage of all the updated document frequency, the updated number of words and the two new df_base and df_factor constants in Equations 4.10 to 4.13. The difference between the CORI/Ext1 and CORI/Ext2 algorithms is the choice of relatively small values of df_base and df_factor (i.e., CORI/Ext1) and the choice of relatively large values (i.e., CORI/Ext2). The comparison between the CORI/Ext1 and CORI/Ext2 algorithms helps us investigate the effectiveness of these parameter settings.

The KL-divergence resource selection algorithm was proposed by Xu and Croft [Xu & Croft, 1999]. It views the resource content representations of available information sources and user queries as probability distributions and calculates the KL-divergence distance between these probability distributions to rank the information sources. The KL-divergence resource selection algorithm can be extended and given an interpretation in the language-modeling framework [Si & Callan, 2004a]. In this framework, all the sampled documents from a specific information source are collapsed into a single large document and a unigram language model (i.e., multinomial distribution) is calculated for each of the large documents. More specifically, the information sources are sorted by the probabilities of P (db_i | Q), which are the generation probabilities of predicting different information sources based on the observation of query Q. By the Bayesian rule, the probabilities of P (db_i | Q) can be further calculated as follows:

$$P(db_i|Q) = \frac{P(Q|db_i) * P(db_i)}{P(Q)} \tag{4.14}$$

where P (Q | db_i) denotes the likelihood of generating query Q from the i^{th} information source; P (db_i) is the prior probability, which represents our preference of this specific information source before observing the query; and P (Q) is the generation probability of the query and acts as the normalization factor. Since P (Q) is not related with the information source ranking, it is ignored in the calculation and Equation 4.14 can be simplified as:

$$P(db_i|Q) \propto P(Q|db_i) * P(db_i) \quad (4.15)$$

Following the Naïve-Bayes principle of assuming query terms are independent with each other, the value of P (Q | db_i) is calculated as:

$$P(Q|db_i) = \prod_{q \in Q} (\lambda P(q|db_i) + (1-\lambda)P(q|G)) \quad (4.16)$$

P (q | db_i) is the probability of generating a specific term q by the content description of the i^{th} information source. P (q | G) is the probability of generating this term by a global unigram language model, which is obtained by collapsing resource descriptions of all the information sources together and building a single unigram language model. The linear interpolation constant λ is introduced to smooth the source-specific language model with the global language model and is usually adjusted in the range of 0 to 1 (It is set to 0.5 in this work).

To calculate the probabilities of P (db_i | Q) in Equation 4.15, it is necessary to estimate the information source prior probabilities of P (db_i). If a simple uniform distribution is used to set the prior probabilities, the values of P (db_i | Q) will be totally determined by the query likelihood P (Q | db_i). In this case, it is not difficult to show that the extended language model resource selection algorithm and the original KL-divergence algorithm in Equation 4.4 are actually equivalent by simply taking the logarithm of Equation 4.15 and noticing that the term of $\sum_{q \in Q} P(q|Q)\log(q|Q)$ in Equation 4.4 is a source-independent constant.

Our strategy to incorporate the information source size factor into this extended language model resource selection algorithm is to assign the information source prior probabilities according to the information source sizes. This is a natural idea as large information sources should be more favorable than small information sources when we are ignorant of the information need. This can be explained by the following example. In a federated search environment which contains two information sources A and B, the source A contains 10 documents and the source B contains 5 documents. Without knowing the information need, all the documents are treated equally. Therefore, the source A can be expected to have two times more useful information than the source B. Formally, the prior distribution is calculated as:

$$P(db_i) = \frac{\widehat{N}_{db_i}}{\sum_i \widehat{N}_{db_i}} \tag{4.17}$$

Finally, Equations 4.16 and 4.17 are plugged into the language model resource selection framework in Equation 4.15, and the information sources can be ranked according to the conditional probabilities of P (db_i | Q) as follows:

$$P(db_i|Q) \propto \prod_{q \in Q} (\lambda P(q|db_i)+(1\text{-}\lambda)P(q|G))^* \frac{\widehat{N}_{db_i}}{\sum_i \widehat{N}_{db_i}} \tag{4.18}$$

The new extended language model resource selection algorithm is denoted as the LM/Ext resource selection algorithm in this work.

The above extensions of the CORI and KL-divergence resource selection algorithms use different methods to incorporate the information source size factor, which promises to improve the original big document approach. However, these extensions still do not directly address the high-recall goal of information source recommendation application for maximizing the number of relevant documents contained in selected information sources. For example, the normalization approach of the extended CORI algorithm utilizes the source size factor to estimate the actual corpus statistics like document frequencies. However, each information source is still treated as a single large document and the source size statistics are not directly used to estimate the number of relevant documents.

The extended language model resource selection algorithm goes a step beyond the extended CORI algorithm. The likelihood probability P (Q | db_i) in Equation 4.15 can be seen as the average value of the query generation probabilities of all the documents in the i^{th} information source, and the source prior probability P (db_i) is introduced to reflect the effect of the information source sizes. If P (Q | db_i) is indicative of whether on average a single document in the i^{th} information source is relevant or not, the value P (db_i | Q) that incorporates source size estimate as prior can be related with the number of relevant documents that this information source contains.

Although the extended language model resource selection algorithm goes a step further than the extended CORI algorithm to approximate the high-recall goal of the information source recommendation application, it still does not explicitly deal with individual documents, which may be a serious problem. That can be explained by the following example. An information source A contains one very long relevant document and another nine irrelevant documents, while another source B contains nine short relevant documents and another one long irrelevant document. The information source A and the information source B may have the same unigram language models such that P (Q | db_A) is equal to P (Q | db_B). Furthermore, as these two information sources contain the same number of documents, P (db_A) is also equivalent to P (db_B) and therefore P (db_A | Q) is equal to P (db_B | Q). The above derivation suggests that the two information sources are equally valuable to us. However, this is not consistent with our expectation as selecting source B gives us nine relevant documents while selecting source A only returns one.

The above discussion indicates that a more effective resource selection algorithm should explicitly consider the individual documents in each information source. In other words, the probability of relevance for each document in the information sources should be estimated one by one in order to achieve more accurate resource selection results.

4.3 Relevant document distribution estimation (ReDDE) resource selection algorithm

The relevant document distribution estimation (ReDDE) resource selection algorithm [Si & Callan, 2003a] was proposed to turn away from the "big document" resource selection approach. It was pointed out that the goal of an information source recommendation system is to select a fixed number of information sources with the largest number of relevant documents. The ReDDE algorithm accomplishes this goal by explicitly estimating the distribution of relevant documents across all the information sources and ranking the information sources according to the distribution.

Formally, the number of documents relevant to a query Q in the i^{th} information source db_i is estimated as follows:

$$\mathrm{Rel\_Q(i)} = \sum_{d \in db_i} \mathrm{P(rel|d)} * \mathrm{P(d|db_i)} * \mathrm{N}_{db_i} \qquad (4.19)$$

where N_{dbi} denotes the number of documents in the i^{th} information source and the probability P (d | db_i) is the generation probability of a particular document d in this information source. If all the documents in this information source can be downloaded and considered in Equation 4.19, this probability as P (d | db_i) will be 1/ N_{dbi} and it can be cancelled with N_{dbi}. This indicates that we want to sum up the probabilities of relevance for all individual documents. However, in uncooperative federated search environments, it is only possible to access the sampled documents, and the actual information source size N_{dbi} is replaced by its corresponding estimate $\widehat{N}_{db_i}$. As long as the sampled documents are representative, Equation 4.19 can be approximated as:

$$\text{Rel\_Q(i)} \approx \sum_{d \in db_{i\_samp}} P(rel|d) * SF_{db_i} \tag{4.20}$$

where db_{i_samp} denotes the set of sampled documents from the i^{th} information source. The idea behind this equation is that when one sampled document from the information source is relevant to a query, we believe that there are about SF_{dbi} similar documents in the complete information source, which are also relevant to the query.

The only item left to be estimated in Equation 4.20 is P (rel | d), which denotes the probability of relevance of an arbitrary sampled document. How to calculate this probability is a fundamental problem of information retrieval research. Many retrieval algorithms such as the Bayesian belief network [Turtle, 1990] and the language model [Ponte & Croft, 1998; Zhai & Lafferty, 2003] have been proposed to address this problem. This problem is not solved in the general case here. Instead, the probability of relevance is approximated in a simple way as described below.

To approximate the probability of relevance, we take advantage of available information from resource descriptions. Particularly, the query-based sampling method generates the content descriptions, which is used for estimating information source sizes. Furthermore, the sampled documents are combined to build the centralized sample database (CSDB), which plays an important role for approximating the probabilities of relevance for all the documents.

The procedure of estimating the probabilities of relevance is associated with a complete version of the centralized sample database, which is called the *centralized complete database* (CCDB). The centralized complete database is the union of all the individual documents in available information sources. Of course, the centralized complete database does not exist; otherwise, a more effective ad-hoc retrieval method can be directly applied on the complete database instead of using the federated search solution. However, the centralized sample database is a representative subset of the centralized complete database and the statistics of the centralized complete database can be estimated by the statistics of the centralized sample database. An example in this section shows how to simulate the retrieval ranked list from the centralized complete database by the ranked list from the centralized sample database.

The probability of relevance is modeled as a step function with respect to the retrieval result from the centralized complete database. More specifically, an effective retrieval algorithm is applied on the

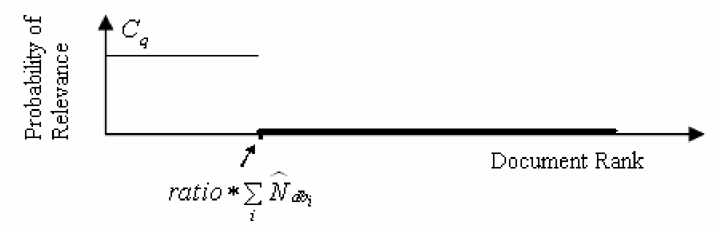

Figure 4.1: The curve of probability of relevance as a step function.

centralized complete database. The documents that rank at the top of the centralized complete database have the probabilities of relevance as positive constants, while the probabilities of relevance of all the other documents are zero. This idea can be formalized as:

$$P(rel|d) = \begin{cases} C_Q & \text{if } Rank_{CCDB}(Q,d) < ratio * \sum_i \widehat{N}_{db_i} \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

where $Rank_{CCDB}$ (Q , d) indicates the rank of document d in the retrieval results from the centralized complete database with respect to query Q. Ratio is a percentage threshold, which separates relevant documents from irrelevant documents. This formula is also visualized in Figure 4.1. Treating the curve of probability of relevance as a step function is a rough approximation. However, this is a common approach in information retrieval research especially when only very limited information is available. For example, the pseudo relevance query expansion method uses the top documents in the initial retrieval as relevant documents to extract expanded query terms [Xu & Croft, 1996]. Note that most prior pseudo relevance feedback research used a rank based threshold while a ratio based threshold is used here to accommodate the large variation of the information source sizes.

The retrieval results from the centralized complete database are not directly accessible, but they can be approximated by the retrieval results from the centralized sample database. More specifically, the user query is sent to search the centralized sample database by an effective ad-hoc retrieval algorithm, which is INQUERY [Turtle, 1999; Callan, Croft & Broglio, 1995] in this work. The obtained ranked list from the centralized sample database is used to construct the ranked list from the centralized complete database. Formally, the rank of a document in the centralized complete database is calculated by:

$$Rank_{CCDB}(Q,d) = \sum_{\substack{d_j: \\ Rank_{CSDB}(Q,d_j) < \\ Rank_{CSDB}(Q,d)}} SF_{db(d_j)} \tag{4.22}$$

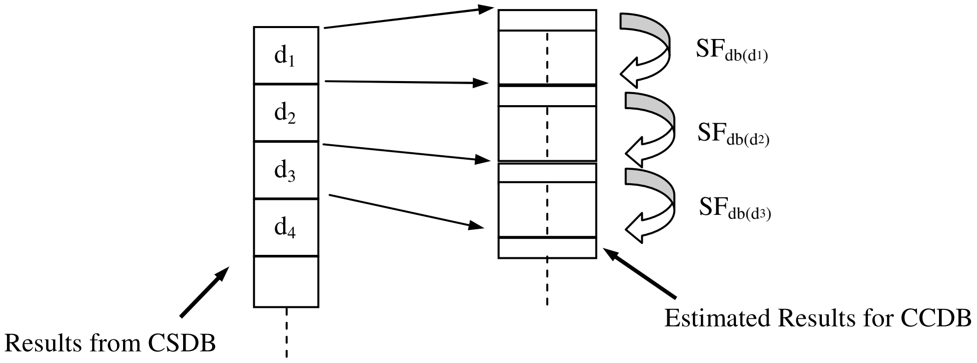

Figure 4.2: The approximation of the ranked list on centralized complete database (CCDB) by the ranked list on centralized sample database (CSDB).

where $\mathrm{Rank}_{\mathrm{CSDB}}$ (Q , d) denotes the rank of a document in the centralized sample database, and db (d_j) indicates which information source the document d_j is from. The approximation procedure of the ranked list from the centralized complete database by the ranked list from centralized sample database is shown in Figure 4.2.

Given Equations 4.21 and 4.22, the number of relevant documents in each information source can be estimated by Equation 4.20. However, note that there still exists a query dependent constant C_Q introduced by Equation 4.21, which makes the calculation of the exact number of relevant documents intractable. Therefore, the number of relevant documents across available information sources is further normalized to compute the distribution of the relevant documents as follows:

$$\mathrm{Dist\_Rel\_Q(i)} = \frac{\mathrm{Rel\_Q(i)}}{\sum_{i} \mathrm{Rel\_Q(i)}} \tag{4.23}$$

Both the numerator and the denominator in Equation 4.23 contain the query specific constant C_Q, which is cancelled during the calculation. Therefore, Equation 4.23 provides us the computable distribution of relevant documents. It serves as the criterion to rank the information sources. Based on this criterion, the information sources with the largest number of relevant documents are selected as they contain larger fractions of relevant documents estimated by Equation 4.23. This indicates that the ReDDE algorithm explicitly meets the high-recall goal of the information source recommendation application, which is to select information sources that contain the largest number of relevant documents.

4.4 Evaluation methodology and experimental results

This section first discusses experimental methodology to evaluate resource selection algorithms. Furthermore, a series of experiment results conducted on testbeds with different characteristics is shown to compare the effectiveness of the basic version of CORI algorithm, the basic version of the KL divergence algorithm, the extended CORI algorithms, the extended language model resource selection algorithm and the ReDDE algorithm.

4.4.1 Evaluation methodology

The desired goal of resource selection algorithms of an information source recommendation system is to select a small number of information sources with the largest proportion of relevant documents. They are typically compared with a desired information source ranking called *Relevance-Based Ranking*, where the information sources are ranked by the actual number of relevant documents they contain. Let E and B be the ranking provided by the resource selection algorithm being evaluated and the oracle relevance-based ranking respectively, and B_i and E_i denote the number of relevant documents in the i^{th} ranked information source of B and E respectively. The recall metric R_k for comparison is defined as follows:

$$R_k = \frac{\sum_{i=1}^{k} E_i}{\sum_{i=1}^{k} B_i} \qquad (4.24)$$

This comparison metric measures the percentage difference of the relevant documents included by the ranking generated by the algorithm in evaluation and the ranking by the relevance-based ranking algorithm, which is always between 0 and 1. Therefore, at a fixed k, a larger value of R_k indicates a better information source ranking result.

The experiments were conducted on four testbeds with six resource selection algorithms. The four testbeds cover a wide range of federated search environments: relatively uniform information source size and content distributions (i.e., Trec123_100Col), moderately skewed information source size and moderately skewed content distributions (i.e., Trec4_kmeans), bimodal information source size distribution where a large proportion of relevant documents are in the large information sources (i.e., Relevant) and bimodal information source size distribution where a small proportion of relevant documents are in the large information sources (i.e., Nonrelevant). Detailed information of these testbeds can be found in Chapter 2.

The six resource selection algorithms are the basic CORI algorithm, the CORI/Ext1 algorithm (Section 4.2), the CORI/Ext2 algorithm (Section 4.2), the basic KL-divergence algorithm, the extended language model

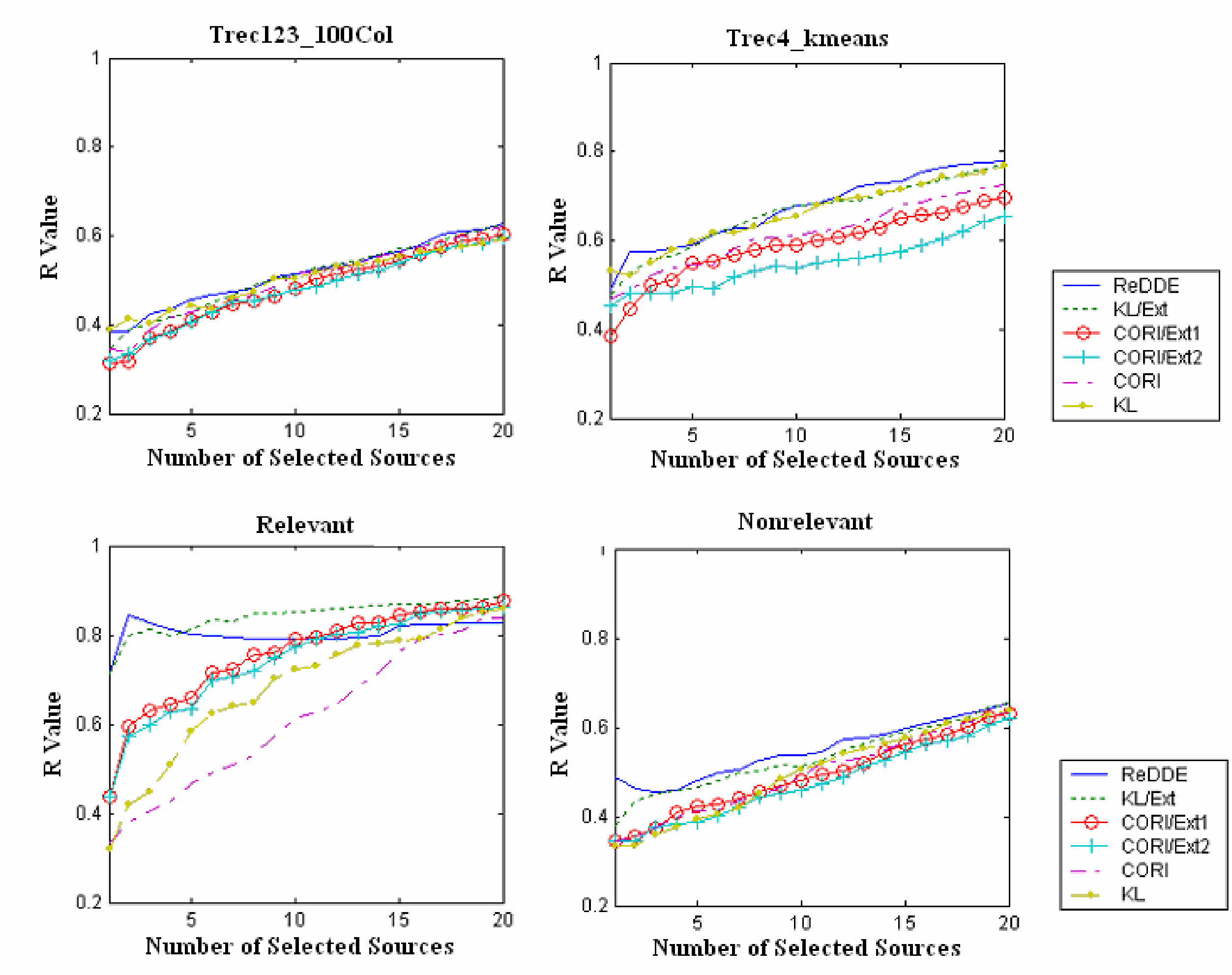

Figure 4.3: Resource selection experiments on four testbeds (with database size estimates).

resource selection algorithm (LM/Ext) (Section 4.2) and the ReDDE algorithm (Section 4.3).

All the resource selection algorithms in the experiments used query-based sampling method to acquire resource representations. Specifically, about 80 queries were sent to search each information source and 300 documents were downloaded for each source. The information source size estimates were obtained by the original Sample-Resample method as described in Chapter 3.

4.4.2 Experiment results

In the first set of experiments, all resource selection algorithms used information source size estimates acquired by the Sample-Resample method (the estimation error rate ranges from 15% to 30%). The experiment results are shown in Figure 4.3. The basic versions of the CORI and KL-divergence algorithms, which do not consider information source size factor, did reasonably well on the testbeds of Trec123_100Col with relatively uniform information source size distribution and Trec4_kmeans with moderately skewed source size distribution. However, their accuracies on the relevant and nonrelevant testbeds with more skewed information source size distributions were not very satisfactory. This indicates that the information source size factor plays a very important role in resource selection, and it should be incorporated into robust resource selection algorithms designed to work in a wide range of federated search environments.

The two extensions of the CORI algorithm showed inconsistent behavior on these four testbeds. They were better than the basic CORI algorithm on the relevant testbed, about the same as CORI on the Trec123_100Col and nonrelevant testbeds, but even worse on the Trec4_kmeans testbed. This does not mean other more sophisticated modifications of the basic CORI algorithm can not work well in the environments with skewed information source size distributions, it only indicates that our two extensions of the CORI/Ext1 and CORI/Ext2 are not consistently successful.

In contrast to the extensions of the CORI algorithm, the extended language modeling resource selection algorithm was at least as effective as the basic KL-divergence algorithm and all the variants of CORI algorithms on the Trec123_100Col testbed, or was even better on the Trec4_kmeans, relevant and irrelevant testbeds. The advantage of the extended language modeling resource selection algorithm can be attributed to the better approximation of the number of relevant documents among available information sources than the simple normalization method of the extended CORI algorithms, which still follow the “big document” approach (Section 4.2).

Furthermore, the ReDDE resource selection algorithm explicitly estimates the probability of relevance of each individual document, and thus approximates the distribution of relevant documents. It was about as accurate as the extended language model resource selection algorithm on all the four testbeds and was more robust than all other methods. This advantage suggests that our discussion of the deficiency of “big document” resource selection algorithms is correct. However, no strong evidence shows that ReDDE algorithm had a large advantage over the extended language model resource selection algorithm on these four testbeds. The disadvantage of the extended language model resource selection algorithm is discussed in Section 4.2 and an example is proposed to demonstrate this weakness in a particular type of federated search environments.

In order to study the effects of imperfect source size estimates on resource selection, the second set of experiments was designed to use actual values of the information source sizes instead of the estimates within resource selection algorithms. The experiment results are shown in Figure 4.4, where the four algorithms,

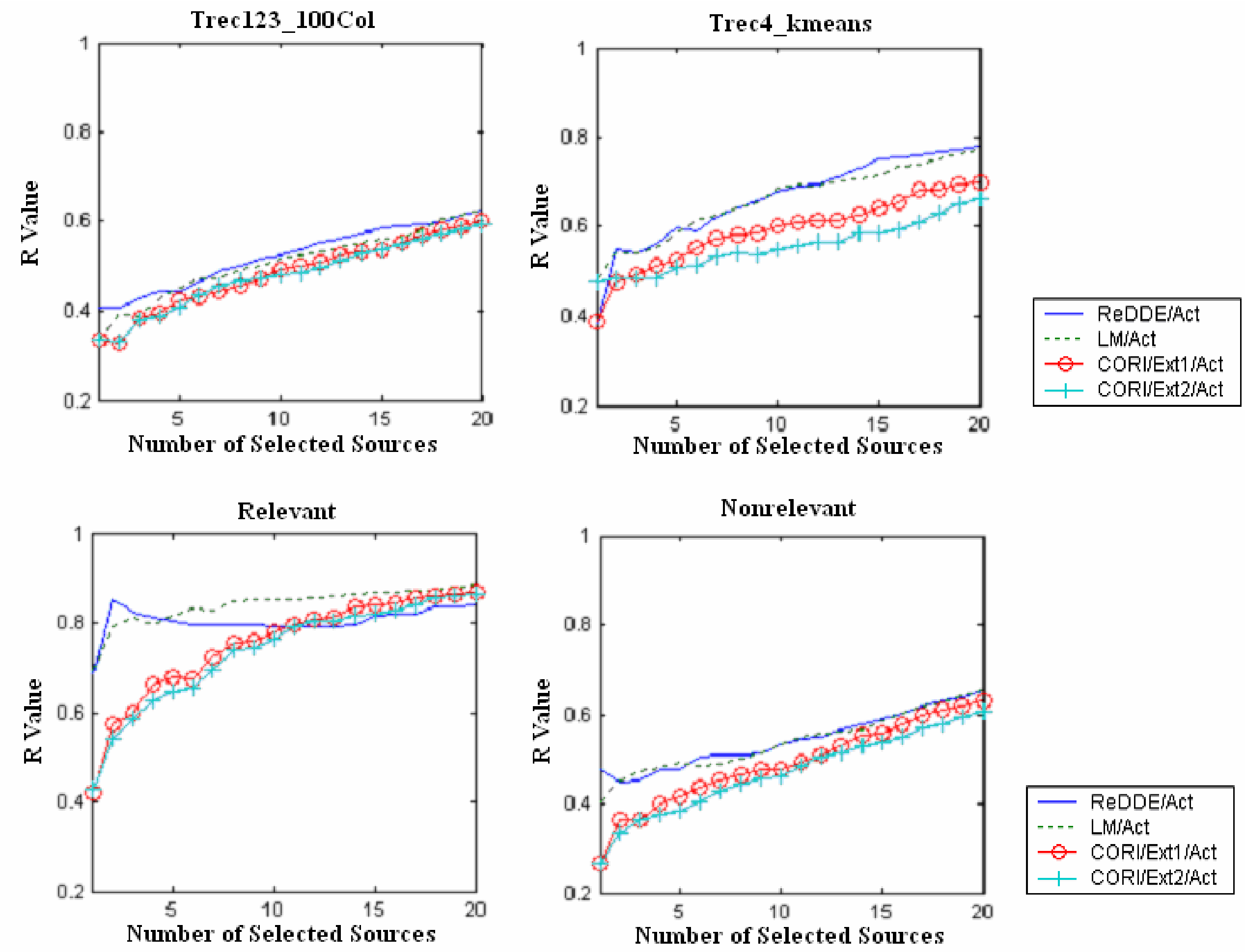

Figure 4.4: Resource selection experiments on four testbeds (with actual database sizes)

which consider information source size factor, were evaluated. It can be seen from Figure 4.4 that the ReDDE and the extended language model resource selection algorithms were more robust than the two extensions of the CORI algorithm, which is consistent with the results of the first set of experiments using estimated database sizes. Another observation from Figure 4.3 and Figure 4.4 is that all the four resource selection algorithms using the estimated information source sizes were almost as accurate as algorithms using actual information source sizes. This suggests that although the Sample-Resample algorithm gives imperfect information size estimates (on average, the estimation error rate ranges from 15% to 30% for the four testbeds), the effects of the inaccurate source size estimates are much less noticeable in the resource selection experiments. This result can be explained by the fact that although some documents are not counted by the Sample-Resample algorithm, they may contain very little valuable information and are not relevant to most user queries so that they do not have much impact on the accuracies of the resource selection algorithms.

4.5 Summary

A large body of resource selection algorithms has been proposed in previous research. Most of them are tied with the "big document" approach. The "big document" approach treats available information sources as single big documents and calculates similarity between the single big documents and user queries to rank available sources. However, the "big document" approach does not consider how many individual documents are relevant or not and thus cannot optimize for the high-recall goal for resource selection algorithms for information source recommendation. Particularly, empirical studies in prior research and in this chapter demonstrate the deficiency of the "big document" approach within federated environments of skewed information source sizes.

Several new variants of two well-known "big document" resource selection methods (CORI and KL-divergence) are proposed in this chapter to incorporate source sizes into resource selection decision. However, detailed theoretical analysis or empirical results suggest that the extensions are not capable of optimizing the resource selection results for the high-recall goal.

Based on this observation, a new resource selection algorithm called relevant document distribution estimation (i.e., ReDDE) is proposed in this chapter to turn away from the "big document" approach. It views information sources as document repositories instead of single big documents. The ReDDE algorithm directly optimizes the high-recall goal by estimating the distribution of relevant documents across available information sources and ranking the sources accordingly. Particularly, it views the centralized sample database as a set of representative documents of complete information sources. The ReDDE algorithm estimates the retrieval results from a centralized complete database by the retrieval results from the centralized sample database and then infers the distribution of all relevant documents across available sources. An extensive set of experimental results within different federated search environments demonstrates the advantage of the ReDDE algorithm over several prior state-of-the-art "big document" resource selection algorithms.

The ReDDE resource selection algorithm models the curve of probability of relevance as a step function with respect to the retrieval result from the centralized complete database. This is a rough approximation. It is empirically effective and works without any training data. A refined method is to estimate the curve of probability of relevance with a small set of training queries. This method can generate more robust resource selection decision. This refined method is introduced and studied within the unified utility maximization framework in Chapter 7.

The success of the ReDDE algorithm (also other resource selection algorithms) depends on the quality of the sampled documents. An interesting direction in future research is to study the reliability of the ReDDE resource selection algorithm with various amounts of sampled documents. It is also helpful to study the variance of the resource selection accuracy of the ReDDE algorithm as well as other algorithms by using multiple sets of sampled documents.

Chapter 5: Results Merging

Federated document retrieval systems automatically search selected information sources and retrieve ranked lists of documents from these information sources. There are several choices to organize these returned results. One approach is to display these results side by side in the same page or in different result pages. This method makes sense when there are a limited number of selected information sources (e.g., less than 3) and the results from these selected information sources contain different types of contents (e.g., one biological source may return a table of gene co-expression data and another source may return a patent document). However, when a larger number of information sources (e.g., more than 5) are selected and most information sources return the same type of contents (i.e., text documents), it might be distracting to display the results separately and users often prefer a single merged ranked list. In the latter case, a results merging component is required in a federated document retrieval system.

The results merging task is difficult because document scores returned by different sources cannot be compared directly. There are at least two reasons that different sources produce incomparable ranked lists: i) different information sources may use different retrieval algorithms so that the ranges of document scores in the ranked lists may be totally different; and ii) even when two information sources use the same type of retrieval algorithms, the corpus statistics (e.g., vocabulary size, inverse document frequency, etc.) used to calculate document scores are often quite different in different sources.

This chapter first introduces previous research on results merging and discusses the advantages and disadvantages of these methods. It then proposes a *semi-supervised learning approach*, which utilizes part of the documents acquired by query-based sampling as training data and learns source-specific and query-specific models to transform all the source-specific scores into comparable document scores. An extensive set of empirical studies in both research environments and a real world application is conducted to show the effectiveness of the new results merging approach.

5.1 Prior research on results merging

Results merging has received limited attention in prior research, partially because of the misunderstanding that it is similar to the meta search problem. However, the results merging subproblem of federated document retrieval is different from meta search [Lee, 1997; Aslam & Montague, 2001; Manmatha, Rath & Feng, 2001]. In meta search, multiple retrieval algorithms are applied to a single centralized database or to sources with overlapping contents. Meta search algorithms depend on the fact that there are multiple scores for a single document. In contrast, in the results merging task of federated search, the contents of the information sources are usually independent and we cannot expect many documents to appear in two or more ranked lists. Note that some people use "metasearching" as having the same meaning as federated search, but this is not the case discussed in the dissertation.

One simple solution in cooperative environments is to make every information source use the same type of retrieval algorithm and the same corpus statistics [Viles & French, 1995; Xu and Callan, 1998; Xu & Croft, 1999], for example, by imposing a common set of corpus statistics among all sources. An alternative approach is for each information source to return its term frequency information for each retrieved document and each query term so that the search client can compute a consistent set of document scores using global corpus statistics [Kirsch, 1997]. These methods are quite accurate, but all of them require significant cooperation from information sources such as using the same type of retrieval algorithm or providing term frequency information upon request, which is not a valid assumption in uncooperative environments.

Another simple solution is the round robin method [Voorhees et al., 1995], which can be easily applied in uncooperative environments. It only utilizes the document rank information in individual ranked lists and the source rank information from resource selection algorithms. Specifically, it chooses the first document returned by the first selected information source as the first document in the merged list and the first document from the second selected information source as the second one, and so on. Round robin is a common choice when source-specific document scores are completely incompatible. However, the disadvantage is that less relevant sources contribute as many documents to the merged ranked lists as more relevant sources. The weighted round robin is an improved method, which allows information sources to contribute documents in proportion to their expected values. These methods are simple and easy to apply in uncooperative environments but are not very effective [Savoy & Rasolofo, 2000].

The CORI results merging formula [Callan, Lu & Croft, 1995; Callan, 2000] follows the idea that both the documents from information sources with high selection scores and the high-scoring documents from information sources with lower selection scores should be favored. It uses a simple heuristic formula to normalize source-specific document scores. The formula is a linear combination of the information source selection scores and the document scores from individual ranked lists. First, associated with the CORI resource selection algorithm, the information source selection scores are normalized as:

$$S'(db_i) = \frac{S(db_i)\text{-}S(db_{min})}{S(db_{max})\text{-}S(db_{min})} \tag{5.1}$$

Equation 5.1 normalizes information source selection score of the ith information source to the range of [0, 1]. The raw score S (db_i) of the ith information source is the information source belief P (Q | db_i) in the CORI resource selection algorithm (described in Chapter 4). S (db_{min}) and S (db_{max}) are calculated by setting the T component in Equation 4.3 (i.e., in Section 4.1) to 0 and 1 respectively. In a similar manner, the source-specific document scores are normalized as:

$$S'(d_{ij}) = \frac{S(d_{ij})\text{-}S(d_{i\_min})}{S(d_{i\_max})\text{-}S(d_{i\_min})} \tag{5.2}$$

S'(d_{ij}) is the normalized document score for the jth document from the ith information source. In cooperative environments, the maximum document score S (d_{i_max}) and the minimum score S (d_{i_min}) are provided by the ith information source, otherwise they are simply set to the maximum and minimum document scores returned by the information sources in uncooperative environments.

Finally, the source-independent comparable scores are calculated as:

$$S_C(d_{ij}) = \frac{S'(d_{ij})+0.4*S'(d_{ij})*S'(db_i)}{1.4} \tag{5.3}$$

The CORI results merging formula has been shown to be effective in previous research [Callan, Lu & Croft, 1995; Callan, 2000]. It can be easily applied in uncooperative environments and in this dissertation it is used as a baseline algorithm to compare with new proposed methods.

Logistic transformation models have been used for results merging in previous research [Le Calv & Savoy, 2000; Savoy, 2002]. This method utilizes human-judged training data to build source-specific query-independent logistic models for different information sources to transform source-specific scores into source-independent scores. The source-specific model provides a more theoretically solid solution than the round robin methods and the CORI formula. It has been shown to be effective in previous research [Le Calv & Savoy, 2000; Savoy, 2002; Savoy, 2003]. However, it uses human-judged relevance information to build separate models for the information sources, so the corresponding human efforts may not be affordable when there are a large number of information sources. Another potential weakness of this approach is that a single query-independent transformation model is applied for each information source but the score characteristics change from query to query. This indicates that query-specific and language-specific transformation models should be more favorable.

The brief review of existing results merging algorithms tells us that all these algorithms try to simulate document ranked lists as if all the documents were stored in a single, global database. However, they are not satisfactory solutions in uncooperative environments as: i) most accurate algorithms require information sources to calculate consistent document scores or recalculate document scores in the central search client by making assumptions which are not practical in uncooperative environments; ii) the CORI and the round robin methods, which only utilize information from individual ranked lists and resource selection results, are simple and efficient but not very accurate; and iii) the previous work of utilizing score transformation models is associated with large cost of human relevance judgment and is not very effective either since the models are query-independent.

5.2 Semi-supervised learning results merging approach

The goal of the Semi-Supervised Learning (SSL) results merging algorithm [Si & Callan, 2002a; Si & Callan, 2003b] is to effectively and efficiently produce a single ranked list of documents that approximates the ranked list in the centralized environment. This method utilizes centralized sample database as an extra source of information. Specifically, a user query is sent to search the documents in the centralized sample database with an effective centralized retrieval algorithm to acquire source-independent scores. The set of sampled documents that exists in both centralized sample database and source-specific ranked lists has both source-independent scores and source-specific scores. These documents serve as the training data to calculate source-specific models that transform all source-specific document scores into the scores similar to what the centralized sample database would have produced for those documents. The transformation models are applied on all returned documents and these documents are merged into a single ranked list according to the calculated source-independent scores.

The semi-supervised learning algorithm does not require human relevant judgment as training data. It uses the automatically calculated centralized document scores as surrogates to build transformation models. This is much more efficient than using human relevance judgments as training data when there are many information sources. The algorithm is called semi-supervised learning because no human supervision is required and the training data is generated automatically. Note this is different from the concept of the semi-supervised learning [Zhu, 2005] in the machine learning community, where a learning algorithm utilizes both labeled and unlabeled training data.

SSL algorithm makes two assumptions: i) some documents retrieved from the selected information sources also exist in the centralized sample database; and ii) given both the source-independent scores and the source-specific document scores of these overlap documents, a linear function can be learned to transform the source-specific scores into the corresponding source-independent scores.

The first assumption indicates the availability of training data. It may be questionable, but one fact enhances the possibility that enough overlap documents can be found: an information source is selected only if the language model derived from its *sampled documents* matches the query, thus some of these documents are likely to be retrieved from this information source. Experiments are presented below to study this problem. Furthermore, when there are not enough overlap documents, a variant of the SSL algorithm is proposed to download some retrieved documents on the fly to create extra training data.

We do not argue that it is the best choice to use linear functions as the transformation models in the second assumption. Other more sophisticated transformation models may be more accurate in some cases. The linear transformation model is chosen here because: i) the linear transformation model can be computed very efficiently and it only requires a very small amount of training data (i.e., as few as two overlap documents), which is very practical when efficiency is of high priority; and ii) the CORI results merging algorithm described in Equation 5.3 can be represented as a linear model, which suggests the effectiveness of using a linear function.

With enough overlap documents that have both source-specific document scores and source-independent scores, the linear model that most accurately transforms the source-specific document scores to the source-independent scores can be estimated. Formally, the linear transformation model is calculated as follows:

$$(a\ ,\ b)^{*} = \underset{(a,b)}{\operatorname{argmin}} \sum_{j} (a * x_{j} + b * y_{j})^{2} \tag{5.4}$$

where (x_j , y_j) is the source-specific score and source-independent score of the j^th^ overlap document respectively. (a , b)* denotes the parameters of the desired linear model. This model can be used to transform source-specific scores of other returned documents into the corresponding source-independent scores.

It is most likely that the information sources in uncooperative environments such as large organizations or the Web are searched by multiple types of search engines (the multiple engine-types case). Different retrieval algorithms may generate source-specific scores with quite different score ranges, and it is not likely that a single linear model can transform all these scores into source-independent scores. Therefore, multiple linear transformation models should be learned for the multiple engine-types case. As our focus is uncooperative federated search environments, we present research for multiple engine-types case in this work.

After selected information sources return source-specific ranked lists, the first step is to identify the overlap documents for each information source that appear in both the centralized sample database and the corresponding ranked list. Formally, for each overlap document d_{ij} that comes from the i^{th} information source, the source-independent score is denoted as $S_c(d_{ij})$ and the normalized source-specific score is denoted as $S_i(d_{ij})$. Given the overlap documents and their scores, the goal is to estimate a linear model for this information source that transforms all its source-specific document scores into the source-independent scores as $S_c(d_{ij}) = a_i* S_i(d_{ij}) + b_i$. The regression problem over all the training data from the i^{th} information source can be formulated in the matrix representation as follows:

$$\underset{X}{\begin{bmatrix} S_i(d_{i,1}) & 1 \\ S_i(d_{i,2}) & 1 \\ \ldots\ldots & 1 \\ S_i(d_{i,m}) & 1 \end{bmatrix}} * \underset{W}{\begin{bmatrix} a_i & b_i \end{bmatrix}^T} = \underset{Y}{\begin{bmatrix} S_c(d_{i,1}) \\ S_c(d_{i,2}) \\ \ldots\ldots \\ S_c(d_{i,m}) \end{bmatrix}} \tag{5.5}$$

where a_i and b_i are the two parameters of the linear model for the i^{th} database. Call these matrices X (source-specific scores and constants), W (the model parameters of the linear transformation model) and Y (the set of source-independent document scores). Simple mathematic manipulation shows that the solution can be expressed as follows:

$$W = (X^T X)^{-1} (Y^T X) \tag{5.6}$$

This solution is acquired by minimizing the squared error criterion described in Equation 5.4.

The same procedure can be repeated for all selected information sources to build source-specific linear models to transform the source-specific document scores into the corresponding source-independent scores. Finally, the merged ranked list is constructed by sorting all returned documents by the corresponding source-independent scores.

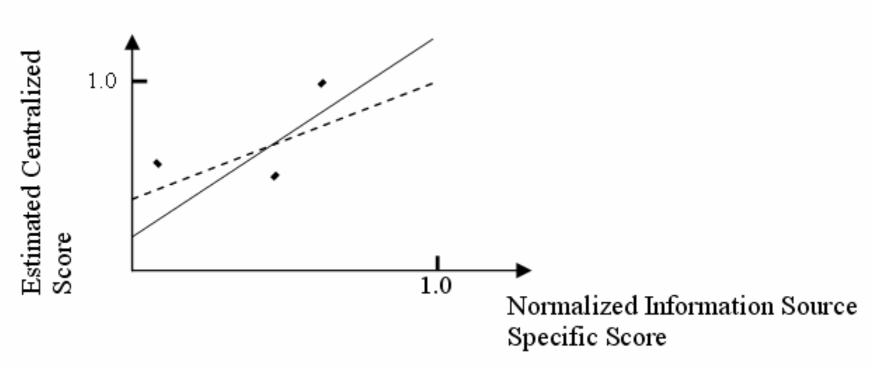

Figure 5.1: Adjust the bias problem of linear regression model. The solid line is the original model and the dash line is the new model.

The above algorithm transforms source-specific document scores to centralized comparable document scores. However, in real world applications, information sources may return source-specific ranked lists without document scores. It is straightforward to convert document ranks to pseudo document scores and apply the above algorithm. More detailed information can be found in the empirical study of the FedLemur system in (Section 5.3.5).

The main procedure of building the linear models is described as above. Some adjustments are also utilized to make the models more robust. Theoretically, as few as two data points are needed to train a linear model. However, more training data usually generates more accurate models, so we require that at least three overlap documents should exist to calculate a linear model. When an information source does not have enough training data, it is called an "unfavorable" information source. One possible way to treat the unfavorable information sources is to simply ignore the returned results from them, as it is believed that a "favorable" information source, which contains a lot of relevant documents, tends to have more relevant documents in the centralized sample database and thus have more overlap documents as the training data. However, when there are too many information sources without enough training data, it may suggest that the SSL approach is not a good choice for this query, and the algorithm backs off to the CORI merging formula, which does not require any training data at all. The back off threshold is set empirically as 40%. For example, when there are more than 4 information sources out of totally 10 selected information sources containing less than 3 overlap documents, the algorithm backs off to the CORI merging formula.

On the other hand, when more than enough training data is available for an information source, the algorithm can be selective to choose the important ones for accuracy and efficiency. As the high-precision metric indicates that it is more important to be accurate at the top of a ranked list, at most top 10 overlap documents from each selected information source are selected as the training data.

Another adjustment is to correct the bias problem of anomalous linear models. INQUERY [Turtle, 1990; Callan, Croft & Broglio, 1995] is used as the centralized retrieval algorithm in this work to produce source-independent document scores, and thus it is impossible to produce a source-independent document score larger than 1. A learned linear transformation model is anomalous when the model generates source-independent document scores great than 1. A document with a score larger than 1 is not a serious problem by itself, but this indicates that the model may be too highly biased and the documents from this information source may have been given excessive advantages.

This bias problem is addressed by replacing the original linear model with another one that intersects the point of (1 , 1) and is closest to the original linear model (as shown in Figure 5.1). Formally, let y = ax + b be the original model and y = a'x + b' be the new model. The new linear model is obtained by solving the following problem:

$$\begin{gathered}(a' , b') = \underset{a', b'}{\operatorname{argmin}} \int_0^1 [(a'-a)*x+(b'-b)]^2 dx \\ \text{s.t.} \quad a'+b'=1\end{gathered} \tag{5.7}$$

Simple mathematical manipulation shows the following update formula for the new model:

$$a' = \frac{3-a-3b}{2} \qquad b' = 1-a' \tag{5.8}$$

These small adjustments have been shown to slightly improve the results merging accuracy in empirical studies. Furthermore, this adjustment can be generalized to other cases when different centralized retrieval algorithms (which may produce different maximum retrieval scores) than INQUERY are utilized.

5.3 Evaluation methodology and experimental results

An extensive set of experiments was conducted to study the effectiveness of the semi-supervising learning results merging algorithm under a variety of conditions. Specifically, this section first describes experimental methodology for results merging algorithms. Next, the sufficiency problem of overlap documents is studied for the case when long ranked lists can be acquired from the selected information sources, followed by the experiment results to compare the CORI results merging formula and the SSL algorithm. The scenario when only short ranked lists are available from selected information sources is also addressed and a variant of the SSL algorithm is proposed to download a minimum number of documents as additional training data. Experiments are conducted to study the effectiveness of the new SSL algorithm. Finally, empirical studies with the FedLemur project are presented to show the behavior of the CORI results merging formula and the SSL algorithm in this real world application.

5.3.1 Evaluation methodology

The experiments in this chapter were conducted on two testbeds in research environments and also with the FedLemur project. This section first introduces the experimental methodology within the research environments.

Two research testbeds were used in the experiments, namely the Trec123_100Col testbed and the Trec4_kmeans testbed. Trec123_100Col is organized by source and publication date. The contents of the information sources are relatively heterogeneous. Trec4_kmeans is organized by topic, where the information sources are relatively homogenous and the word distribution is more skewed than the Trec123_100Col testbed.

The two testbeds provide different advantages and disadvantages to different components of a federated search system. As the contents of information sources are more homogenous and relatively different from each other, the Trec4_kmeans testbed is generally considered to be easier for resource selection than the Trec123_100Col testbed where the contents are much more heterogeneous. In contrast, the Trec123_100Col testbed is believed to be easier for results merging than the Trec4_kmeans testbed, as the documents ranked at the nth positions by different information sources on the Trec123_100Col testbed are more comparable than the corresponding set of documents on the Trec4_kmeans testbed. The difficulty of results merging for a similar testbed with skewed word distribution has been reported in previous research [Larkey, Connell & Callan, 2000].

Three effective retrieval algorithms introduced in Chapter 2 as INQUERY [Callan, Croft & Broglio, 1995], language model [Lafferty & Zhai, 2001] and vector-space algorithms were chosen in the experiments. All these three algorithm were implemented with the Lemur toolkit [Ogilvie & Callan, 2001b], and they were assigned to the information sources in a round-robin manner (more details in Section 2.3).

In our experiments, the content resource descriptions were created by query-based sampling. About 80 single-word queries were sent to each information source to download 300 hundred documents, and the top 4 (or fewer when there were not enough) documents from each ranked list were acquired. The sampled documents were collected to form the centralized sample database, which contained 30,000 documents (100 information sources time 300 documents per information source). The INQUERY retrieval algorithm was used as the search algorithm for the centralized sample database.

The CORI resource selection algorithm was used to rank the information sources for the experiments in this chapter. It was chosen for two reasons: i) CORI resource selection algorithm has been commonly used in many federated document retrieval applications; and ii) information source selection scores from the CORI resource selection algorithm are required by the CORI results merging formula, which serves as the baseline in the experiments. Document retrieval experiments with other more effective resource selection algorithms are discussed in Chapter 6.

5.3.2 Experimental results: Overlap documents

A sufficient amount of training data is crucial to the SSL results merging algorithm. Many factors such as the information source sizes, the number of sampled documents from the information sources, the lengths of the returned ranked lists and the query characteristics determine the number of overlap documents. Generally, it can be expected that the number of overlap documents will be larger when more documents are acquired during query-based sampling; the number may also be larger when queries are longer, when there are a limited number of documents in a particular information source, or when more documents are returned from each selected source.

A set of experiments was conducted to investigate the number of overlap documents in the case where a small proportion of documents are sampled (300 out of 10,000 on average for Trec123_100Col, 300 out of 5,600 on average for Trec4_kmeans) and long returned ranked lists are provided (up to 1,000 documents per selected information source). Requiring long returned ranked lists from information sources that only provide short ranked lists by single requests may cause substantial communication costs. Based on this observation, a variant of the SSL algorithm is introduced in Section 5.3.4 to work in the case of short ranked lists (e.g., 50 documents). As the accuracy of the SSL algorithm in the case of long ranked lists can serve as the baseline to compare its variant in the short ranked list case, the behavior of the SSL algorithm with long ranked lists is first studied.

Specifically, the experiments were conducted on two testbeds of Trec123_100Col and Trec4_kmeans. Both of them contain 100 information sources, and three retrieval algorithms of INQUERY, language model and vector space model were assigned to the information sources in a round-robin manner. Resource descriptions were built by query-based sampling to acquire 300 sampled documents from each information source. The

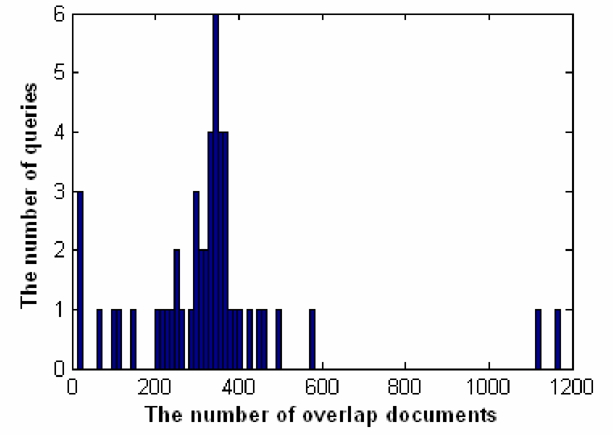

Figure 5.2: The histogram of overlap documents for 50 "title" queries on Trec123_100Col testbed.

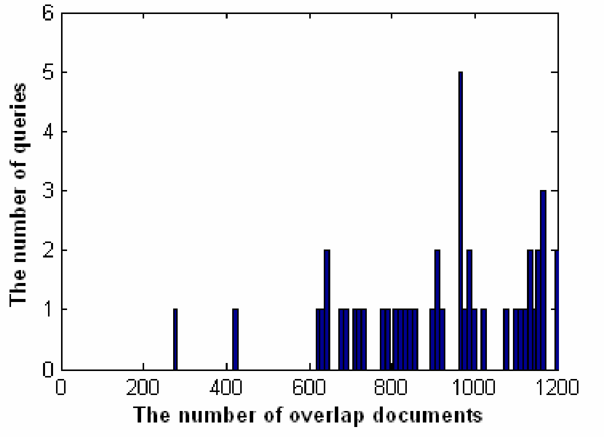

Figure 5.3: The histogram of overlap documents for 50 "description" queries on Trec4_kmeans testbed.

CORI resource selection algorithm was used to select 10 most relevant information sources, and result lists of up to 1,000 document ids with their source-specific scores were returned by the selected information sources.

Very few queries (3 out of 50) on the Trec123_100Col testbed and no query (0 out of 50) on the Trec4_kmeans testbed were short of overlap documents (i.e., more than 4 among the 10 selected information sources had less than 3 overlap documents). For more detailed information, Figures 5.2 and 5.3 show the histograms of the number of overlap documents for the queries on the Trec123_100Col testbed and the Trec4_kmeans testbed respectively.

The experiment results indicate that our assumption of sufficient number of overlap documents is satisfied for the environments where a small amount of documents are sampled and long ranked result lists can be returned. Particularly, the queries on the Trec4_kmeans testbed tend to have more overlap documents than the queries on the Trec123_100Col testbed, which is consistent with our expectation that the higher coverage of the sampled documents (i.e., on average, the information sources on the Trec4_kmeans testbed are smaller than the information sources on the Trec123_100Col testbed while all of them contribute 300 sampled documents), the longer the queries (i.e., the description queries on the Trec4_kmeans testbed are longer than the title queries on the Trec123_100Col testbed), the more number of overlap documents can be acquired.

5.3.3 Experimental results: Comparison with CORI results merging formula

This section presents experimental results to compare the accuracy of the CORI results merging formula and the SSL results merging algorithm for the multiple engine-types case on the two testbeds of Trec123_100Col and Trec4_kmeans. Three retrieval algorithms of INQUERY, language model or vector space models were assigned to the information sources in a round-robin manner. To investigate the effectiveness of results merging algorithm in a wide range of experimental configurations, the number of information sources selected to search for each query was varied by 3, 5 or 10. Each selected source provided long returned ranked lists (up to 1,000 documents per selected information source)

Detailed experiments results are shown in Tables 5.1 to 5.3. Several interesting issues can be observed from these experiment results. First, the SSL result merging algorithm was more accurate than the CORI merging formula in all configurations on these two testbeds. Particularly, the advantage of the SSL algorithm against the CORI formula was much more notable on the Trec4_kmeans testbed than on the Trec123_100Col testbed. It is known that the Trec4_kmeans testbed has more skewed word distribution than the Trec123_100Col testbed so that the information source statistics on the Trec4_kmeans testbed are more diverse than those on the Trec123_100Col testbed. This causes a serious problem for the CORI merging formula, which was also pointed out in previous research [Larkey, Connel & Callan, 2000]. The SSL algorithm addresses this problem by building different models for different selected information sources. This strategy helps to correct the skewed word distribution and the experiment results reflected this advantage of the SSL algorithm.

Second, on the Trec123_100Col testbed, both the SSL algorithm and the CORI formula tended to be more effective as more information sources were searched. In contrast, on the Trec4_kmeans testbed, although the accuracy of the SSL algorithm has been improved by selecting more information sources, the accuracy of the CORI algorithm deteriorated slightly. We attribute this to the fact that the contents of the information sources on the Trec4_kmeans testbed are much more homogenous than that on the Trec123_100Col testbed. Therefore, the relevant documents for a query are distributed in fewer information sources on the Trec4_kmeans testbed than on the Trec123_100Col testbed. Relatively more irrelevant documents from

Table 5.1: Precision at different document ranks using the CORI and semi-supervised learning approaches to merge retrieval results from INQUERY, language model and vector space search engines. 3 sources were selected to search for each query. Results are averaged over 50 queries.

Document Rank	**Trec123_100Col Testbed**		**Trec4_kmeans Testbed**	
	CORI Merge	**SSL Merge**	**CORI Merge**	**SSL Merge**
5	0.3240	0.3680 (+13.6%)	0.2440	0.3440 (+41.0%)
10	0.3260	0.3420 (+4.9%)	0.2320	0.2840 (+22.4%)
15	0.3147	0.3253 (+3.4%)	0.1987	0.2520 (+26.8%)
20	0.2930	0.3090 (+5.5%)	0.1820	0.2260 (+24.2%)
30	0.2627	0.2747 (+4.6%)	0.1573	0.1993 (+26.7%)

Table 5.2: Precision at different document ranks using the CORI and semi-supervised learning approaches to merge retrieval results from INQUERY, language model and vector space search engines. 5 sources were selected to search for each query. Results are averaged over 50 queries.

Document Rank	**Trec123_100Col Testbed**		**Trec4_kmeans Testbed**	
	CORI Merge	**SSL Merge**	**CORI Merge**	**SSL Merge**
5	0.3520	0.4040 (+14.8%)	0.2360	0.3720 (+57.6%)
10	0.3360	0.3700 (+10.1%)	0.1980	0.3160 (+59.6%)
15	0.3333	0.3627 (+8.8%)	0.1893	0.2853 (+50.7%)
20	0.3210	0.3470 (+8.1%)	0.1710	0.2520 (+47.4%)
30	0.3067	0.3233 (+5.4%)	0.1480	0.2133 (+44.1%)

Table 5.3: Precision at different document ranks using the CORI and semi-supervised learning approaches to merge retrieval results from INQUERY, language model and vector space search engines. 10 sources were selected to search for each query. Results are averaged over 50 queries.

Document Rank	**Trec123_100Col Testbed**		**Trec4_kmeans Testbed**	
	CORI Merge	**SSL Merge**	**CORI Merge**	**SSL Merge**
5	0.3520	0.4320 (+22.7%)	0.2360	0.3640 (+54.3%)
10	0.3500	0.4080 (+16.6%)	0.1860	0.3220 (+73.1%)
15	0.3453	0.4013 (+16.2%)	0.1733	0.2933 (+69.2%)
20	0.3390	0.3820 (+12.7%)	0.1630	0.2720 (+66.8%)
30	0.3287	0.3627 (+10.3%)	0.1460	0.2400 (+66.4%)

"bad" information sources were introduced by selecting more information sources on the Trec4_kmeans testbed. The CORI result merging formula suffered from those irrelevant documents while the SSL algorithm was still able to gain advantage by distinguishing irrelevant documents in the information sources of lower quality when more sources were selected.

Third, the advantage of the SSL algorithm over the CORI merging formula was generally larger at the top of the ranked list (i.e., 5 or 10) than at bottom (i.e., 15, 20 or 30). This is exactly more favorable according to the high-precision goal and is also more consistent with our strategy in the SSL algorithm to put more weight on the top returned documents (e.g., select top 10 overlap documents as training data for each information source).

The power of the SSL algorithm lies in the ability to build query-specific and source-specific transformation models and dynamically adjust parameters in the models. Another set of experiments was conducted to more carefully demonstrate the effectiveness of the SSL algorithm against the CORI results merging formula.

The CORI results merging formula in Equation 5.3 can be rewritten in another way as a linear function of the information source-specific document scores:

$$S_C(d_{ij}) = \frac{S'(d_{ij})(1+k*S'(db_i))}{1+k} \tag{5.9}$$

where k measures the importance of information source selection score, and it is set to 0.4 by default in the CORI results merging formula. The linear function only has the slope as $(1 + k*S(d_i)')/(1 + k)$ and has no intercept. Therefore, Equation 5.9 can be seen as a special case of the linear transformation model in the SSL algorithm.

One set of experiments was conducted to study the accuracy of the CORI merging formula by setting k to multiple values (when k is set to infinity, the CORI formula becomes $S_c\ (d_{ij}) = S\ (d_{ij})^{'} * S\ (d_i)^{'}$).

The results are shown in Figures 5.4 and 5.5. It can be seen from Figure 5.4 that the adjustment of parameter k did not make much difference to the effectiveness of the CORI merging algorithm on the Trec123_100Col testbed. On the Trec4_kmeans testbed, the CORI merge formula with an infinity value of k had a slight advantage over other values as shown in Figure 5.5, which can be explained by the fact that a much larger proportion of relevant documents are in the top few selected information sources on the Trec4_kmeans testbed than that on the Trec123_100Col testbed. Therefore, there was no particular choice of k that could improve the accuracy consistently in the experiments. Although the parameter k in the CORI merging formula was tuned in the experiments, essentially linear models with only one parameter of slope were applied to all of the selected information sources, and thus the linear transformation models across the information sources were associated with each other by the value k and the information source selection scores. In contrast, the SSL algorithm does allow each selected information source to choose its own query-specific and source-specific linear transformation model to achieve better accuracy. The experimental results in Figures 5.4 and 5.5 indicate that the SSL algorithm is more effective than all the variants of the CORI merging formula. Therefore, the experiments confirm that the power of the SSL algorithm derives from its ability to adjust the linear transformation models for different queries and information sources.

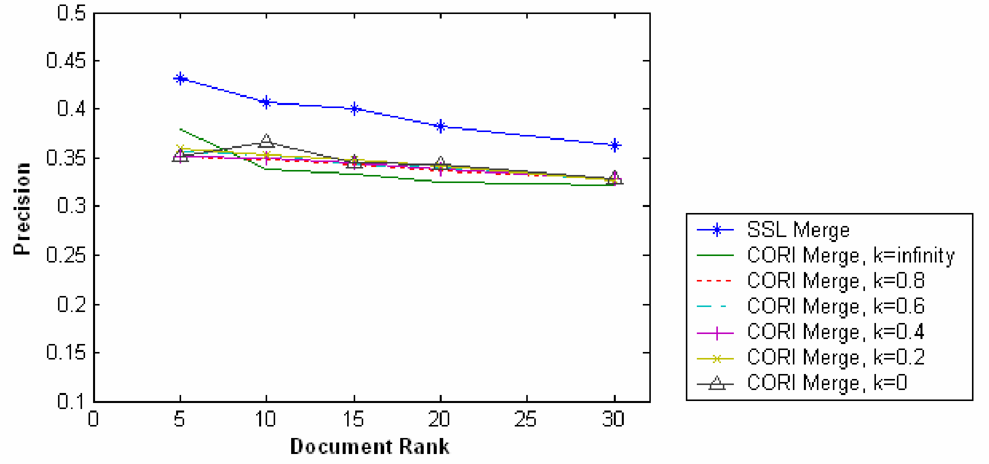

Figure 5.4: How varying the k parameter in the CORI result merging formula affects precision on the Trec123_100Col testbed.

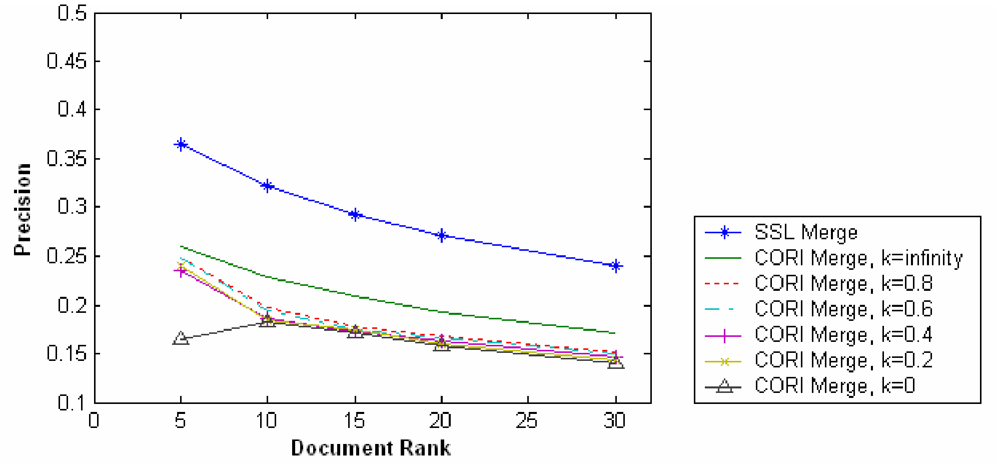

Figure 5.5: How varying the k parameter in the CORI result merging formula affects precision on the Trec4_kmeans testbed.

5.3.4 Experimental results: The effect of returned ranked list length

Experiments in Section 5.3.1 have shown that the SSL algorithm is very likely to have enough overlap documents when the information sources provide long ranked document lists. Let us assume that an information source provides a ranked list of 1,000 document ids and their scores by a single interaction with the central search client. If the corresponding communication cost can be estimated by 80 bytes per document (i.e., 60 bytes for document id and 20 bytes for document score) for each of the 1,000 documents, the total communication cost of this ranked list is about 80,000 bytes and in general the communication costs are not excessive.

Table 5.4: The percentage of selected information sources that are short of overlap documents for various lengths of ranked lists on the Trec123-100Col and the Trec4-kmeans testbeds. 10 information sources were selected per query. Results are averaged over 50 queries.

Results List Length	Percentage of Selected Information Sources with Fewer than 3 Overlap Documents	
	Trec123-100Col	Trec4-kmeans
50	54.2%	17.4%
100	27.8%	5.6%
200	10.4%	2.6%
500	6.0%	0.0%
1000	5.2%	0.0%

Table 5.5: The average number of downloaded documents to meet the requirement of at least 3 overlap documents per selected information sources on the Trec123-100Col and the Trec4-kmeans testbeds. 10 information sources were selected per query. Results are averaged over 50 queries.

Result List Length	Trec123-100Col		Trec4-kmeans testbed	
	Download Docs Per Source	Total Download Docs (10 Sources)	Download Docs Per Source	Total Download Docs (10 Sources)
50	1.0	10.2	0.3	3.3
100	0.4	4.4	0.1	1.1

However, in real world uncooperative environments of federated document retrieval systems, information sources may only provide short ranked list (e.g., top 10 or 20) for the initial search requests and additional interactions are required to obtain the results further down the list. If an information source returns a page of ranked list containing 20 document ids, it requires a total number of 50 interactions to obtain a ranked list of 1,000 documents. This is exactly our argument against the Capture-Recapture information source size estimation algorithm in Chapter 2, and we view it as excessive communication costs. Therefore, it is more desirable to rely on short ranked lists. However, one consequence of using short ranked lists is that the SSL algorithm is likely to have much fewer overlap documents for training data.

A set of experiments was conducted to study how many information sources are short of overlap documents for training data with various lengths of ranked document lists. The results are shown in Table 5.4. It can be seen from Table 5.4 that most information sources have enough overlap documents with long ranked lists (i.e., 500 or 1,000) on both these two testbeds, which is consistent with the experiments in Section 5.3.2. However, with short ranked lists (i.e., 50 or 100), the percentage of information sources without enough overlap documents grows dramatically. This problem is more serious when ranked lists of 50 or 100 documents are returned on the Trec123_100Col testbed, which contains more heterogeneous information sources.

An alternative approach is to download a minimum amount of returned document on the fly and intentionally create more overlap documents as the training data for information sources that do not have enough overlap documents. If the communication costs are measured by the number of interactions with information sources, this method may substantially reduce the cost without degrading the accuracy.

Specifically, when a selected information source does not provide enough overlap documents (i.e., fewer than 3), the rest of required overlap documents are downloaded on the fly and their centralized document scores are calculated with the corpus statistics of the centralized sample database to create the training data. Documents ranked at 1^{st}, 10^{th} and 20^{th} in the result lists are the candidates for downloading. These ranks are chosen as they cover a relatively wide range of the top ranked documents in the ranked lists, which are most important to meet the high-precision goal of federated document retrieval systems.

More experiments were conducted to study the amount of downloaded documents with short ranked lists of 50 or 100 documents. The summary statistics are shown in Table 5.5. In all these experiments, 10 information sources were selected to search. It can be seen that only 0.3-1.0 documents were required to download per information source on average when information sources returned ranked lists of 50 documents and only 0.1-0.4 documents were downloaded when information sources returned ranked lists of 100 documents. In both configurations, the number of interactions with a selected information source for downloading and acquiring a minimum number of overlap documents was much fewer than that for obtaining long ranked lists with 500 or 1,000 documents.

The above set of experiments demonstrates that it is efficient to download a minimum amount of overlap documents. Another set of experiments was conducted to show the effectiveness. The experiment results in Tables 5.6-5.8 show the accuracies of several methods of obtaining enough overlap documents for the SSL results merging algorithm. These methods either require long ranked lists (i.e., 1,000 documents), which is associated with a large amount of communication costs when only a small piece of ranked list can be provided by a single interaction, or download the minimum number of documents to create enough overlap documents with much shorter ranked lists (i.e., 50 or 100 documents). It can be seen that the effectiveness of these methods were about the same on the Trec123-100Col testbed. The method with long ranked lists had a small advantage over the minimum downloading method with ranked lists of 50 documents when 5 or 10 sources were selected on the Trec4-kmeans testbed where relatively more overlap documents could be obtained by the long ranked lists. Another possible reason is that the Trec4-Kmeans testbed contains

Table 5.6: Precision at different document ranks for three methods of obtaining enough overlap documents for the SSL results merging algorithm. Three types of search engines were used. 3 sources were selected to search for each query. Results are averaged over 50 queries.

Document Rank	Trec123-100Col			Trec4-Kmeans		
	List of 1,000	**List of 50 Docs + Downloads**	**List of 100 Docs + Downloads**	**List of 1,000 Docs**	**List of 50 Docs + Downloads**	**List of 100 Docs + Downloads**
5	0.3680	0.3640 (-1.1%)	0.3800 (+3.3%)	0.3440	0.3440 (0.0%)	0.3440 (0.0%)
10	0.3420	0.3360 (-1.8%)	0.3440 (+0.6%)	0.2840	0.2940 (+3.5%)	0.2980 (+4.9%)
15	0.3253	0.3253 (0.0%)	0.3320 (+2.1%)	0.2520	0.2547 (+1.1%)	0.2520 (0.0%)
20	0.3090	0.3140 (+1.6%)	0.3060 (-1.0%)	0.2260	0.2220 (-1.8%)	0.2270 (+0.4%)
30	0.2747	0.2780 (+1.2%)	0.2733 (-0.5%)	0.1993	0.1913 (-4.0%)	0.2000 (+0.4%)

Table 5.7: Precision at different document ranks for three methods of obtaining enough overlap documents for the SSL results merging algorithm. Three types of search engines were used. 5 sources were selected to search for each query. Results are averaged over 50 queries.

Document Rank	Trec123-100Col			Trec4-Kmeans		
	List of 1,000	**List of 50 Docs + Downloads**	**List of 100 Docs + Downloads**	**List of 1,000 Docs**	**List of 50 Docs + Downloads**	**List of 100 Docs + Downloads**
5	0.4040	0.4000 (-1.0%)	0.4120 (+2.0%)	0.3720	0.3440 (-7.5%)	0.3640 (-2.2%)
10	0.3700	0.3800 (+2.7%)	0.3920 (+6.0%)	0.3160	0.3040 (-3.8%)	0.3240 (+2.5%)
15	0.3627	0.3560 (-1.9%)	0.3653 (+0.7%)	0.2853	0.2560 (-10.2%)	0.2760 (-3.3%)
20	0.3470	0.3430 (-1.2%)	0.3470 (0.0%)	0.2520	0.2260 (-10.3%)	0.2440 (-3.2%)
30	0.3233	0.3240 (+0.2%)	0.3213 (-0.6%)	0.2133	0.1940 (-9.1%)	0.2073 (-2.8%)

Table 5.8: Precision at different document ranks for three methods of obtaining enough overlap documents for the SSL results merging algorithm. Three types of search engines were used. 10 sources were selected to search for each query. Results are averaged over 50 queries.

Document Rank	Trec123-100Col			Trec4-Kmeans		
	List of 1,000	**List of 50 Docs + Downloads**	**List of 100 Docs + Downloads**	**List of 1,000**	**List of 50 Docs + Downloads**	**List of 100 Docs + Downloads**
5	0.4320	0.4440 (+2.8%)	0.4360 (+0.9%)	0.3640	0.3320 (-8.8%)	0.3560 (-2.2%)
10	0.4080	0.4300 (+5.4%)	0.4280 (+4.9%)	0.3220	0.3000 (-6.8%)	0.3420 (+6.2%)
15	0.4013	0.4187 (+4.3%)	0.4133 (+3.0%)	0.2933	0.2600 (-11.3%)	0.2960 (+0.9%)
20	0.3820	0.3980 (+4.2%)	0.3900 (+2.1%)	0.2720	0.2420 (-11.0%)	0.2650 (-2.6%)
30	0.3627	0.3653 (+0.7%)	0.3707 (+2.2%)	0.2400	0.2113 (-11.9%)	0.2380 (-0.8%)

homogenous information sources and thus relevant documents are contained in a relatively small set of sources. Therefore, the choice of selecting more information sources (e.g., 5 or 10) introduces more irrelevant documents into the final document lists. All these factors require more training data to build more accurate models that can be used to distinguish those irrelevant documents. Still, the minimum downloading methods were much more effective than the CORI merging formula.

In our current setting, the centralized sample database does not change over time, but in an operational environment the centralized sample database may evolve over time. For example, the documents downloaded by the minimum downloading method can be accumulated in the centralized sample database as extra data for future queries. It can be imagined that when more and more downloaded documents have been accumulated in the centralized sample database, more training data is available for results merging and the requirement of downloading new documents can be reduced, especially for queries on popular topics. Furthermore, more training data may enable a gradual transition from simple transformation models to more complex ones. Therefore, the long-term learning of accumulated downloaded documents with the minimum downloading method of SSL algorithm gives us an opportunity to increase results merging accuracy and reduce communication costs for future queries.

To summarize, generally the accuracy of the minimum downloading method of the SSL algorithm is comparable to that of the SSL algorithm with long ranked lists. Since the minimum downloading method is more efficient, this method is more practical in environments where only short result lists are provided. Furthermore, it provides an opportunity to improve results merging accuracy and reduce communication costs in the long run by accumulating the downloaded documents as extra training data.

5.3.5 Evaluation methodology and experimental results with the FedLemur system

The goal of the FedLemur project was to build a prototype federated search system that connects to 20 government Web sites based on the Lemur toolkit. More detailed information about the Web sites can be seen in Section 2.1.

The minimum downloading method of acquiring enough overlap documents was not utilized in the FedLemur project. This was mainly an engineering problem due to the resource constraints imposed by the FedStats portal to search the selected information sources sequentially. Therefore, downloading documents on the fly could be very slow.

In order to acquire enough training data for user queries without using the minimum downloading method, the system used a variant of the query-based sampling method with a heuristic stopping criterion to acquire resource descriptions with more sampled documents than previous choices. Specifically, the system continued to sample documents from an information source until an upper limit is met (e.g., 2000 documents) or there were no new documents for a consecutive set of queries (e.g., 30 queries).

When the FedLemur system was developed, CORI was the only resource selection algorithm in the Lemur toolkit. Therefore, it was chosen in the FedLemur system. As the CORI resource selection algorithm is

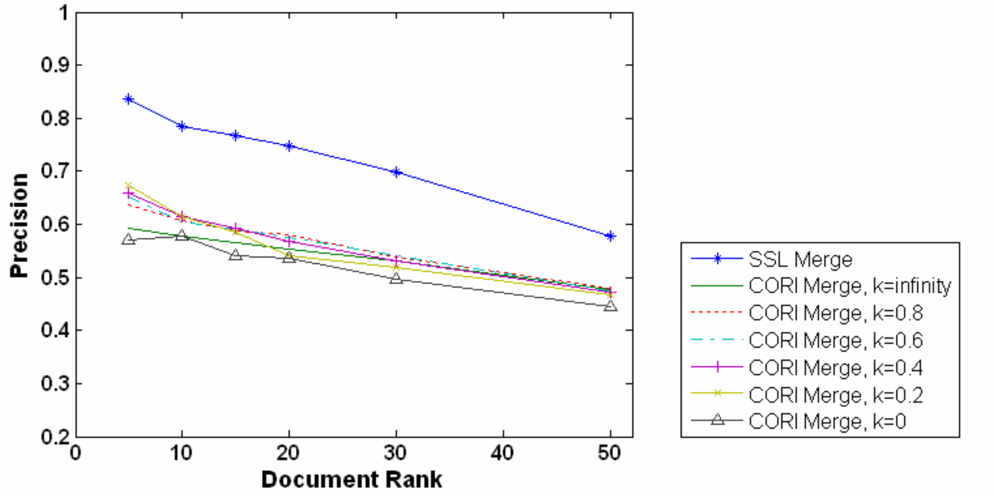

Figure 5.6: Accuracies of the SSL results merging algorithm and the CORI result merging formula by varying the K parameter in the FedLemur system.

compatible with both the CORI results merging formula and the semi-supervised learning algorithm, this provided a good opportunity to compare the accuracies of these two results merging algorithms in a real world setting.

The SSL results merging algorithm has been shown to be very effective in research environments when information sources return document scores with their ranked lists. However, most information sources within the FedLemur system return only document ranks. The system generated pseudo scores for top ranked documents from these sites. Specifically, the top document was assigned a score of 0.6, and the scores descended by even increments down to 0.4. This range was chosen because it is compatible with the CORI result merging formula. A pseudo score is not necessarily an accurate representation of a document's quality, but they are unavoidable when information sources do not return real document scores.

A set of experiments was conducted to study the effectiveness of the SSL algorithm and the CORI results formula with pseudo document scores. Specifically, 27 queries were used in the experiments. Top 3 or 5 information sources were selected for each query and each information source returned 35 or 50 documents. Members of the CMU and FedStats teams were invited to give the relevance judgments. More detailed information about the queries and the relevance judgment can be found in Section 2.1.

The experiment results in Figure 5.6 show the accuracies of a variety of the CORI results merging formula and the SSL results merging algorithm. The CORI merging formula was varied by setting different values of the parameter k in Equation 5.9. The SSL algorithm in this experiment utilized a centralized sample database that sampled up to 2000 documents from each information source. It can be seen from Figure 5.6 that the

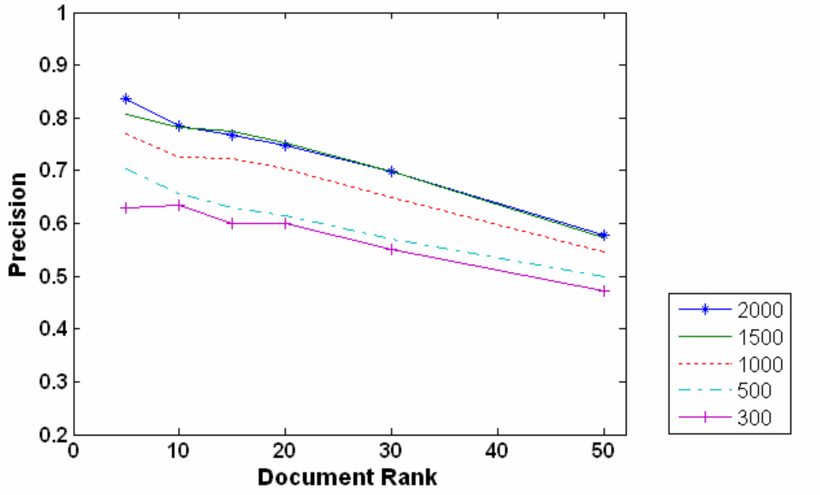

Figure 5.7: Accuracy of the SSL results merging algorithm by varying the sample size of query-based sampling in the FedLemur system.

Table 5.9: Number of queries that are unable to use SSL results merging algorithm at various sample sizes.

Sample Size Limit	**300**	**500**	**1000**	**1500**	**2000**
Queries back off to CORI merge	20	10	3	0	0

SSL result merging algorithm is better than the CORI result merging formula in all configurations. This is consistent with our observation in the research environments (i.e., experiment results in Section 5.3.3).

Another set of experiment was conducted to evaluate the effect of various sample database sizes. Specifically, the centralized sample database was created with a query-based sampling process that sampled up to 300, 500, 1000, 1500 and 2000 documents from each information source. Sampling fewer documents does not hurt the CORI resource selection algorithm, but it causes the back off of the SSL algorithm to the less effective CORI results merging formula, which reduces overall accuracy. Table 5.9 shows the number of queries that were unable to use SSL results merging algorithm when the sampling size limit varied. It can be seen that when query-based sampling could acquire 1000 documents from each information source, the SSL algorithm only needed to back off to the CORI results merging formula for about 10 percent (i.e., 3 out of 27) of all the queries.

The accuracy of the SSL results merging algorithm with centralized sample databases of various sizes is shown in Figure 5.7. It can be seen that when query-based sampling was allowed to acquire at most 1000 sampled documents, the overall accuracy improved substantially over the cases when fewer sampled documents were allowed. However, the cost/benefit ratio seems to peak around 1500 document limit while more sampled documents gave only marginal improvement of the accuracy.

5.4 Summary

The final step for a federated document retrieval system is results merging, which makes the document scores in different ranked lists comparable and merges them into a single list. This is a difficult task in uncooperative federated search environments as different information sources may use different retrieval algorithms and they use different corpus statistics to calculate document scores. Previous methods either directly calculate centralized comparable scores for all returned documents from selected information sources, which is associated with large communication or computation costs, or try to mimic the behavior of comparable document scores with heuristic methods, which is often not effective.

Based on this observation, this chapter proposes a Semi-Supervised Learning (SSL) results merging approach, which tries to approximate centralized comparable scores more effectively and efficiently. Following the theme of utilizing the centralized sample database in this dissertation (more comprehensive discussion can be found in Section 3.2), the SSL algorithm shows a successful example of using information within the centralized sample database to guide results merging. Specifically, the SSL algorithm models results merging as a task of transforming different sets of sources-specific document scores or pseudo document scores generated from rank information into a single set of source-independent comparable scores. It first applies a centralized retrieval algorithm on the centralized sample database to calculate comparable document scores for sampled documents. During results merging, it recognizes a set of overlap documents in both centralized sample database and individual ranked lists that have both centralized comparable scores and source-specific scores. This set of documents serves as training data to train query-specific and source-specific linear models that can be used to map source-specific document scores to comparable scores. Finally, the learned models are applied to all returned documents to estimate comparable document scores and the returned documents can be merged into a single final list.

This chapter addresses an important issue for the SSL result-merging algorithm to work in operational environments: the demand of obtaining enough training data from ranked lists may depend on receiving result list information about a large number of documents (e.g., 1,000) from each selected source. This chapter proposes an alternative method that works with short ranked lists by downloading a small number of documents on the fly from selected information sources. This method provides considerable freedom in how to trade off ranked list length and the number of documents downloaded on the fly.

Empirical studies in a wide range of federated search environments either within research environments or with the real world FedLemur prototype system have been conducted to demonstrate the effectiveness of the SSL algorithm. The SSL algorithm is shown to be consistently more accurate than several variants of the CORI results merging formula. Furthermore, the variant of the SSL algorithm that downloads a minimum number of documents on the fly is also studied to show its effectiveness and efficiency of obtaining enough training data with a small amount of communication costs.

Note that the SSL algorithm still uses a heuristic parameter for the back-off threshold (i.e., 40%) and the heuristic approach for adjusting a bias problem of the linear regression model (i.e., Figure 5.1). One topic for future research is to design an equally effective solution without these heuristics.

Chapter 6: Results Merging for Multilingual Ranked Lists

Most prior federated search algorithms work in environments where all documents are in the same language. However, in real world federated search environments, it is common that different information sources may contain documents in different languages. Multilingual federated search finds relevant documents in multiple languages for a query in one language (i.e., English in this dissertation). This chapter is focused on a key task of multilingual federated search: extending the monolingual results merging solutions from the previous chapter to merge multilingual ranked lists.

Our approach for results merging for multilingual federated search adopts a similar approach as the semi-supervised learning results merging algorithm proposed in Chapter 5 for monolingual federated search environments. Recall that the SSL algorithm requires an effective centralized monolingual retrieval algorithm to provide source-independent document scores on a centralized sample database. Similarly, it is important to design an effective multilingual centralized retrieval algorithm that provides source-independent document scores for comparing the documents in multilingual ranked lists. This algorithm is not the focus of this chapter. However, an effective multilingual centralized retrieval algorithm is briefly introduced in this chapter because it is crucial for obtaining an accurate results merging algorithm.

The multilingual results merging task is our primary interest. It is viewed in this chapter as results merging within uncooperative multilingual federated search environments. Particularly, with retrieved multilingual ranked lists, the comparable document scores for a small number of documents are calculated by the proposed multilingual centralized retrieval algorithm in this chapter. These documents have both centralized comparable scores and source-specific scores and they serve as the training data to estimate query-specific and language-specific transformation models. Then the learned transformation models are applied to estimate comparable scores for all retrieved documents and thus the documents can be sorted into a final ranked list.

The Cross-Language Evaluation Forum (CLEF) provides a good opportunity to develop and test both the multilingual centralized retrieval algorithm and the results merging algorithm for multilingual federated search. Particularly, the systems developed by the algorithms proposed in this chapter were evaluated in two multilingual retrieval tasks at CLEF 2005: Multi-8 two-years-on retrieval and Multi-8 results merging. The first task is a multilingual centralized retrieval task and the second task is a results merging task in multilingual federated search environments. An extensive set of experiments demonstrates the advantage of

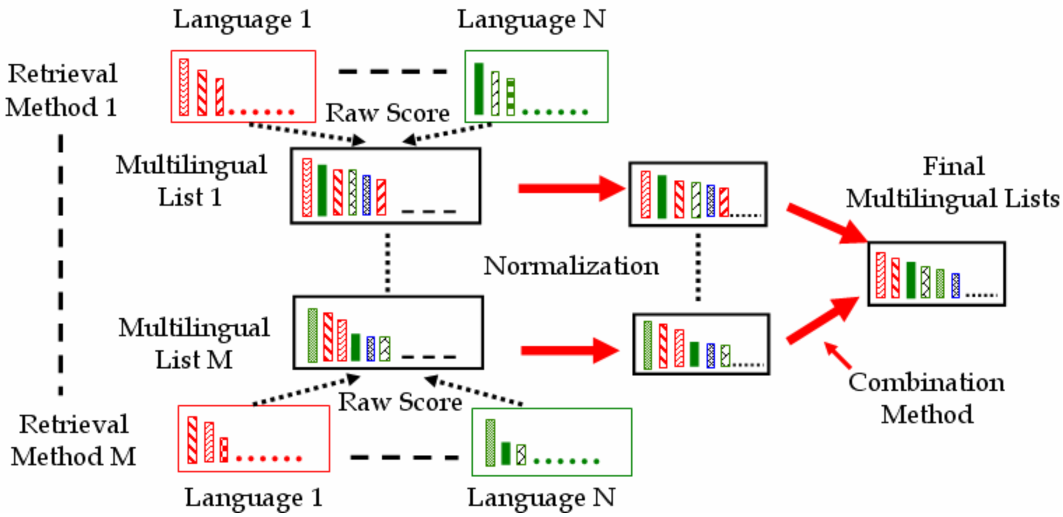

Figure 6.1: The approach of multilingual centralized retrieval method combines all results from a specific method into a multilingual result, and then combines the results from all methods into a final list.

the new methods against several alternatives.

The next section briefly describes our multilingual centralized retrieval algorithm. Section 6.2 proposes new results merging algorithms for multilingual federated search task. Section 6.3 explains the experimental methodology and presents the experimental results.

6.1 Multilingual centralized retrieval

Multilingual centralized retrieval is not the goal of this dissertation. However, an effective multilingual centralized retrieval solution is very important to obtain high merging accuracy of multilingual results merging algorithms. This section briefly presents our multilingual centralized retrieval algorithm. It utilizes different retrieval methods with different translation methods or different retrieval configurations.

Most previous retrieval methods [Kamps et al., 2003; Savoy, 2003] first generate accurate language-specific results by merging results from different types of retrieval and translation methods (meta search) and then merge the language-specific results together (results merging). However, it is not easy to merge these language-specific results because the ranges and distributions of the document scores within the language-specific lists can be very diverse as different retrieval methods are tuned to generate accurate language-specific results separately.

An alternative approach generates results across languages by the same retrieval algorithm with the same configuration and then merges all the disjoint ranked lists from the particular method into a simple multilingual ranked list (results merging) [Chen & Gey, 2003]. Similarly, many simple multilingual results can be obtained by applying different retrieval algorithms. These ranked lists contain the same set of documents but have different rankings. Finally, those simple multilingual ranked lists can be merged into a more accurate multilingual ranked list (meta search). Figure 6.1 shows a more detailed representation of this multilingual centralized retrieval method. This approach has been shown to be more effective than the approach described in the previous paragraph [Chen & Gey, 2003], and it is utilized in this dissertation.

Section 6.1.1 briefly discusses one translation tool. Section 6.1.2 presents two multilingual retrieval algorithms based on query translation and document translation respectively, then Section 6.1.3 proposes several methods to combine the results from multiple multilingual retrieval algorithms.

6.1.1. Translation matrices by learning from parallel corpus

Translation tools have been widely used by multilingual retrieval algorithms to cross the language barriers. The translation process in this work is mainly accomplished in a word-by-word manner by using translation matrices generated by parallel corpus as the European Parliament proceedings[17], which contains aligned sentences in multiple languages. Furthermore, the GIZA++ [Och & Hermann, 2000] tool is utilized to construct the translation matrices from English words to the words of the other languages or from words in other languages to English words. A probability value is assigned to each translation pair, indicating how probable the translation is.

6.1.2. Multilingual retrieval via query translation or document translation

One straightforward multilingual retrieval method is to translate the queries into different languages, search using translated queries to acquire language-specific results, and merge these results into a single multilingual ranked list.

Specifically, each English query word is translated into the top three candidates in the translation matrices of other languages. All three translated words are associated with normalized weights (i.e., the sum of the weights is 1) proportional to the weights in translation matrices. As the vocabulary of the parallel corpus is limited, we also utilize word-by-word translation results from the online machine translation software Systran[18] as a complement. A weight of 0.2 is assigned to the translation by Systran and a weight of 0.8 is

[17] http://people.csail.mit.edu/koehn/publications/europarl

[18] http://www.systransoft.com/index.html

Table 6.1: Several retrieval methods used for multilingual centralized retrieval.

Methods	Translation Method	Pseudo Relevance-Feedback
Qry_nofb	Query	N
Qry_fb	Query	Y
Doc_nofb	Document	N
Doc_fb	Document	Y

assigned to the translation with parallel corpora. The translated queries are used to search the index built for each language. Each query term weight is its weight in the translation representation. The Okapi retrieval algorithm is applied to do the retrieval. Okapi is chosen here as it has been widely used for multilingual retrieval [Lam-Adesina & Jones, 2003; Kamps et al., 2003]. However, we also believe other popular algorithms like INQUERY should work. As the same retrieval algorithm is applied on the corpus of different languages with original/translated queries of the same lengths (3 translations per query term), the raw scores in the ranked lists are somewhat comparable. Therefore, these ranked lists are merged together by their language-specific scores into a final ranked list. This retrieval method is formally denoted as Qry_nofb as shown in Table 6.1.

Another multilingual retrieval algorithm based on query translation, which utilizes query expansion by pseudo relevance feedback, is also applied by adding 10 most common query terms within top 10 ranked documents of the initial retrieval result for each language and then doing the search and merging again. This retrieval method is denoted as Qry_fb.

An alternative multilingual retrieval method is to translate all documents in other languages into English and apply the original English queries. The retrieval method based on document translation may have an advantage compared to the retrieval method based on query translation as the translation of long documents may be more accurate to represent the semantic meaning than the translation of short queries. For example, previous research [Chen & Gey, 2003] has shown an example that although one English query term is not correctly translated into the corresponding Spanish word by query translation, this Spanish word may still be correctly translated into the English query term by document translation. Translation matrices built from parallel corpus are utilized for translating documents. For each word in a language other than English, its top three English translations are considered. Five word slots are allocated to the three candidates with weight proportion to their normalized translation probabilities. Then all of the translated documents as well as the original English documents are collected into a single monolingual database and indexed. Furthermore, the original English queries are used to search the monolingual database with the Okapi retrieval algorithm without using pseudo relevance feedback. This method is formally denoted as Doc_nofb as shown in Table 6.1.

Similarly, another variant of the Okapi retrieval algorithm using document translation is applied with query expansion by pseudo relevance feedback (i.e., 10 additional query terms from top 10 ranked documents). This method is represented as Doc_fb.

In summary, two types of multilingual retrieval methods based on query expansion (i.e., with or without query expansion) and two types of multilingual retrieval methods based on document expansion (i.e., with or without query expansion) are utilized in this chapter. These retrieval methods have been summarized in Table 6.1.

6.1.3. Combining multilingual ranked lists

One simple combination algorithm is proposed to favor documents returned by more retrieval methods as well as high ranking documents returned by single types of retrieval methods. Let $S_{rm}(d_{k\_j})$ denote the raw document score for the jth document retrieved from the mth multilingual ranked list for the kth query, $S_{rm}(d_{k\_max})$ and $S_{rm}(d_{k\_min})$ represent the maximum and minimum document scores in this ranked list respectively. The normalized score of the jth document is calculated as:

$$S_m(d_{k\_j})=\frac{(S_{rm}(d_{k\_j})\text{-}S_{rm}(d_{k\_min}))}{(S_{rm}(d_{k\_max})\text{-}S_{rm}(d_{k\_min}))} \qquad (6.1)$$

where $S_m(d_{k\_j})$ is the normalized document score. Then the normalized document scores among all ranked lists are summed up for each individual document and all the documents can be ranked accordingly. This method is called the *equal weight combination method* in this work, which can be seen as a variant of the well-known CombSum [Lee, 1997] algorithm for meta search.

The equal weight combination method treats the votes from multiple retrieval methods with equal weight. One more sophisticated idea is to learn the weights indicating the effectiveness of each retrieval method. Formally, for M ranked lists to combine, the final combined document scores for a specific document d is calculated as:

$$S_{final}(d)=\frac{1}{M}\sum_{m=1}^{M} w_m S_m(d)^{r_m} \qquad (6.2)$$

where $S_{final}(d)$ is the final combined document score and $S_m(d)$ (which is zero if the document is not in the mth ranked list) represents the normalized score for this document from the mth ranked list.

$\overrightarrow{w}=\{w_1,...,w_M\}$ and $\vec{r}=\{r_1,...,r_M\}$ are the model parameters, where the pair of (w_m , r_m) represents the weight of the vote and the exponential normalization factor for the mth ranked list respectively.

Our criterion of estimating desired model parameters is to maximize the ranking accuracy. In this work, the ranking accuracy is represented formally as the *mean average precision* (MAP) criterion, which is used by the multilingual retrieval task of CLEF as well as many other evaluation tasks*. Let us assume that there are K training queries, then the value of MAP is calculated as follows:

$$\frac{1}{K}\sum_{k}\sum_{j\in D_k^+}\frac{\text{rank}_k^+(j)}{j} \tag{6.3}$$

where D_k^+ is the set of the ranks of relevant documents in the final ranked list for the kth training query, and $\text{rank}_k^+(j)$ is the corresponding rank only among relevant documents. To avoid the overfitting problem of model parameter estimation, two regularization items are introduced for $\overrightarrow{w}$ and $\vec{r}$ respectively. Together with the ranking accuracy criterion in Equation 6.3, the training problem is represented as follows:

$$(\overrightarrow{w},\vec{r})^*=\underset{\overrightarrow{w},\vec{r}}{\text{argmax}}\,(\log\left(\frac{1}{K}\sum_{k}\sum_{j\in D_k^+}\frac{\text{rank}_k^+(j)}{j}\right)-\sum_{m=1}^{M}\frac{(w_m-1)^2}{2*a}-\sum_{m=1}^{M}\frac{(r_m-1)^2}{2*b}) \tag{6.4}$$

where $(\overrightarrow{w},\vec{r})^*$ is the estimated model parameters and (a , b) are two regularization factors that are empirically set to 4 in this work. As this problem is not a convex optimization problem and there exist multiple local maximal values, a common solution is used in this work to search with multiple initial points. Finally, the model with desired parameters can be applied on test queries for combining ranked lists of different retrieval systems. This method is called the *learning combination method* in this work. The detailed experimental methodology (e.g., use of training data) is shown in Section 6.3.

This section presents several multilingual centralized retrieval algorithms. They are not the focus of this chapter but an effective multilingual retrieval solution is very important to compare document scores in multilingual ranked lists for accurate results merging algorithms. This section presents multilingual centralized retrieval algorithms to first combine all results from a particular retrieval algorithm into a multilingual ranked list, and then combine the results from all retrieval methods into a final multilingual list.

* Note that this is different from the statistical concept of maximum a posterior, although the two concepts have the same acronym representation.

6.2 Results merging for multilingual federated search

Merging ranked lists in multiple languages is an important task for multilingual federated search. This is the primary focus of this chapter. In this section, the results merging algorithms are proposed to work in uncooperative multilingual federated search environments. The documents within the information sources can only be accessed through their source-specific searching services while different sources may use different retrieval algorithms. It is assumed in this section that each source is monolingual in a different language. The methods can be extended to multiple sources per language with no significant changes.

Previous research [Savoy, 2002; Savoy, 2003] proposed the solution of learning transformation models from the relevance judgments of queries in the past to map language-specific document scores into the probabilities of relevance and thus the retrieved documents across different languages can be ranked by their estimated probabilities of relevance. However, the same model is applied for different queries within a single language, which may be problematic as the document scores across different queries can be quite different. An alternative approach [Martinez-Santiago et al., 2002; Rogati & Yang, 2003] is to translate and index all returned documents across different languages for each query and apply a centralized retrieval algorithm to compute comparable document scores. Although this method is accurate, it is often associated with a large amount of computation costs and communication costs in federated search environments.

This section proposes a new approach to learn query-specific and language-specific models of transforming language-specific document scores into comparable document scores. In particular, a small set of returned documents from each language is downloaded, translated and indexed at retrieval time to compute comparable document scores, and then the query-specific and language-specific models are trained by both comparable document scores and language-specific document scores for these small sets of documents. The models are applied to the ranked lists of all languages to obtain comparable document scores and finally of all the returned documents are merged into a single list using their comparable scores. This approach is a variant of the semi-supervised learning results merging method proposed in Chapter 5. It uses automatically computed document scores (by a multilingual centralized retrieval method from Section 6.1) as the surrogates of relevance judgment data.

This section is organized as follows: Section 6.2.1 describes the approach of learning query-independent and language-specific transformation merging models and proposes an extended method of learning the model by maximizing the mean average precision criterion; and Section 6.2.2 proposes the new approach of learning query-specific and language-specific result merging algorithm.

6.2.1 Learning query-independent and language-specific merging models with training data

To make the retrieved results from different ranked lists comparable, one natural idea is to map all of the document scores into the probabilities of relevance and rank the documents accordingly. A logistic transformation model has been utilized in previous studies to achieve this goal [Savoy, 2002; Savoy, 2003]. This method has been shown to be more effective than several other alternatives such as the round robin results merging method and the raw score results merging method [Savoy, 2002; Savoy, 2003]. Let us assume that there are altogether I ranked lists from different languages to merge, each of them provides J documents for each query and there are altogether K training queries with human relevance judgments. Particularly, d_{k_ij} represents the j^{th} document within the ranked list in the i^{th} language of training query k. The pair (r_{k_ij}, $S_i(d_{k\_j})$) represents the rank of this document and the document score (normalized by Equation 6.1) respectively. Then the probability of relevance of this document can be estimated using the logistic transformation model as:

$$P(rel|d_{k\_ij})=\frac{1}{1+\exp(a_i r_{k\_ij}+b_i S_i(d_{k\_j})+c_i)} \tag{6.5}$$

where a_i ,b_i and c_i are the parameters of the language-specific model that transforms all document scores from the i^{th} language into the corresponding probabilities of relevance. Note that the same model is applied for all documents retrieved for different queries, which indicates that the model is query-independent. The optimal model parameter values are acquired generally by maximizing the log-likelihood (MLE) of training data [Savoy, 2002] as follows:

$$\sum_{k,i,j} P^*(rel|d_{k\_ij})\log(P(rel|d_{k\_ij})) \tag{6.6}$$

where $P^*(rel|d_{k\_ij})$ is the empirical probability value of a particular document. This is derived from human relevance judgment data of training queries, which is 1 when the document is relevant and 0 otherwise. In contrast to the optimization problem in Equation 6.4, this objective function is convex, which guarantees the existence of a single global optimal solution.

This method treats each relevant document equally. However, this may not be a desired criterion in real world applications. For example, query A has two relevant documents and query B has one hundred relevant documents. A relevant document for query A is generally more important to users than a relevant document for query B. Therefore, if all the queries are of equal importance to us, a more reasonable criterion for information retrieval evaluation is to treat all the queries equally instead of the individual relevant documents. The *mean average precision* (MAP) criterion described in Equation 6.3 reflects this criterion. Another important reason to use MAP is that the two multilingual retrieval tasks of CLEF use MAP as the metric for the competition. Using MAP enables us to directly optimize the performance of the algorithms in this chapter for the CLEF evaluation. Generally speaking, the model can be optimized for many different metrics.

Based on the above observation of prior research, this dissertation proposes a natural extension of training the logistic transformation model by the MAP criterion. Different sets of model parameters $\{a_i\ ,b_i$ and $c_{i,}\ 1<=i<=I\}$ generate different sets of relevant documents for all K training queries as $\{D_k^+, 1 <= k <= K\}$ and thus achieve different MAP values. The training goal is to find a set of model parameters that generates the highest MAP value. However, this problem is not a convex optimization problem and there exist multiple local maximal values. In this work, a common solution is utilized to search with multiple initial points. This new algorithm of training logistic model for maximum mean average precision is called logistic model with the MAP goal, while the previous algorithm [Savoy, 2002; Savoy, 2003] trained for maximum likelihood is called logistic model with the MLE goal.

6.2.2. Learning query-specific and language-specific merging models

Savoy's query-independent and language-specific logistic transform model (Section 6.2.1) applies the same model on the results of different queries for each language. This is problematic when the ranked lists of different queries have similar score distributions but different distributions of probabilities of relevance. This suggests that a query-specific model should be studied to improve the results merging accuracy.

One query-specific solution is to download and translate all returned documents from different languages at the retrieval time and compute comparable document scores to merge them together [Martinez-Santiago et al., 2002; Rogati & Yang, 2003]. This results merging method downloads (also indexes and translates) all returned documents; it is called the *complete downloading method.*

Our implementation of the complete downloading method downloads all returned documents at the retrieval time and applies a variant of the multilingual centralized retrieval method proposed in Section 6.1 to compute comparable document scores. Particularly, after downloading the documents, queries are translated into different languages and the Okapi retrieval algorithm is applied to obtain language-specific document scores. The returned documents are merged by the raw scores into a multilingual ranked list. Then all downloaded documents are translated into English and indexed by the method described in Section 6.1. The original English queries are applied to the translated documents by the Okapi retrieval algorithm to obtain document scores based on the document translation method, and then all downloaded documents are merged by the raw scores into another multilingual ranked list. Finally, these two multilingual ranked lists are combined by the method introduced in Section 6.1 into a final ranked list.

The complete downloading method is effective. However, this method is associated with the large amount of communication costs of downloading the documents and computation costs of translating and indexing many documents.

In this section, a more efficient results merging algorithm is proposed to work in the multilingual federated search environments. This approach is a variant of the semi-supervised learning results merging method proposed in Chapter 5. It only downloads and calculates comparable document scores for a small set of returned documents and trains query-specific and language-specific models, which transform language-specific document scores to comparable scores for all returned documents.

Particularly, only top ranked documents in the ranked list of each information source are selected for downloading and calculating comparable document scores. Let us assume that the top L documents in the ranked list of each information source are downloaded. Let the pair ($S_c(d_{k'\_l})$, $S_i(d_{k'\_l})$) denote the normalized comparable document score and the normalized language-specific score for the l^{th} downloaded document of the i^{th} information source for the query k'. Let the pair ($a_{k'\_i}$, $b_{k'\_i}$) denote the parameters of the corresponding query-specific and language-specific model. These parameters are learned by solving the following optimization problem to minimize the mean squared error between exact normalized comparable scores and the estimated comparable scores as:

$$\left(a^*_{k'\_i}, b^*_{k'\_i}\right) = \underset{(a,b)}{\operatorname{argmin}} \sum_{d_{k'\_l} \in D_L \cup D_{NL}} (S_c(d_{k'\_l}) - \frac{1}{1+\exp(a*S_i(d_{k'\_l})+b)})^2 \qquad (6.7)$$

where D_L is the set of L downloaded documents from the source. D_{NL} is a pseudo set of L documents that have no document representations but have pseudo normalized comparable scores of zero and pseudo normalized language-specific scores of zero. This set of pseudo documents is introduced in order to make sure that the learned model ranks documents in the correct way (i.e., documents with large language-specific scores are assigned with large comparable document scores and thus rank higher in the final ranked list).

Finally, logistic transformation models are learned for all information sources in a similar way. These models are applied to all returned documents from all sources and thus the returned documents can be ranked according to their estimated comparable scores. Note that only language-specific document scores are used in the logistic model in Equation 6.7 while document ranks in language-specific ranked lists are not considered. This choice is different from previous research (Equation 6.5), and is used here to avoid the overfitting problem for the limited amount of training data (i.e., the training data of the query-specific model in Equation 6.7 is less than that of the query-independent model in Equation 6.5). Exact comparable document scores are available for all of the downloaded documents. One method to take advantage of the evidence is to combine them with the estimated comparable scores. In this work, they are combined together with equal weights (i.e., 0.5). This approach has been found to generate slightly better empirical results.

The query-specific and language-specific results merging algorithm proposed in this section is different from the methods proposed in Chapter 5 in several perspectives. First, a logistic model instead of a linear model is utilized here to estimate comparable document scores. Logistic models have been commonly used in previous work [Savoy, 2002; Savoy, 2003] for merging multilingual documents and they are combined with the semi-supervised learning method in this chapter to further improve the state-of-the-art and to compare with previous research.

Second, the results merging algorithms in Chapter 5 heavily utilize overlap documents that exist in both centralized sample database and individual ranked lists for creating training data. The results merging algorithms in Chapter 5 are more probable to obtain more overlap documents in centralized sample database due to resource selection procedure. As an information source is selected only if the language model derived from its sampled documents matches the query, some of these sampled documents are likely to be retrieved from this information source and it is more probable to find more overlap documents from centralized sample database. However, the results merging algorithms in this section were developed for the CLEF results merging task, which searches all available sources without resource selection. Therefore, the help from overlap documents is substantially less and the algorithm in this chapter downloads some documents to create training data.

6.3 Evaluation methodology and experimental results

This section first describes the experimental methodology of the two CLEF tasks: the multilingual centralized retrieval task and the multilingual results merging task. Next, it presents experimental results to demonstrate the power of the algorithms proposed in this chapter.

6.3.1 Experimental methodology

The Cross-Language Evaluation Forum (CLEF) provides a good opportunity to evaluate both centralized multilingual retrieval algorithms and results merging algorithms for multilingual federated search. We participated in two tasks of CLEF 2005: Multi-8 two-years-on multilingual retrieval and Multi-8 results merging.

The first task of Multi-8 two-years-on is a multilingual retrieval task, which is to search documents in eight languages with queries in a single language (i.e. English) in a centralized environment where we have full access to all the documents. 20 training queries with relevance judgments are provided to tune the behavior of multilingual retrieval algorithms and the algorithms are evaluated on 40 test queries. More detailed information about the eight information sources and the queries can be found in Chapter 2.

The second task of Multi-8 results merging task is to merge ranked lists of eight different languages into a single final list. This is viewed in this chapter as a results merging task within uncooperative multilingual federated search environments. There are eight information sources that contain documents in eight different languages. The documents can be only accessed through source-specific search engines (assigned by the results merging task of CLEF 2005). In the offline phase, query-based sampling is used to acquire 3,000 sampled documents from each information source. A relatively large number of sampled documents are acquired from each source to generate more accurate language-specific corpus statistics for the multilingual centralized retrieval algorithm. This is different from the case in monolingual environments where all the sampled documents from multiple sources in a single language can be collapsed together for generating corpus statistics. For each user query, eight ranked lists are generated by the search engines of these information sources (one per language). The Multi-8 results merging task uses the same set of documents and queries (i.e., training and test) as the Multi-8 two-years-on task.

Some basic text preprocessing techniques have been utilized to process multilingual documents in both of the two tasks are:

- Stopword lists: The INQUERY [Turtle 1990; Callan, Croft & Harding, 1992] stopword list was used for English documents. Stopword lists of Finnish, French, German, Italian, Spanish and Swedish were acquired from[19], while the snowball stopword[20] list was used for Dutch;
- Stemming: Porter stemmer was used for English words. Dutch stemming algorithm was acquired from[20] and the stemming algorithms from[21] were used for the other six languages;
- Decompounding: Dutch, Finnish, German and Swedish are compound rich languages. The same set of decompounding procedures as described in previous research [Kamps, et al., 2003] was utilized.

[19] http://www.unine.ch/info/clef/

[20] http://www.snowball.tartarus.org/

[21] http://people.csail.mit.edu/koehn/publications/europarl

Table 6.2: Mean average precision of multilingual retrieval methods. Qry means by query translation. Doc means by document translation, nofb means no pseudo relevance feedback, fb means pseudo relevant back. UniNE is the results from the UniNE system [Savoy, 2003]

Methods	**Train**	**Test**	**All**
Qry_fb	0.317	0.353	0.341
Doc_nofb	0.346	0.360	0.356
Qry_nofb	0.312	0.335	0.327
Doc_fb	0.327	0.332	0.330
UniNe	0.322	0.330	0.327

Table 6.3: Mean average precision of merged multilingual lists of different methods. M_X means to combine X results in the order of: 1) query translation with feedback; 2) document translation without feedback; 3) query translation without query expansion; 4) document translation with query expansion; and 5) UniNE system. (For example, M2 means Qry_fb plus Doc_nofb). W1: means combine with equal weight, Trn means combine with trained weights.

Methods	**Train**	**Test**	**All**
M2_W1	0.384	0.431	0.416
M2_Trn	0.389	0.434	0.419
M3_W1	0.373	0.423	0.406
M3_Trn	0.383	0.431	0.415
M4_W1	0.382	0.432	0.415
M4_Trn	0.389	0.434	0.419
M5_W1	0.401	0.446	0.431
M5_Trn	0.421	0.449	0.440

6.3.2 Experimental results: Multilingual centralized retrieval

Multilingual centralized retrieval is not the focus of this chapter. However, the proposed federated multilingual results merging algorithms need an effective multilingual centralized retrieval algorithm to generate training data automatically. Therefore, the accuracy of the proposed multilingual centralized retrieval algorithm is briefly presented here.

The proposed multilingual centralized retrieval algorithm in this dissertation combines the results from a specific retrieval method into a multilingual result list, and then combines all the multilingual results from different methods into a single list. It is helpful to first investigate the accuracy of individual multilingual ranked lists from different retrieval methods. Table 6.2 shows the accuracies of five multilingual retrieval algorithms on training queries (first 20 queries), test queries (next 40 queries) and the overall accuracies. It can be seen that these methods produced results of similar accuracies, while the retrieval method based on document translation that does not use query expansion has a small advantage. The merged multilingual results from the UniNE system [Savoy, 2003] (i.e., eight ranked lists of different languages merged together by logistic transformation models trained by maximizing the MAP criterion) are also shown in Table 6.2 as it is utilized in this work. It can be seen that the accuracy of the UniNE system was very close to the other four algorithms.

Table 6.4: Language-specific retrieval accuracy in mean average precision of results from UniNE system.

Language	Dutch	English	Finnish	French	German	Italian	Spanish	Swedish
All (MAP)	0.431	0.536	0.192	0.491	0.513	0.486	0.483	0.435

Table 6.5: Language-specific retrieval accuracy in mean average precision of results from HummingBird system

Language	Dutch	English	Finnish	French	German	Italian	Spanish	Swedish
All (MAP)	0.236	0.514	0.163	0.350	0.263	0.325	0.298	0.269

The key idea of the multilingual centralized retrieval algorithm in this chapter is to improve the accuracy of multilingual retrieval result by combining results of several multilingual retrieval methods. Two combination methods described in Section 6.1 as the equal weight combination method and the learning combination method were applied. They were used to combine the results of the five retrieval algorithms described above. The combination results are shown in Table 6.3. It can be seen that the combination methods improved the accuracy of individual multilingual retrieval results shown in Table 6.2. Careful analysis shows that although the training combination method was consistently better than the equal weight combination method for the same set of configurations (i.e., the same number of ranked lists to combine), its advantage was very small. One possible explanation is that the accuracies of the five retrieval algorithms (i.e., shown in Table 6.2) were very close and adjusting the voting weights could not make much difference.

6.3.3 Experimental results: Results merging for multilingual federated search

This section presents experimental results of multilingual results merging algorithms in federated search environments. Two sets of language-specific ranked lists (i.e., lists of eight languages) from the UniNE system [Savoy, 2003] and the HummingBird system[22] were provided for each query from the results merging task of CLEF 2005. The results merging algorithms were required to merge each set of ranked lists into a single list and were evaluated by the accuracies of the final merged lists.

The language-specific retrieval accuracies of ranked lists of UniNE and HummingBird systems are shown in Table 6.4 and Table 6.5 respectively. It can be seen from Table 6.4 that the UniNE system generates accurate language-specific ranked lists for all languages expect for Finnish. In contrast, the accuracies of the ranked lists generated by HummingBird system shown in Table 6.5 are much lower than those of the UniNE system. These two sets of ranked lists are good candidates to evaluate result merging algorithms with both accurate language-specific ranked lists and inaccurate language-specific ranked lists.

[22] http://www.hummingbird.com/products/searchserver/

Table 6.6: Mean average precision of merged multilingual lists of different methods on UniNE result lists. TrainLog_MLE means trained logistic transformation model by maximizing MLE. TrainLog_MAP means trained logistic transformation model by maximizing MAP.

Methods	**Train**	**Test**	**All**
TrainLog_MLE	0.301	0.301	0.301
TrainLog_MAP	0.322	0.330	0.327

Table 6.7: Mean average precision of merged multilingual lists of different methods on HummingBird result lists. TrainLog_MLE means trained logistic transformation model by maximizing MLE. TrainLog_MAP means trained logistic transformation model by maximizing MAP.

Methods	**Train**	**Test**	**All**
TrainLog_MLE	0.186	0.171	0.176
TrainLog_MAP	0.210	0.192	0.198

The first two sets of experiments were conducted to evaluate two *query-independent* and language-specific results merging algorithms by optimizing the maximum likelihood criterion (MLE) and the mean average precision (MAP) criterion respectively. Their accuracies on the ranked lists of the UniNE system and the HummingBird system are shown in Table 6.6 and Table 6.7. It can be seen that the accuracies of the merged results of the UniNE system was much higher than those of the HummingBird system. This is consistent with our expectation as the language-specific ranked lists of the UniNE system are more accurate than those of the HummingBird system. Furthermore, it can be seen from both Tables 6.6 and 6.7 that the learning algorithm optimized for the mean average precision criterion was always more accurate than that optimized for the maximum likelihood criterion (~10%). This demonstrates the power of the method to directly optimize for mean average precision by treating different queries equally versus the method of optimizing for maximum likelihood.

Our key idea of improving the merging accuracy is to introduce *query-specific* and language-specific results merging algorithms. Two types of algorithms were evaluated in this work. The first method (i.e., complete downloading method) downloaded all documents from ranked lists of different languages and calculated comparable document scores (C_X). The second method only downloaded a small set of top ranked documents and calculated their comparable document scores to build logistic transformation models. Then these models estimated comparable document scores for all retrieved documents and the estimated scores were combined with exact comparable scores wherever they were available (Top_X_C05). Note that both of these two algorithms do not require human relevance judgments for training data. Therefore, the results on the training query set and the results on the test query set were obtained independently and it is not necessary that the results on training queries are better than the results on test queries.

Table 6.8: Mean average precision of merged multilingual lists of different methods on UniNE result lists. Top_x indicates x top documents are downloaded to generate logistic transformation model, C05 indicates both scores from logistic transformation model and centralized document scores are utilized when they are available and they are combined with a linear weight as 0.5. Top_[1,10,20] indicates to download top 1, 10 and 20 documents. C_X means top X documents from each list are merged by their centralized doc scores.

Methods	Train	Test	All
Top_150_C05	0.360	0.412	0.395
Top_30_C05	0.357	0.399	0.385
Top_15_C05	0.346	0.402	0.383
Top_10_C05	0.330	0.393	0.372
Top_5_C05	0.296	0.372	0.347
Top_[1,10,20]_C05	0.298	0.352	0.334
C_1000	0.356	0.382	0.373
C_500	0.356	0.384	0.374
C_150	0.352	0.391	0.378

Table 6.9: Mean average precision of merged multilingual lists of different methods on HummingBird result lists. Top_x indicates x top documents are downloaded to generate logistic transformation model, C05 indicates both scores from logistic transformation model and centralized document scores are utilized when they are available and they are combined with a linear weight as 0.5. Top_[1,10,20] indicates to download top 1, 10 and 20 documents. C_X means top X documents from each list are merged by their centralized doc scores.

Methods	Train	Test	All
Top_150_C05	0.278	0.297	0.291
Top_30_C05	0.260	0.268	0.265
Top_15_C05	0.235	0.253	0.247
Top_10_C05	0.222	0.248	0.239
Top_5_C05	0.210	0.234	0.226
Top_[1,10,20]_C05	0.199	0.212	0.208
C_1000	0.324	0.343	0.337
C_500	0.315	0.333	0.326
C_150	0.290	0.302	0.298

The experimental results of different variants of these two algorithms are shown in Tables 6.8 and 6.9. It can be seen that both the two query-specific and language-specific results merging algorithms substantially outperformed the query-independent and language-specific algorithms (i.e., results shown in Tables 6.6 and 6.7). The accuracies of the two query-specific and language-specific methods were close on the UniNE system. It is interesting that the Top_150_C05 method outperformed all C_X runs on the UniNE system. One possible explanation is that the estimated document scores can be seen as the combination results from not only the two retrieval methods based on query translation and document translation but also the retrieval method of the UniNE system. Therefore, the combined results that are related with three retrieval systems may be better than those of the exact comparable scores from two retrieval systems. It is encouraging to see that with very limited amount of downloaded documents, the Top_10_C05 method still had a more than 10 percent advantage over the query-independent algorithms. Note that Top_[1, 10, 20]_C05 follows the same strategy of choosing documents to download as described in Chapter 5. On both the two testbeds, Top_[1, 10, 20]_C05 is at least as effective as the query-independent and language-specific algorithms (i.e., results in Tables 6.6 and 6.7). One interesting issue is that the results merging task of multilingual federated search tends to require more training data than the task of monolingual federated search. This may be explained by the hypothesis that multilingual ranked lists tend to be more heterogeneous than monolingual ranked lists. However, more research should be conducted to provide more information.

It can be seen from Table 6.9 that the advantage of the query-specific and language-specific algorithms over the query-independent and language-specific algorithms was even larger for the results on the HummingBird system than those on the UniNE system. This demonstrates the power of the query-specific and language-specific merging algorithms for ineffective bilingual ranked lists. It is interesting to note that the Top_X_C05 runs were not as effective as the C_X runs on the HummingBird System. This may be because the bilingual ranked lists of the HummingBird system are not as accurate as those of the UniNE systems. Therefore, the influence of the HummingBird bilingual ranked lists on the estimated comparable scores is not as helpful as that from the UniNE system.

6.4 Summary

In real world federated search applications, many information sources may contain documents in multiple languages. This use scenario demands a multilingual federated search solution. Particularly, this chapter focuses on the results merging problem in uncooperative multilingual federated search environments, which makes multiple ranked lists in different languages comparable and merges them into a single multilingual ranked list.

To obtain comparable document scores for documents in different languages, it is important to design an effective multilingual centralized retrieval algorithm. This is not our focus in this chapter. However, an effective multilingual centralized retrieval algorithm is briefly introduced in this chapter, which utilizes multiple retrieval methods and multiple translation techniques.

With multilingual centralized retrieval algorithms, some previously proposed multilingual results merging algorithms download, translate and index all returned documents at retrieval time and apply multilingual centralized retrieval algorithms to compute comparable scores for all returned documents and merge them together. This approach is associated with large communication and computation costs. An alternative approach utilizes the relevance judgments of past queries to build query-independent and language-specific models for estimating the probabilities of relevance for returned documents. However, this approach is not as effective.

This chapter proposes an effective and efficient approach to learn query-specific and language-specific models for transforming language-specific document scores into comparable document scores. This is an extension of the semi-supervised learning results merging method proposed in Chapter 5 for monolingual federated search environments. Particularly, this method downloads a small number of documents at retrieval time and utilizes the proposed multilingual centralized retrieval method to calculate their comparable document scores. These documents serve as training data to learn query-specific and language-specific models, which transform the language-specific document scores for all returned documents into centralized comparable scores. Finally, the returned documents can be merged into a single list according to their centralized scores.

A large set of experiments has been conducted to show the effectiveness of proposed algorithms. In order to generate more accurate results for the evaluation, more sophisticated logistic transformation models are utilized in the algorithms with relatively more training data than the algorithms developed in Chapter 5. An extensive set of experiments has demonstrated the greater effectiveness and efficiency of the new results merging algorithm over several other alternatives.

Chapter 7: Unified Utility Maximization Framework

It is common to view the three main subproblems of federated search, namely resource description, resource selection and results merging, in isolation from each other. Effective solutions of these three subproblems are proposed and discussed separately from Chapter 3 through Chapter 6. However, individual solutions for the subproblems optimize different criteria (e.g., high recall for resource selection, high precision for federated document retrieval). These subproblems are correlated with each other in federated search applications; exploring the relationship between these subproblems is as important as proposing separate effective solutions. A unified probabilistic framework is proposed in this chapter for the federated search task in uncooperative environments. The new model integrates and adjusts individual solutions of different subproblems to achieve effective results for different applications. Specifically, when used for an information source recommendation system, the model targets the high-recall goal (select a small number of information sources with as many relevant documents as possible); when used for federated document retrieval, the model is optimized for the high-precision goal (high precision at the top of final merged lists). The new research proposed in previous chapters for individual subproblems of federated search task is integrated into the unified framework. Empirical studies demonstrate the power of this new framework for different federated search applications.

The utility maximization method developed in this chapter can be seen as an extension of the ReDDE resource selection algorithm proposed in Chapter 4. The new resource selection algorithm utilizes a transformation model, which is obtained with a small amount of training data, to estimate the probabilities of relevance for available documents. This generates more robust results than the step function approximation by the ReDDE algorithm.

7.1 High recall of information source recommendation vs. high precision of document retrieval

Information source recommendation and federated document retrieval are two important types of federated search applications, and there has been considerable prior research.

An information source recommendation system is composed of two components: resource representation and resource selection. It recommends the most relevant information sources for users' information needs. This system is very useful when the users want to browse and search the selected information sources manually for broader content instead of asking the system to retrieve relevant documents automatically. Most current resource selection algorithms are evaluated for the high-recall goal, which is to recommend a small number of information sources that contain as many relevant documents as possible.

A federated document retrieval system searches selected information sources automatically and merges returned ranked lists into a single ranked list. Therefore, all the three subproblems of federated search need to be addressed in this application. In operational federated document retrieval application, users rarely browse far down the final ranked list, so the precision at top ranked documents is the most important evaluation metric. Therefore, federated document retrieval application is often evaluated with the high-precision goal.

Most previous algorithms simply combine effective resource selection algorithms and results merging algorithms together in order to achieve accurate results for federated document retrieval. However, this simple approach suffers from the important fact that resource selection algorithm optimized for the high-recall goal of the information source recommendation application is not necessarily optimal for the high-precision goal of the federated document retrieval application. This type of inconsistency has also been observed in previous research [Craswell, 2000].

Some prior research like the decision-theoretic framework (DTF) [Fuhr, 1999; Nottelmann & Fuhr, 2003b] has recognized the high-precision goal for federated document retrieval. This method yields an information source selection solution that minimizes a cost function of overall costs (e.g., retrieval accuracy, query processing cost and communication cost) for federated document retrieval application. When it is focused on retrieval accuracy, the DTF model estimates the probabilities of relevance for the documents among available information sources, and then it generates a resource selection decision for the high-precision goal. However, its empirical results have been shown to be at most as good as those of the CORI resource selection algorithm [Nottelmann & Fuhr, 2003b].

Three issues limit the power of the DTF algorithm: i) the DTF model was proposed for federated document retrieval application and does not address the high-recall goal of information source recommendation application explicitly; ii) the DTF model assumes that the same type of retrieval algorithm is used by all of the information sources, which is not valid in uncooperative environments; and iii) the DTF model builds a separate model for each information source to estimate the probabilities of relevance. This requires human relevance judgments for the results retrieved from each information source, which can be expensive if there are many information sources.

Based on this observation, a unified utility maximization framework is proposed in this chapter to integrate the high-recall goal and the high-precision goal in a single probabilistic model. It works in uncooperative environments and is much more efficient than the DTF model.

7.2 Unified utility maximization framework

A unified utility maximization framework is proposed in this section to address the inconsistency problem of different goals for different federated search applications. This framework integrates the two applications of information source recommendation and federated document retrieval together by assigning them different utility maximization goals.

First, a logistic transformation model is learned off line with a small amount of training queries that have human relevance judgments to map the centralized document scores from the centralized sample database to the corresponding probabilities of relevance. More detailed information about this step is described in Section 7.2.1.1

Second, in resource selection, for each user query (i.e., test query), the probabilities of relevance of all of the (mostly unseen) documents among available information sources can be inferred from the estimated probabilities of relevance of the sampled documents in the centralized sample database. This method is further explained in Section 7.2.1.2.

Third, based on these probabilities, the information sources are ranked by solving different utility maximization problems according to either the high-recall goal or the high-precision goal for different applications. More detailed information can be found in 7.2.2. Note that the probabilities of relevance are used here instead of document scores on centralized sample database. One reason is that probabilities of relevance more precisely represent the utilities of available sources if they are measured by the number of relevant documents contained among available sources (or at the top of ranked lists).

Furthermore, for federated document retrieval, the SSL (Semi-Supervised Learning) results merging algorithm is utilized to rank the returned documents by their estimated centralized document scores (thus also by the probabilities of relevance as we assume the mapping function between the centralized document scores and the probabilities of relevance is monotonically increasing).

When the unified utility maximization framework is optimized for the high-recall goal for information source recommendation, it follows an approach similar to the ReDDE resource selection algorithm proposed in Chapter 4. Both of the two algorithms turn away from the "big document" resource selection approach by explicitly estimating the probabilities of relevance for all documents across available information sources to calculate the expected number of relevant documents. One key point that distinguishes these two methods lies in the different methods to estimate the probability of relevance for each document. Specifically, ReDDE uses a heuristic method and treats the curve of probabilities of relevance as a step function, where only documents' ranks in the centralized complete database are considered; while the new algorithm takes advantage of some training data and builds a query-independent logistic model to transform the *centralized document scores* to their corresponding probabilities of relevance. The improvement enables the new algorithm to generate more robust resource selection decisions than the ReDDE algorithm.

In this section, we first discuss how to estimate the probabilities of relevance for all the documents, and then show how to apply the framework for the information source recommendation application and the federated document retrieval application respectively.

7.2.1 Estimate probabilities of relevance for all documents

For both the information source recommendation application and the federated document retrieval application, a desired resource selection algorithm needs to estimate the probabilities of relevance for all documents across all available information sources. This is a hard problem in an uncooperative environment as query-based sampling only allows a federated search system to observe a very limited proportion of the contents of the information sources. Note that the goal of the ReDDE algorithm proposed in Chapter 4 was to estimate the distribution of relevant documents among available sources. The target in this section is a more precise goal.

To accomplish this goal, our solution addresses the following problems: i) in resource representation, a logistic model is estimated to map source-independent centralized document scores to the probabilities of relevance; and ii) in resource selection, the source-independent centralized document scores for all documents are estimated from the centralized scores for sampled documents, and then the corresponding probabilities of relevance are estimated by the learned logistic model. These two problems are discussed in Sections 7.2.1.1 and 7.2.1.2 respectively.

7.2.1.1 Resource representation: Estimating probabilities of relevance from centralized document scores

There are several transformation methods that can map centralized retrieval document scores to the corresponding probabilities of relevance, for example linear transformation, logistic transformation of raw centralized scores, and logistic transformation of normalized centralized scores. Prior research [Nottelmann & Fuhr, 2003a; Nottelmann & Fuhr, 2003b] has measured the error of these methods and has shown the logistic transformation model using normalized centralized scores to be more effective than other alternatives. The approach of using a transformation model in this section is similar to the approach for results merging in Chapters 5 and 6. However, the goal of the method in this section is to estimate probability of relevance while the goal of the methods in Chapters 5 and 6 is to estimate comparable document scores.

A centralized logistic transformation model is used in this work to map the normalized centralized retrieval score of a document to its corresponding probability of relevance. Formally, the logistic transformation model is expressed as follows:

$$R(d) = P\left(rel|d\right) = \frac{\exp(a_c + b_c \bar{S}_c(d))}{1 + \exp(a_c + b_c \bar{S}_c(d))} \qquad (7.1)$$

where $\bar{S}_c(d)$ denotes the normalized centralized document score (i.e., source-independent score) for a particular document d (i.e., document scores divided by the maximum centralized document score for each query). a_c and b_c are the two parameters of the logistic model. These two parameters are estimated by maximizing the probabilities of relevance of training queries.

More specifically, during training a small set of queries (e.g., 50) is utilized to search on the centralized sample database by the INQUERY retrieval algorithm [Callan, Croft & Broglio, 1995]. The CORI resource selection algorithm is used to rank available information sources and 10 selected information sources are searched for each query (the choice of INQUERY and CORI is not unique, other effective retrieval methods like the language model retrieval algorithm and other resource selection algorithms like ReDDE are also good candidates here). The individual ranked lists for each query are merged into a single list by the SSL results merging algorithm. Then, the top 50 documents in the final ranked list are downloaded and the corresponding centralized scores are calculated by the INQUERY algorithm with the corpus statistics of the centralized sample database. The centralized scores of the downloaded documents are further normalized by dividing the maximum centralized score for each query, as this approach has been suggested to improve estimation accuracy in previous research [Nottelmann & Fuhr, 2003a]. The normalized document scores are used as inputs for training transformation model, while human relevance judgments for these documents are obtained as outputs. Finally, the logistic transformation model is built by the maximum likelihood estimation criterion.

Only a single centralized logistic transformation model is built on centralized sample database. The centralized sample database serves as a bridge to connect the centralized transformation model and all of the information sources to estimate the probabilities of relevance of all documents across the information sources. The human efforts required to train the single centralized logistic model do *not* scale with the number of information sources. This marks a big distinction between the unified utility maximization model and the prior research that required building separate models for different information sources such as the RDD [Voorhees et al., 1995] or the DTF [Fuhr, 1999; Nottelmann & Fuhr, 2003b] methods. This advantage suggests that the unified utility maximization method can be trained much more efficiently and thus is much easier to apply in large federated search environments with many information sources.

Note that this method uses one model for all queries. This solution works with a limited amount of training queries. It does not need to be retrained when a few new information sources are added. However, when there exist a large amount of new sources, the centralized sample database may be rebuilt and the logistic model can be re-estimated to reflect the content change.

7.2.1.2 Resource selection: Estimating probabilities of relevance from centralized document scores for all documents

In resource selection, the probabilities of relevance for all of the documents in available information sources are estimated based on centralized retrieval document scores by using the centralized logistic transformation model described in Section 7.2.1.1. Therefore, it is necessary to estimate the centralized document scores for all of the (mostly unseen) documents. These scores are inferred from the centralized retrieval scores of the sampled documents in the centralized sample database and also the information source size estimates. Particularly, a mapping is described in this section to transform the rank of a document in the centralized environment to the corresponding centralized document scores. This follows a main theme of this dissertation to utilize the centralized sample database as described in Section 3.2.

The concept of information source size factor was introduced in Chapter 4. It is associated with a particular i^{th} information source and is defined as the ratio of the estimated information source size and the number of

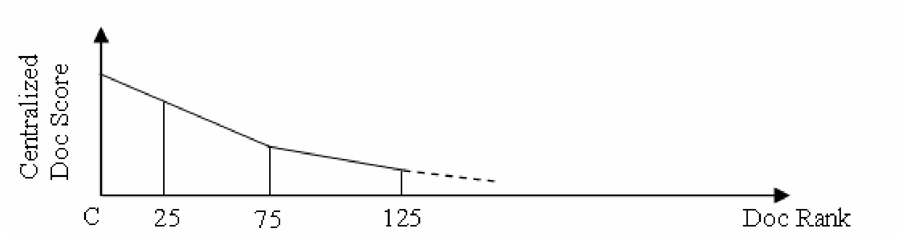

Figure 7.1: The complete centralized document score curve constructed by linear interpolation for a particular information source with scale factor of 50.

sampled documents from this information source. Formally:

$$SF_{db_i} = \frac{\hat{N}_{db_i}}{N_{db_i\_samp}} \tag{7.2}$$

where $\hat{N}_{db_i}$ denotes the information source size estimate for the ith information source and N_{dbi_samp} denotes the number of sampled documents from this information source. The intuition behind the information source size factor is that: for a particular information source with an information source size factor of 50, if a sampled document from this information source has a centralized document score of 0.8, it can be roughly estimated that there are another 49 similar unseen documents (i.e., not sampled) in the information source, which have centralized document scores of about 0.8, as long as we assume the sampled documents are representative.

Instead of the simple histogram non-parametric estimator for centralized document scores, we can choose a finer linear interpolation estimator. Formally, all of the sampled documents from the particular ith information source are first ranked by their centralized document scores to get the sampled centralized document score list as {$S_c(ds_{i1})$, $S_c(ds_{i2})$, $S_c(ds_{i3})$,…..}. Suppose that we can calculate the centralized document scores for all of the documents from this information source and obtain the complete centralized document score list, then the top document in the sampled list would rank at the position of $SF_{dbi}/2$ in the complete list, the second document in the sampled list would rank at the position of $SF_{dbi}3/2$, and so on. Therefore, the data points of sampled documents in the complete centralized document score curve are: {($SF_{dbi}/2$, $S_c(ds_{i1})$), ($SF_{dbi}3/2$, $S_c(ds_{i2})$), ($SF_{dbi}5/2$, $S_c(ds_{i3})$),…}. Piecewise linear interpolation is applied to estimate the centralized document curve, as illustrated in Figure 7.1. Finally, the whole complete centralized document score list $\hat{S}_c(d_{ij})$, $j \in [1, \hat{N}_{db_i}]$ can be extracted from the curve accordingly.

It can be seen from Figure 7.1 that more sampled data points produce more accurate complete centralized document score curves. In contrast, with very sparse sampled data points, the estimation may be inaccurate especially for the top ranked documents (e.g., [1, $SF_{dbi}/2$]), which are more important to users as they are more likely to be relevant. This problem is more serious for information sources with large source scale factors. Based on this observation, an alternative approach is proposed to adjust the estimated centralized document scores of top ranked documents for information sources with large information source scale factors (i.e., empirically set to be larger than 100 in this chapter). Specifically, a logistic transformation model is learned for each information source with a large information source scale factor to estimate the centralized document score of the top 1 document by using the centralized document scores of the top two sampled documents from this information source as:

$$\hat{S}_c(d_{i1}) = \frac{\exp(\alpha_{i0}+\alpha_{i1}S_c(ds_{i1})+\alpha_{i2}S_c(ds_{i2}))}{1+\exp(\alpha_{i0}+\alpha_{i1}S_c(ds_{i1})+\alpha_{i2}S_c(ds_{i2}))} \qquad (7.3)$$

where $\hat{S}_c(d_{i1})$ is the estimated centralized document score of the top 1 document in the i^{th} information source. a_{i0}, $a_{i1}$and a_{i2} are the three parameters of the corresponding logistic model. For each of the information source with a large information source scale factor, the top retrieved documents for all training queries are downloaded and their centralized document scores are calculated. Together with the centralized document scores of the top two sampled documents for these queries, the three parameters of the logistic model can be estimated. Note that this logistic model can be trained in an automatic way without utilizing any human relevance judgment. Therefore, it is a reasonable choice to build a separate model for each large information source.

After a logistic transformation model has been built for an information source to estimate the centralized score of the top document, an exponential function is fitted for the top part ([1, $SF_{dbi}/2$]) of the complete centralized document score curve as follow:

$$\hat{S}_c(d_{ij}) = \exp(\beta_{i0}+\beta_{i1} * j) \quad j\in[1,SF_{db_i}/2] \qquad (7.4)$$

$$\beta_{i0} = \log(\hat{S}_c(d_{i1})) - \beta_{i1} \qquad (7.5)$$

$$\beta_{i1} = \frac{(\log(S_c(ds_{i1})-\log(\hat{S}_c(d_{i1}))}{(SF_{db_i}/2-1)} \qquad (7.6)$$

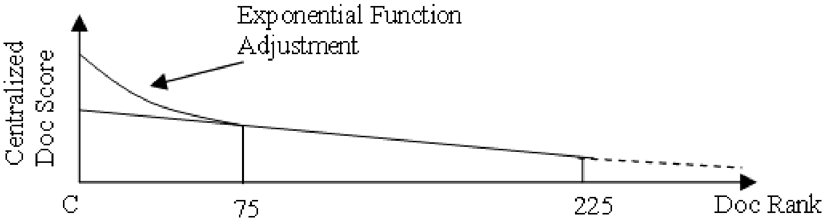

Figure 7.2: The adjusted centralized document score curve for a large information source with scale factor of 150.

The two parameters β_{i0} and β_{i1} are fitted to make the exponential function pass through the two points of $(1, S_c(\hat{d}_{i1}))$ and $(SF_{dbi}/2,\ S_c(ds_{i1}))$. The exponential function is only used to adjust the top part of the centralized document score curve and the lower part is still fitted with the linear interpolation approach. This adjustment is shown in Figure 7.2.

From the adjusted centralized document score curves, the complete centralized document score lists can be estimated. Then with the logistic model described in Section 7.2.1.1, the most probable complete lists of probabilities of relevance for all the documents across available information sources can be derived from the estimated centralized document scores as: $\theta^* = \left\{(R(\hat{d}_{1j}), j \in [1, \hat{N}_{db_1}]),\ (R(\hat{d}_{2j}), j \in [1, \hat{N}_{db_2}]), \ldots\right\}$ ($R(\hat{d}_{ij})$ denotes the estimated probability of relevance for the top ith document by the centralized retrieval algorithm from the jth information source). This information is very important for the unified utility maximization model.

7.2.2 Unified utility maximization model

Information source recommendation systems rank and select available information sources for user queries, while federated document retrieval systems not only need to rank information sources but also need to decide how many documents to retrieve from each selected information source. The resource selection action of the information source recommendation application can be generalized as a special case of the resource selection action of the federated document retrieval application, which implicitly recommends all of the documents in the selected information sources. Formally, let d_i denote the number of documents to select from a particular ith information source and the vector of $\vec{d} = \{d_1, d_2, \ldots..\}$ denote the selection decision for all of the available information sources.

The resource selection decision is made based on the complete lists of probabilities of relevance for all available information sources, which are derived in Section 7.2.1. Those lists are inferred from all the available information, namely $\overrightarrow{R_s}$ that stands for the resource descriptions acquired by query-based sample and information source size estimates obtained by the Sample-Resample method, and $\overrightarrow{S_c}$ that stands for the centralized document scores of the documents in the centralized sample database. The random vector θ denotes an arbitrary set of the complete lists of probabilities of relevance and $P(\theta|\overrightarrow{R_s},\overrightarrow{S_c})$ denotes the corresponding generation probability. Furthermore, a utility function $U(\theta,\vec{d})$ is defined as the benefit that can be gained by making the selection action $\vec{d}$ when the true complete lists of probabilities of relevance are θ. Finally, the desired selection action derived from the Bayesian framework to maximize the utility is as follows:

$$\vec{d}^* = \underset{\vec{d}}{\operatorname{argmax}} \int_\theta U(\theta,\vec{d})\, P(\theta|\overrightarrow{R_s},\overrightarrow{S_c})\, d\theta \tag{7.7}$$

However, it is not easy to derive an accurate expression for $P(\theta|\overrightarrow{R_s},\overrightarrow{S_c})$; even when one exists, the computation costs are generally not acceptable as there are infinite choices of θ. A common approach to simplify the computation in the Bayesian framework is to calculate the utility function $U(\theta^*,\vec{d})$ at the most probable parameter values instead of calculating the whole expectation. This section has shown how to derive the most probable parameter θ^*. Then Equation 7.7 can be simplified as:

$$\vec{d}^* = \underset{\vec{d}}{\operatorname{argmax}}\ U(\theta^*,\vec{d}) \tag{7.8}$$

This equation serves as the basic model from which the desired resource selection decisions can be derived for both the information source recommendation application and the federated document retrieval application.

7.2.2.1 Desired resource selection decision for information source recommendation application

The high-recall goal of the information source recommendation application is to select a small number of information sources (e.g., N_{sdb} information sources) that contain as many relevant documents as possible. This criterion can be formalized as follows:

$$U(\theta^*,\vec{d}) = \sum_i I(db_i) \sum_{j=1}^{\hat{N}_{db_i}} R(\hat{d}_{ij}) \tag{7.9}$$

where $I(db_i)$ is a binary indication function, which is 1 when a particular i^{th} information source is selected and

0 otherwise. Plugging this utility function into the basic model in Equation 7.8 and associating it with the constraint of the number of selected information sources, the following utility maximization problem can be obtained:

$$\begin{aligned} &\vec{d}^{*} = \underset{\vec{d}}{\operatorname{argmax}} \sum_{i} I(db_i) \sum_{j=1}^{\hat{N}_{db_i}} R(\hat{d}_{ij}) \\ &\text{Subject to:} \sum_{i} I(db_i) = N_{sdb} \end{aligned} \tag{7.10}$$

As the contribution of the available information sources is not coupled with each other (the number of relevant documents that an information source contains is not affected by another information source), the solution of the above utility maximization problem is simple. The expected number of relevant documents in each information source can be calculated as follows:

$$Rel\_Q(i) = \sum_{j=1}^{\hat{N}_{db_i}} R(\hat{d}_{ij}) \tag{7.11}$$

where $R(\hat{d}_{ij})$ denotes the estimated probability of relevance for the top j^{th} document by the centralized retrieval algorithm from the i^{th} information source, section 7.2.1.2 has shown how to derive this value. The information sources are sorted by the expected number of relevant documents they contain and the top N_{sdb} information sources can be selected to obtain the high-recall results. This method is called the UUM/HR algorithm (Unified Utility Maximization for High-Recall).

7.2.2.2 Desired resource selection decision for federated document retrieval application

The above section discusses how to use the complete lists of probabilities of relevance to achieve the high-recall goal for information source recommendation application. However, for a federated document retrieval application, the situation is more complex and we need to address two additional problems: i) documents with high probability of relevance may not be returned by the search engines of individual information sources; and ii) when they are returned, they may not be ranked highly in the final merged result list. First, it is assumed in this chapter that all the individual search engines are effective (the assumption of effective search engines is revisited and relaxed in Chapter 8). Second, the SSL (Semi-Supervised Learning) results merging algorithm is applied to automatically transform source-specific scores to the centralized document scores and rank all of the returned documents accordingly. We assume that the SSL algorithm can obtain the centralized document scores with high accuracy, so the final result list created by the SSL algorithm is actually sorted by the centralized document scores (thus the probabilities of relevance). Based on these assumptions, the complete lists of probabilities of relevance can also be utilized for the federated document retrieval application.

The accuracy of a federated document retrieval system is measured by the high-precision criterion as the Precision at the top of the final merged document list. Thus, the utility function for federated document retrieval should reflect this high-precision goal as follows:

$$U(\theta^*,\vec{d}) = \sum_{i} I(db_i)\sum_{j=1}^{d_i} R(\hat{d}_{ij}) \qquad (7.12)$$

Note that the key difference between this utility function and the utility function for the information source recommendation application in Equation 7.9 is that for the information source recommendation application, the probabilities of relevance of all the documents in an information source are summed together, while here a much smaller proportion of the documents are considered.

Combining the basic model in Equation 7.8 and the above utility function, the general resource selection utility maximization problem of a federated document retrieval application can be obtained as follows:

$$\vec{d}^* = \underset{\vec{d}}{\mathrm{argmax}} \sum_{i} I(db_i)\sum_{j=1}^{d_i} R(\hat{d}_{ij}) \qquad (7.13)$$

In the most general case, a simple deterministic method can solve the above utility maximization problem when it is not associated with any constraint. The simple method first ranks all the documents $\left\{(d_{1j},\ j\in[1,\hat{N}_{db_1}]),\ (d_{2j},\ j\in[1,\hat{N}_{db_2}]),\ldots\right\}$ into a unified ranked list by their estimated probabilities of relevance as $\left\{R(\hat{d}_{ij})\right\}$. Then, this method simply runs down the unified ranked list to select the top ranked documents. This method is very efficient.

In practice, different constraints are associated with the resource selection problems for different federated document retrieval applications. For example, there is communication cost, time delay or monetary cost to search each additional information source. Therefore, many federated document retrieval systems only select a few of the information sources available. One simple configuration, which is consistent with the settings of most prior federated document retrieval research, is to select a maximum number (e.g., N_{sdb}) of information sources and retrieve a maximum number (e.g., N_{rdoc}) of documents from each selected information source. This can be formalized as follows:

$$\begin{aligned}&\vec{d}^{\,*} = \underset{\vec{d}}{\operatorname{argmax}} \sum_{i} I(db_i) \sum_{j=1}^{d_i} R(\hat{d}_{ij}) \\ &\text{Subject to:} \sum_{i} I(db_i) = N_{sdb} \\ &d_i = N_{rdoc}\ ,\ \text{if } I(db_i) \neq 0\end{aligned} \tag{7.14}$$

This problem can be easily solved by a variant of the deterministic algorithm. The algorithm first estimates the number of relevant documents at the top of each information source's complete list of probabilities of relevance:

$$Rel\_Q(i) = \sum_{j=1}^{N_{rdoc}} R(\hat{d}_{ij}) \tag{7.15}$$

The N_{sdb} information sources with the highest $Rel\_Q(i)$ values are selected. As this resource selection algorithm described in Equation 7.14 focuses on the high-precision goal and it retrieves fixed lengths of document ranked lists from the selected information sources, it is called the UUM/HP-FL algorithm.

For the toy example of the resource selection problem described in Table 7.1, which wants to retrieve 2 documents from 2 of the 4 available sources, this algorithm will select 2 documents from both Source 1 and Source 3, and the obtained total utility is 1.35.

Another simple algorithm using the unified ranked list simply runs down the unified ranked list to select top ranked documents and skip the documents that do not satisfy the constraints (i.e., the number of sources to search and the number of document to retrieve from each source). This method works fast but may not obtain a guaranteed optimal solution in contrast with the utility maximization problem in Equation 7.13 where there are no constraints. For example, for the resource selection problem in Table 7.1 to retrieve 2 documents from 2 sources, the simple algorithm using unified ranked list obtains the total utility as 1.3 by selecting Source 1 and Source 2.

In the above resource selection problem, a fixed number of documents are retrieved from each selected information source. Information sources of high quality may be able to contribute more relevant documents than information sources of low quality. Based on this observation, a more complex configuration is proposed for the federated document retrieval application to vary the number of retrieved documents from selected information sources. Specifically, the selected information sources are allowed to return ranked document lists of different lengths. The lengths of the ranked lists are required to be multiples of a baseline number as 10. This requirement simulates the behavior of commercial search engines on the Web, which provide retrieval results in a page-by-page manner or that allow the length of a results list to be specified in

Table 7.1: A toy example of a resource selection problem that selects 2 sources out of 4 available sources.

	Estimated Probabilities of Relevance			
	Source 1	**Source 2**	**Source 3**	**Source 4**
Rank 1	0.5	0.4	0.35	0.3
Rank 2	0.2	0.2	0.3	0.3
Rank 3	0.05	0.05	0.05	0.3

10-document increments (e.g., Google). For further simplification, each selected information source is allowed to return at most 100 documents. This utility maximization problem can be formalized as follows:

$$\begin{aligned}
&\vec{d}^{*} = \underset{\vec{d}}{\operatorname{argmax}} \sum_{i} I(d_i) \sum_{j=1}^{d_i} R(\hat{d}_{ij}) \\
&\text{Subject to: } \sum_{i} I(d_i) = N_{sdb} \\
&\sum_{i} d_i = N_{Total\_rdoc} \\
&d_i = 10 * k, \quad k \in [0,\ 1,\ 2,\ ..,\ 10]
\end{aligned} \tag{7.16}$$

where N_{Total_rdoc} denotes the total number of documents to be retrieved.

There is no simple solution for this utility maximization problem, as there was for the UUM/HR or UUM/HP-FL algorithms. For example, one simplified configuration of the problem is to retrieve a total 4 of documents from 2 sources among the 4 available sources, as described in Table 7.1. These constraints allow each selected source to return 1, 2 or 3 documents. The two simple algorithms proposed for UUM/HP-FL obtain the total utilities of 1.35 and 1.3 respectively. However, a better choice obtains a total utility of 1.4 by retrieving 1 document from Source 1 and 3 documents from Source 4. Therefore, although simple algorithms can generate reasonably good solutions in many cases, their performance is not optimal.

A dynamic programming solution is proposed to solve this utility maximization problem. The basic steps of the dynamic programming method are described in Figure 7.3. This resource selection algorithm is called the UUM/HP-VL algorithm (Unified Utility Maximization for High-Precision with Variable Length ranked lists).

The utility maximization methods in Equations 7.14 and 7.16 solve the resource selection problems of federated document retrieval applications in two different configurations. After the resource selection decision is made, the user queries are sent to search these selected information sources and the returned ranked lists are merged into a single list by the SSL algorithm. The SSL algorithm ranks the returned

```
Input:  Complete lists of probabilities of relevance for all the |DB| information sources.
Output: Optimal selection solution for the optimization problem in Equation 7.16.
    i)   Create the data structure of a three-dimensional array:
             $Sel\ (1..|DB|, 1..N_{Total\_rdoc/10}, 1..N_{sdb})$
         Each Sel (x, y, z) is associated with a selection decision $\vec{d}_{xyz}$, which represents the best selection
         decision in the condition: only information sources from number 1 to number x are considered for
         selection; totally y*10 documents will be retrieved; only z information sources are actually chosen
         out of the x source candidates. Sel (x, y, z) is the corresponding utility value generated by choosing
         the best selection.
    ii)  Initialize $Sel\ (1, N_{Total\_rdoc/10}, 1..N_{sdb})$ with only the complete list of probabilities of relevance for the
         1st information source.
    iii) Iterate the current database candidate i from 2 to |DB|
           For each entry Sel (i, y, z):
             Find k* such that:
               $k^* = \underset{k}{\operatorname{argmax}} (Sel(i\text{-}1,y\text{-}k,z\text{-}1)+ \sum_{j\le 10*k} R(\hat{d}_{ij}))$
               subject to: $1 \le k \le \min(y,10)$
             If $(Sel(i\text{-}1,y\text{-}k^*,z\text{-}1)+ \sum_{j\le 10*k^*} R(\hat{d}_{ij})) > Sel(i\text{-}1,y,z)$
               We should retrieve $10*k^*$ documents from the ith information source, then update the
               previous values of Sel (i-1, y, z) and set $\vec{d}_{iyz}$ accordingly.
             If $(Sel(i\text{-}1,y\text{-}k^*,z\text{-}1)+ \sum_{j\le 10*k^*} R(\hat{d}_{ij})) \le Sel(i\text{-}1,y,z)$
               We should not select this information source and the previous best solution Sel (i-1, y, z)
               should be kept. Set $\vec{d}_{iyz}$ accordingly.
    iv)  The best selection solution is given by $\vec{d}_{|DB|N_{Toral\_rdoc/10}N_{sdb}}$ and the corresponding utility value is
         $Sel\ (|DB|, N_{Total\_rdoc/10}, N_{sdb})$.
```

Figure 7.3: The dynamic programming optimization procedure for Equation 7.16.

documents by their estimated probabilities of relevance, which is consistent with our assumption described in the beginning of this section.

7.3 Evaluation methodology and experimental results

In contrast to many previous algorithms, a small number of training queries with human relevance judgments is required by the unified utility maximization framework to train its model. This section first explains the experimental methodology, and then shows the empirical results of the information source recommendation application and the federated document retrieval application respectively.

7.3.1 Experimental methodology

The unified utility maximization framework needs a small amount of queries with relevance judgments as the training data. The Trec123_100Col testbed was chosen in the experiments as there are 100 TREC queries on this testbed, where 50 queries served as the training data and another 50 served as the test data. Furthermore, the three testbeds of representative, relevant and nonrelevant, which are built based on the Trec123_100Col testbed (more detail in Chapter 2), were also used. All the four testbeds provide a wide range of corpus characteristics for conducting thorough evaluation.

100 queries were created from the title fields of TREC topics 51-150. 50 queries from topics 101-150 served as the training queries and another 50 queries from topics 51-100 were used as the test queries. This arrangement was made as a lot of prior research [Callan, 2000; Si and Callan, 2003b; Nottelmann & Fuhr, 2003b] has been evaluated on TREC topics 51-100 and it makes easier to compare the experimental results in this section to prior research as well as other results reported in this dissertation (e.g., Chapters 4 and 5).

Uncooperative federated search environments often contain multiple types of search engines. In order to simulate this characteristic, three types of search engines as INQUERY, language modeling and the vector space model introduced in Chapter 2 were implemented in the Lemur toolkit and assigned to the information sources in a round-robin manner.

Query-based sampling was used to acquire the resource descriptions and build the centralized sample database. About 80 queries were sent to each information source to download 300 documents. The Sample-Resample method was used to obtain the information source size estimates.

7.3.2 Experimental results for information source recommendation application

An information source recommendation system suggests relevant information sources to users. It is typically evaluated using the recall metric R_n, which compares a particular algorithm with the relevance-based ranking strategy (more detail is in Chapter 4).

The experiments in this section compare three resource selection algorithms for information source recommendation application, namely CORI, ReDDE (Chapter 4) and UUM/HR, on the four testbeds. The experiment results are shown in Figure 7.4.

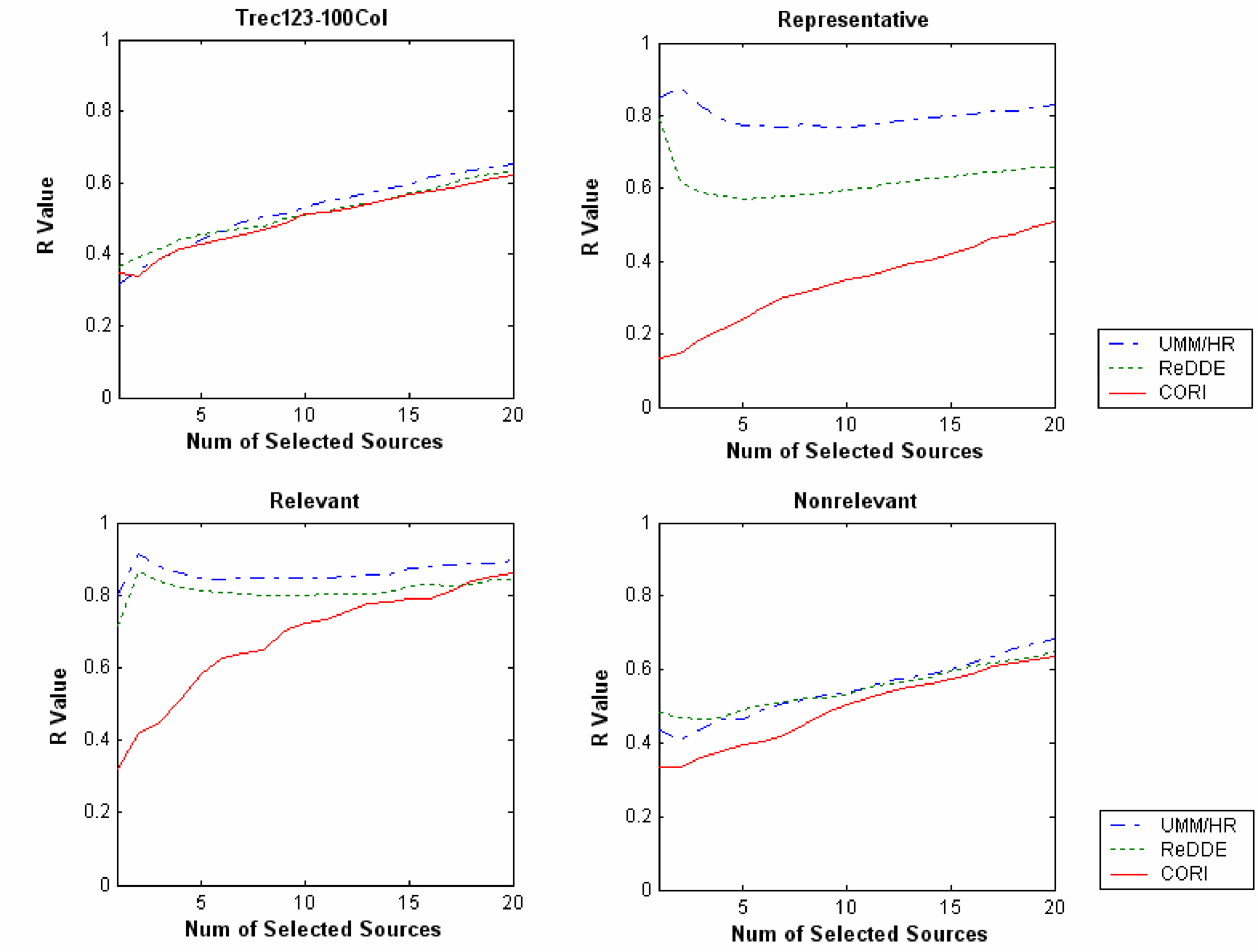

Figure 7.4: Resource selection experiments on four testbeds.

It can be seen from this figure that the UUM/HR resource selection algorithm and the ReDDE algorithm were more effective (on the representative, relevant and nonrelevant testbeds) or as accurate as (on Trec123_100Col testbed) the CORI resource selection algorithm. The advantages of the UUM/HR algorithm and the ReDDE algorithm are more notable on the representative and relevant testbeds, where large information sources contain a large proportion of relevant documents and the CORI algorithm suffered substantially from the "big document" assumption without considering the information source size factors. This indicates that the power of the UUM/HR algorithm and the ReDDE algorithm comes from introducing information source size factors and explicitly estimating the probabilities of relevance for all documents across available information sources to optimize the high-recall goal.

Another observation can be drawn from Figure 7.4 is that the UUM/HR resource selection algorithm was more accurate than the ReDDE algorithm on the representative testbed and the relevant testbed, and it was about as effective as the ReDDE algorithm on the Trec123_100Col testbed and the nonrelevant testbed. This suggests that the UUM/HR algorithm is more robust than the ReDDE algorithm, due to introducing the centralized logistic model to estimate the probabilities of relevance and taking advantage of the training data. However, careful analysis shows that when only a few information sources were selected on the Trec123_100Col testbed or on the nonrelevant testbed, the ReDDE algorithm had a small advantage over the UUM/HR algorithm. Two reasons can be used to explain this minor puzzle: i) the ReDDE algorithm was tuned on the Trec123_100Col testbed (set the optimal threshold value); and ii) although the difference is small, this may indicate that our centralized logistic model of estimating probabilities of relevance is not effective enough. More training data or a more sophisticated model may help to solve this minor puzzle.

7.3.3 Experimental results for the federated document retrieval application

For federated document retrieval, user queries are sent to search the selected information sources and the individual ranked lists are merged into a single list by the minimum downloading variant of the SSL algorithm as described in Chapter 5. This type of SSL results merging algorithm downloads a small number of returned documents on the fly to create enough training data.

Experiments have been conducted to compare the effectiveness of five algorithms, namely the CORI, ReDDE, UUM/HR, UUM/HP-FL and UUM/HP-VL algorithms. The Trec123_100Col and representative testbeds were selected as they represent two extreme cases: the CORI resource selection algorithm is about as effective as the ReDDE algorithm and the UUM/HR algorithm for the high-recall goal on the Trec123_100Col testbed and is much worse than the ReDDE and HMM/HR algorithms on the representative testbed. Three configurations were conducted on both the two testbeds to select 3, 5 or 10 information sources. The four algorithms of CORI, ReDDE, UUM/HR and UUM/HP-FL retrieved a fix number of 50 documents from each selected information source, while the UUM/HP-VL algorithm was allowed to adjust the number from 10 to 100 and a multiple of 10.

The experiment results on the Trec123_100Col testbed and the representative testbed are shown in Tables 7.2-7.4 and Tables 7.5-7.7 respectively.

It can be observed from both sets of experiments that the difference between the accuracies of the algorithms was reduced when more information sources were selected. This is consistent with our expectation that the overlap in the selected information sources among different algorithms is much larger in the case of selecting many information sources than in the case of selecting a small number of information sources.

More specifically, on the Trec123_100Col testbed, the document retrieval accuracy with the CORI selection

Table 7.2: Precision on the Trec123_100Col testbed when 3 sources were selected. (CORI is the baseline.)

Precision at Doc Rank	CORI	ReDDE	UMM/HR	UMM/HP-FL	UMM/HP-VL
5 docs	0.3640	0.3480 (-4.4%)	0.3960 (+8.8%)	0.4680 (+28.6%)	0.4640 (+27.5%)
10 docs	0.3360	0.3200 (-4.8%)	0.3520 (+4.8%)	0.4240 (+26.2%)	0.4220 (+25.6%)
15 docs	0.3253	0.3187 (-2.0%)	0.3347 (+2.9%)	0.3973 (+22.2%)	0.3920 (+20.5%)
20 docs	0.3140	0.2980 (-5.1%)	0.3270 (+4.1%)	0.3720 (+18.5%)	0.3700 (+17.8%)
30 docs	0.2780	0.2660 (-4.3%)	0.2973 (+6.9%)	0.3413 (+22.8%)	0.3400 (+22.3%)

Table 7.3: Precision on the Trec123_100Col testbed when 5 sources were selected. (CORI is the baseline.)

Precision at Doc Rank	CORI	ReDDE	UMM/HR	UMM/HP-FL	UMM/HP-VL
5 docs	0.4000	0.3920 (-2.0%)	0.4280 (+7.0%)	0.4680 (+17.0%)	0.4600 (+15.0%)
10 docs	0.3800	0.3760 (-1.1%)	0.3800 (0.0%)	0.4180 (+10.0%)	0.4320 (+13.7%)
15 docs	0.3560	0.3560 (0.0%)	0.3720 (+4.5%)	0.3920 (+10.1%)	0.4080 (+14.6%)
20 docs	0.3430	0.3390 (-1.2%)	0.3550 (+3.5%)	0.3710 (+8.2%)	0.3830 (+11.7%)
30 docs	0.3240	0.3140 (-3.1%)	0.3313 (+2.3%)	0.3500 (+8.0%)	0.3487 (+7.6%)

Table 7.4: Precision on the Trec123_100Col testbed when 10 sources were selected. (CORI is the baseline.)

Precision at Doc Rank	CORI	ReDDE	UMM/HR	UMM/HP-FL	UMM/HP-VL
5 docs	0.4400	0.4400 (0.0%)	0.4800 (+9.1%)	0.4680 (+6.4%)	0.4800 (+9.1%)
10 docs	0.4300	0.4080 (-5.1%)	0.4400 (+2.3%)	0.4520 (+5.1%)	0.4540 (+5.6%)
15 docs	0.4187	0.3840 (-8.3%)	0.4187 (+0.0%)	0.4320 (+3.2%)	0.4333 (+3.5%)
20 docs	0.3980	0.3750 (-5.8%)	0.3980 (+0.0%)	0.4040 (+1.5%)	0.4120 (+3.5%)
30 docs	0.3653	0.3513 (-3.8%)	0.3720 (+1.8%)	0.3820 (+4.8%)	0.3793 (+3.8%)

algorithm was about the same or a little bit better than that with the ReDDE selection algorithm, while the UUM/HR algorithm had a small advantage over both of them. Note that Chapter 4 suggests that the ReDDE algorithm should produce better resource selection results for information source recommendation. However, here the goal is federated document retrieval instead of source recommendation. The experiments results have shown that ReDDE algorithm does not have advantage over CORI algorithm.

The main difference between the UUM/HR algorithm and the ReDDE algorithm is that ReDDE uses a heuristic method to assume the curve of the probabilities of relevance as a step function, while UUM/HR takes advantage of training data and builds a finer logistic model to transform the centralized document scores to the corresponding probabilities of relevance. This difference makes the UUM/HR better than the ReDDE algorithm in some extent at distinguishing the documents with high probabilities of relevance from

Table 7.5: Precision on the representative testbed when 3 sources were selected. (CORI is the baseline.)

Precision at Doc Rank	CORI	ReDDE	UMM/HR	UMM/HP-FL	UMM/HP-VL
5 docs	0.3720	0.4080 (+9.7%)	0.4640 (+24.7%)	0.4600 (+23.7%)	0.5000 (+34.4%)
10 docs	0.3400	0.4060 (+19.4%)	0.4600 (+35.3%)	0.4540 (+33.5%)	0.4640 (+36.5%)
15 docs	0.3120	0.3880 (+24.4%)	0.4320 (+38.5%)	0.4240 (+35.9%)	0.4413 (+41.4%)
20 docs	0.3000	0.3750 (+25.0%)	0.4080 (+36.0%)	0.4040 (+34.7%)	0.4240 (+41.3%)
30 docs	0.2533	0.3440 (+35.8%)	0.3847 (+51.9%)	0.3747 (+47.9%)	0.3887 (+53.5%)

Table 7.6: Precision on the representative testbed when 5 sources were selected. (CORI is the baseline.)

Precision at Doc Rank	CORI	ReDDE	UMM/HR	UMM/HP-FL	UMM/HP-VL
5 docs	0.3960	0.4080 (+3.0%)	0.4560 (+15.2%)	0.4280 (+8.1%)	0.4520 (+14.1%)
10 docs	0.3880	0.4060 (+4.6%)	0.4280 (+10.3%)	0.4460 (+15.0%)	0.4560 (+17.5%)
15 docs	0.3533	0.3987 (+12.9%)	0.4227 (+19.6%)	0.4440 (+25.7%)	0.4453 (+26.0%)
20 docs	0.3330	0.3960 (+18.9%)	0.4140 (+24.3%)	0.4290 (+28.8%)	0.4350 (+30.6%)
30 docs	0.2967	0.3740 (+26.1%)	0.4013 (+35.3%)	0.3987 (+34.4%)	0.4060 (+36.8%)

Table 7.7: Precision on the representative testbed when 10 sources were selected. (CORI is the baseline.)

Precision at Doc Rank	CORI	ReDDE	UMM/HR	UMM/HP-FL	UMM/HP-VL
5 docs	0.3920	0.3720 (-5.1%)	0.4480 (+14.3%)	0.4360 (+11.2%)	0.4520 (+15.3%)
10 docs	0.3860	0.3840 (-0.5%)	0.4520 (+17.1%)	0.4440 (+15.0%)	0.4560 (+18.1%)
15 docs	0.3813	0.3680 (-3.5%)	0.4373 (+14.7%)	0.4387 (+15.1%)	0.4453 (+16.8%)
20 docs	0.3710	0.3710 (+0.0%)	0.4250 (+14.5%)	0.4300 (+15.8%)	0.4370 (+17.7%)
30 docs	0.3513	0.3640 (+3.6%)	0.4140 (+17.9%)	0.4247 (+20.9%)	0.4227 (+20.3%)

documents with low probabilities of relevance. Therefore, the UUM/HR tends to be more robust and more accurate than the ReDDE algorithm. However, the advantage of the UUM/HR was small since it does not explicitly optimize the selection action according to the high-precision goal as what the UUM/HP-FL and UUM/HP-VL algorithms are designed to do. It can be noted from Tables 7.2-7.4 that the UUM/HP-FL and UUM/HP-VL algorithms were much more effective than the other algorithms when a small number of information sources were selected (i.e., 3 or 5) and still had a small advantage with a large number of selected information sources (i.e., 10). This suggests that the power of these two algorithms comes from the explicit optimization of the high-precision goal of the federated document retrieval application.

On the representative testbed, the CORI algorithm was not as effective as the other algorithms especially when a small number of information sources (i.e., 3 and 5) were selected. This can be explained by the fact that CORI does not consider information source size factor and thus the two large information sources with more relevant documents were not ranked highly in its information source ranking. The document retrieval results with the ReDDE algorithm were more accurate (when 3 or 5 information sources were selected) or at least as good as (when 10 information sources were selected) than those with the CORI algorithm, but were consistently worse than the results with the UUM/HR algorithm. It can be noticed from Tables 7.5-7.7 that all three variants of the UUM algorithms (UUM/HR, UUM/HP-FL and UUM/HP-VL) were about equally effective. Carefully analysis shows that the overlap of the selected information sources among these three algorithms was much larger on the representative testbed than that on the Trec123_100Col testbed as all of the three algorithms tended to select the two large information sources with a lot of relevant documents on the representative testbed. Therefore, the document retrieval results produced by the three UUM algorithms were roughly the same.

To summarize, the strategy of explicitly optimizing the high-precision goal is very important to obtain accurate results for the federated document retrieval application. The algorithms designed according to this criterion have been shown to be very effective in different environments.

7.4 Summary

Most prior research in federated search treated the three main subproblems, namely resource description, resource selection and results merging, in isolation from each other, and built systems for different federated search applications by simply combining individually effective solutions. However, these individual solutions optimize different criteria (e.g., high recall for resource selection, high precision for federated document retrieval). It is not a good choice to simply combine them together. For example, a resource selection algorithm optimized for high recall may not work well for a federated document retrieval application.

Based on this observation, a unified utility maximization framework is proposed in this chapter for federated search tasks in uncooperative environments. This model adjusts and integrates the individual solutions of different subproblems to obtain effective results for different federated search applications. Specifically, the model is optimized for the high-recall goal for an information source recommendation application and it focuses on the high-precision goal for a federated document retrieval task. When the framework is optimized for information source recommendation, it can be seen as an extension of the ReDDE resource selection algorithm proposed in Chapter 4. However, a refined transformation model is utilized here to estimate probabilities of relevance of all available documents instead of the rough step function approximation of the ReDDE algorithm.

In order to accomplish its goal, the unified utility maximization framework estimates the probabilities of relevance for all documents (mostly unseen) across available information sources. It first uses a small amount of training data in resource representation to build a *query-independent* logistic model on centralized sample database to transform source-independent centralized document scores to probabilities of relevance. Furthermore, in resource selection, the unified utility maximization framework calculates the centralized documents scores for all sampled documents within the centralized sample database. It then estimates the centralized document scores for all documents across the available sources and finally obtains the estimated probability of relevance for each document.

With the estimated probabilities of relevance, the unified utility maximization model formulates the resource selection decision in frameworks with different goals for different federated search applications. Solutions have been derived to obtain desired resource selection decisions for different applications.

An extensive set of experiments with different federated search environments has been conducted to show the advantage of this unified utility maximization framework. Particularly, for information source recommendation, the new model has been shown to obtain more robust resource selection results for the high-recall goal than the CORI and ReDDE algorithm. For federated document retrieval, the unified utility maximization model optimized for the high-precision goal outperformed several resource selection methods optimized for the high-recall goal. All of these results explicitly demonstrate the power of the unified utility maximization as an integrated solution for different federated search applications.

The utility maximization framework can be extended in several directions. First, this chapter studies two goals of high recall (i.e., Equation 7.9) and high precision (i.e., Equation 7.12) in two resource selection algorithms. It is natural to combine the two goals linearly into a single objective function (e.g., Equation 7.9 plus lambda times Equation 7.12) within one framework. The additional parameter of lambda is introduced to balance the preference for these two criteria. Second, the utility maximization method still needs a small amount of human relevance judgments as training data. This requirement might be relaxed by utilizing implicit user feedback as relevance judgments for returned documents. These documents can be downloaded and their centralized document scores can be obtained for training the logistic model for estimating probabilities of relevance.

One big advantage of the unified utility maximization framework is that it considers a set of evidence besides relevance. Multiple factors can be naturally incorporated into the unified framework for maximizing utilities for different federated search applications. One particular example is to consider search engine retrieval effectiveness. This is the main topic of Chapter 8.

Chapter 8: Modeling Search Engine Retrieval Effectiveness

Chapter 7 proposes a unified utility maximization framework to integrate effective solutions of the main subproblems of federated search into a single framework and optimize various goals of different federated search applications. The new framework provides a theoretically solid view of federated search algorithms and is open for many extensions. However, it makes an assumption, like most previous research, that all the search engines of available information sources are effective to return their relevant documents. When such an assumption is not valid, it further claims that the semi-supervised learning results merging algorithm can be used to detect and remove irrelevant documents returned by ineffective search engines so that those irrelevant documents do not hurt the final ranked list. However, the problem is that the failure to consider the retrieval effectiveness of search engines has already damaged the overall accuracy in the resource selection phase. In contrast, a better resource selection method can provide more accurate results by choosing information sources that can return more relevant documents.

This chapter proposes a federated search technique that extends the utility maximization framework to model the retrieval effectiveness of search engines in a federated search environment. The new algorithm ranks available information sources by estimating the number of relevant documents they can *return*, instead of the amount of relevant documents they contain.

Empirical studies were conducted on several testbeds with different characteristics to show the advantage of the new research over prior research that did not consider retrieval effectiveness of search engines. The results show that the new algorithm provides more accurate results than the previous state-of-the-art algorithms when some search engines are not effective, and is at least as effective as prior solutions when all the search engines are effective.

The next section provides the motivation of the research work in this chapter. Section 8.2 briefly discusses related prior research. Section 8.3 describes our new approach to model search engine effectiveness with a variant of the unified utility maximization framework as the returned utility maximization framework. Section 8.4 explains our experimental methodology and presents the experimental results.

8.1 Motivation of modeling search engine effectiveness

A practical problem for federated search solutions in uncooperative environments is that the search engines of some information sources may not be effective at returning their relevant documents. The search engines may use ineffective retrieval methods like the Boolean retrieval algorithm [Baeza-Yates & Ribeiro-Neto, 1999]. They may choose effective retrieval algorithms but have had trouble tuning the algorithms on various corpora to obtain good parameters for the retrieval algorithms (e.g., the smoothing parameter in the language model retrieval algorithms [Zhai & Lafferty, 2001]). Even when the search engines use effective retrieval algorithms with good parameters at the early stages of system development, the effects of those parameters may degrade as the systems evolve by adding documents with diverse characteristics.

Hidden information sources with ineffective search engines are very common in real world federated search applications. The well-known PubMed search engine uses exact match Boolean retrieval algorithm and orders documents by when they were added to the database instead of by relevance. Our experience with the FedStats portal shows examples of information sources with search engines that return unranked or randomly ranked results, or return many documents that do not exist, as in the case of broken links [Avrahami et al., 2006].

Simply ignoring the search engine effectiveness factor can cause a serious problem as the selected resources may not return as many relevant documents as expected because of the ineffectiveness of their search engines.

8.2 Returned utility maximization method

A new algorithm is proposed in this section to incorporate the factor of retrieval effectiveness of search engines into the unified utility maximization framework for the federated document retrieval application. Our new algorithm ranks information sources by the criterion as the number of relevant documents they *return* instead of the criterion as the number of relevant documents they *contain*. The new framework is called *returned utility maximization framework*. It supports both the evidence of relevance and search engine retrieval effectiveness. Specifically, it measures the effectiveness of search engines by: i) sending a small number of training queries to retrieve documents from available resources; and ii) investigating the consistency between the ranked lists of individual search engines and the lists generated by an effective centralized retrieval algorithm on the same set of returned documents. The accuracy of how each search engine ranks its documents can be learned from these steps. Then, in the resource selection phrase, information sources are ranked by considering both the factor of how many relevant documents each source may contain and the factor of how effectively each search engine has ranked its returned documents in the past. This is accomplished by formalizing the resource selection procedure as a problem to maximize the amount of relevant documents to be returned.

This section first presents our method to measure the retrieval effectiveness of search engines; and then shows the framework of the returned utility maximization method for resource selection.

8.2.1 Measuring the retrieval effectiveness of search engines

It is a difficult problem to measure the retrieval effectiveness of search engines in uncooperative federated search environments as very limited information can be directly obtained from available information sources. One possibility is to estimate the parameters of ranking functions when these functions can be assumed to have special forms [Liu et al., 2001]. But this assumption is rarely true in real world applications and it is not discussed here. The unified utility maximization in Chapter 7 utilizes a small amount of hand-labeled training data to learn a model for estimating probabilities of relevance. However, that model only addresses the problem of how many documents are relevant within an information source instead of whether the information source can return those relevant documents.

In this work, a new method is proposed to measure the retrieval effectiveness of search engines in uncooperative environments. It utilizes an effective centralized retrieval algorithm that works on the centralized sample database, which provides a standard for measuring the effectiveness of search engines. The basic idea of this method is to investigate how consistent the ranked lists returned from individual search engines are with the lists generated by the centralized retrieval algorithm on the same set of documents. The intuition is that if a search engine tends to generate consistent ranked lists with those of a centralized retrieval algorithm, this search engine is likely to be effective. Otherwise, it may not be effective. This strategy follows the trend in this dissertation for utilizing the centralized sample database for different federated search applications.

An alternative approach for measuring the retrieval effectiveness of search engines is to evaluate their returned results with human relevance judgments. However, this requires relevance judgment data to evaluate the results from each search engine, which has excessive cost when there are many information sources in the federated search environment. On the other hand, our method uses the results of an effective centralized retrieval algorithm with corpus statistics from centralized sample database as a surrogate for the human relevant judgment data to evaluate the results from each search engine. This is much more efficient as the results by the centralized retrieval algorithm can be generated automatically without requiring human efforts. Similar ideas of applying a centralized retrieval algorithm with corpus statistics from centralized sample database have been successfully utilized for other federated search subproblems in Chapter 4 for resource selection [Si & Callan, 2004b] and Chapter 5 for results merging [Si & Callan, 2003b].

Specifically, the retrieval effectiveness profiles of search engines are built during the resource representation phase. Again, the centralized sample database is first constructed by all sampled documents from query-based sampling. A small set of training queries (e.g., 50 queries) is sent to search all available information sources[23]. For this set of training queries, no human relevant judgments are required and the results automatically generated from centralized retrieval algorithms are used instead. There are many possible choices for selecting the training queries. Our approach is to utilize the TREC queries to simulate real world user queries, which cover multiple topics. For each ranked list from one information source, some representative documents (e.g., every fifth document in the top 250 documents) are downloaded. The purpose of downloading only representative documents (e.g., one sample document for the block size of 5) is to reduce the communication cost. Furthermore, a centralized retrieval algorithm (i.e., INQUERY) is applied on these documents with the corpus statistics from the centralized sample database. Then two ranked lists as the source-specific ranked list and the ranked list by the centralized retrieval algorithm can be acquired for each training query for each search engine. A mapping function can be learned based on the two ranked lists, which transforms the document rank in the source-specific ranked list to the document rank in the list generated by the centralized retrieval algorithm as follows:

$$\varphi_{ij}: \quad d_{db\_i}(r_1) \rightarrow d_c(r_2) \tag{8.1}$$

where φ_{ij} is the ranked list transformation of the jth training query for the ith information source, $d_{db\_i}(r_1)$ represents the r_1^{th} document in the source-specific ranked list from the ith information source and $d_c(r_2)$ represents the r_2^{th} document in the ranked list generated by centralized the retrieval algorithm. Equation 8.1 indicates that the r_1^{th} document in the ranked list of the jth training query from the ith resource is mapped to the r_2^{th} document in the corresponding ranked list generated by the centralized retrieval algorithm. If the single document representing a block of nearby documents in the source-specific ranked list is mapped to a

[23] It is possible to use different sets of queries for different sources. However, the same set of queries is utilized here to reduce the variance caused by the different characteristics of different sets of queries.

particular position in the list by the centralized retrieval algorithm, all the documents in this block are mapped to similar ranks around the image of the representative document.

The retrieval effectiveness profile for a search engine is built by collecting the learned transformations for all training queries. Formally, for the search engine of the ith resource, the profile is $\{\varphi_{ij}, 1 \le j \le J\}$, where there are altogether J transformations learned for the search engine. The profile can be represented as an array with J rows of permutations that indicate the rank transformations. All the profiles for available search engines represent the knowledge of how effectively each search engine has ranked its returned documents for the training queries. Note that all the profiles contain J transformations as all of them are built on the same set of training queries. More training queries may generate more accurate search engine retrieval effectiveness profiles or may enable topic-specific effectiveness profiles. These are interesting topics for future research.

This section describes the approach to model search engine effectiveness by learning the rank to rank transforms between source-specific ranked lists and the ranked lists by an effective centralized retrieval algorithm. This may be a more appropriate approach than learning score to rank or score to score transforms as the documents returned from different queries generally have different types of score distributions and it is hard to compare the distributions across multiple queries.

8.2.2 Returned utility maximization method

A federated document retrieval system automatically searches selected information sources and merges returned ranked lists into a final list to present to the end user. Users' information needs are maximally satisfied when there are as many relevant documents in the final ranked lists as possible. Therefore, the utility for a federated document retrieval system should be the number of relevant documents *returned* from selected information sources. Formally, let d_i denote the number of documents to be retrieved from the ith information source and $\vec{d}=\{d_1,d_2,.....\}$ as the resource selection decision for all information sources. The returned utility of a particular resource selection decision is calculated as:

$$\sum_{i} I(db_i) \sum_{k=1}^{d_i} R_i(\hat{d}_{db}(r)) \tag{8.2}$$

where $I(db_i)$ is an indicator function, which is 1 when the ith source is selected and 0 otherwise. $R_i(\hat{d}_{db}(r))$ is the probability of relevance of the top rth document in the source-specific ranked list returned from the information source. This formula is related to Equation 7.12 for unified utility maximization framework. The key difference is that the new framework explicitly estimates utilities that can be *returned* from source-specific ranked lists via $R_i(\hat{d}_{db}(r))$, while the unified utility maximization framework considers utilities that are contained in available sources.

In order to calculate the returned utility represented by Equation 8.2, the probabilities of relevance for documents at the top of source-specific ranked lists should be provided, which are not known. On the other hand, Section 7.2.1 describes a method to estimate the probabilities of relevance for documents in the ranked list (i.e., $\{R_i(d_C(k)),\ k \in [1, \hat{N}_{db_i}]\}$), which is generated by the centralized retrieval algorithm with corpus statistics from the centralized sample database. Therefore, there is a gap between the source-specific ranked lists and the ranked list generated by centralized retrieval algorithm.

To bridge the gap, we utilize the retrieval effectiveness profile built for each search engine, which is described in Section 8.2.1. The top ranked documents within source-specific ranked lists can be mapped to their corresponding ranks by the centralized retrieval algorithm. Specifically, for a user query, for a document that ranks at the r^{th} position in source-specific ranked list, its rank by the centralized retrieval algorithm can be seen as a weighted average of the ranks by the centralized retrieval algorithm of documents in all training queries, which also rank at the r^{th} position in source-specific ranked lists. We assume that all training queries have equal weights (i.e. 1/J). This process can be conducted for all returned documents in available information sources, and thus the returned utility can be estimated as:

$$\sum_i I(db_i) \sum_{j=1}^{J} \frac{1}{J} \sum_{r=1}^{d_i} R_i\,(\varphi_{ij}(\,d_{db\_i}(r))\,) \qquad (8.3)$$

Furthermore, only a small number (i.e., N_{sdb}) of information sources should be selected to retrieve a fixed number (i.e., N_{rdoc}) of documents. Finally, the resource selection problem for maximizing returned utility can be represented as:

$$\begin{aligned} \overrightarrow{d}^{*} = \underset{\vec{d}}{\arg\max} \sum_i I(db_i) \sum_{j=1}^{J} \frac{1}{J} \sum_{r=1}^{d_i} R_i\,(\varphi_{ij}(d_{db\_i}(r)) \\ \text{subject to:} \sum_i I(db_i) = N_{sdb} \\ d_i = N_{rdoc}, \text{if } d_i \neq 0 \end{aligned} \qquad (8.4)$$

As the calculation of the returned utilities from available information sources is not coupled with each other, the solution of the above returned utility maximization problem is simple. The expected returned utility from each information source can be calculated as follows:

$$\hat{R}U_i = \sum_{j=1}^{J} \frac{1}{J} \sum_{r=1}^{d_i} R_i\,(\varphi_{ij}(d_{db\_i}(r)) \qquad (8.5)$$

The resource selection decision is made by selecting a few information sources that contribute the largest amount of returned utilities. The selected information sources are searched and finally the returned results are merged into a single ranked list by the semi-supervised learning algorithm.

An alternative method to consider search engine effectiveness is to calculate the correlation coefficient (e.g., Spearman Rank Correlation Coefficient [Press et al., 1992]) between source-specific ranked lists and the ranked lists produced by an effective centralized retrieval algorithm. The correlation can be used as a weight factor in the utility maximization resource selection framework (e.g., combined linearly with a weight in the objective function in Equation 7.14). If the combination weight can be set appropriately, a resource selection decision can be obtained to reach a good trade-off between the evidence of relevance and the evidence of search engine effectiveness. This approach is not adopted in this dissertation because we integrate the search engine effectiveness directly to estimate the number of relevant documents that can be returned by Equation 8.4.

8.3 Evaluation methodology and experimental results

This section presents experimental results to demonstrate the advantage of the returned utility maximization framework. It first introduces the evaluation methodology by creating federated search environments with search engines of different qualities. Furthermore, an extensive set of experiments is conducted to compare the returned utility maximization framework with several other alternatives.

8.3.1 Evaluation methodology

Three testbeds were used in this chapter, namely the Trec123_100Col testbed, the representative testbed and the WT10g testbed. Detailed information about these testbeds can be found in Chapter 2. Trec123_100Col and representative testbeds contain news/government data. On these two testbeds, 50 TREC title queries (101-150) were used as training queries and another 50 queries (51-100) were used as test queries. The arrangement was made to be consistent with the results reported in much prior research [Callan, 2000; Si & Callan, 2003b; Nottelmann & Fuhr, 2003b; Si & Callan, 2004b] and the experiments in Chapter 5 and Chapter 6. WT10g contains Web data. TREC Web queries 451-550 were used on the WT10g testbed. The

Table 8.1: Precision of six search engines on the Trec123_100Col testbed when 5 sources were selected.

Precision at Doc Rank	INQUERY	LM	SMART	INQUERY RAND	LM Bad Parameter	Extended Boolean
5 docs	0.4460	0.4920	0.4400	0.3200	0.3000	0.2600
10 docs	0.4340	0.4520	0.4080	0.2900	0.2720	0.2440
15 docs	0.4053	0.4040	0.3733	0.2680	0.2560	0.2320
20 docs	0.3750	0.3800	0.3590	0.2480	0.2540	0.2170
30 docs	0.3400	0.3527	0.3333	0.2240	0.2373	0.1920

Table 8.2: Precision of six search engines on the WT10g testbed when 20 sources were selected.

Precision at Doc Rank	INQUERY	LM	SMART	INQUERY RAND	LM Bad Parameter	Extended Boolean
5 docs	0.2247	0.2144	0.1918	0.1299	0.1155	0.1196
10 docs	0.1938	0.1814	0.1825	0.1206	0.1247	0.1093
15 docs	0.1718	0.1684	0.1636	0.1031	0.1134	0.0942
20 docs	0.1536	0.1572	0.1546	0.0918	0.1036	0.0871
30 docs	0.1271	0.1278	0.1316	0.0746	0.0849	0.0780

first set of fifty queries and the second set of fifty queries were used as training and test data alternatively.

Six types of search engines were used in the experiments to reflect the characteristics of uncooperative environments: three types of effective search engines and three types of ineffective retrieval algorithms were used. The three effective retrieval algorithms are INQUERY [Turtle 1990; Callan, Croft & Harding, 1992], language model with linear smoothing (the smooth parameter is set to be 0.5) [Lafferty & Zhai, 2001; Ogilvie & Callan, 2001] and a TFIDF retrieval algorithm with the "lnc.ltc" weighting [Buckley et al., 1995]. The three ineffective retrieval algorithms are: an extended Boolean retrieval algorithm, which adds up the term frequencies of matched query terms without considering the idf factor; a language model method with bad linear smoothing parameter, which is set to be 0.99 (bias towards the corpus language model); and an INQUERY retrieval algorithm with random noise added to the original retrieval scores, where the random noise ranges from 0 to 0.3 (the original scores range from 0.4 to 1). More detailed information about the retrieval algorithms is in Chapter 2.

Two sets of federated search experiments were conducted on the Trec123_100Col and the WT10g testbeds to show the effectiveness of the six retrieval algorithms. All the search engines were assigned a single type of retrieval algorithm. The UUM algorithm was used to select 5 sources and 20 sources on the Trec123_100Col and WT10g testbeds respectively. The results are shown in Tables 8.1 and 8.2. It can be seen that the three

Table 8.3: Precision on the Trec123_100Col testbed when 3 or 5 sources were selected. All search engines were assigned effective retrieval algorithms.

Document Rank	**3 Sources Selected**			**5 Sources Selected**		
	UUM	**CORI**	**RUM**	**UUM**	**CORI**	**RUM**
5	0.4640	0.3640 (-21.6%)	0.4720 (+8.8%)	0.4720	0.4000 (-5.3%)	0.4720 (+8.8%)
10	0.4240	0.3340 (-21.2%)	0.4160 (+4.8%)	0.4260	0.3800 (-10.8%)	0.4300(+4.8%)
15	0.3933	0.3253 (-17.3%)	0.3947 (+2.9%)	0.3960	0.3560 (-10.1%)	0.4120 (+2.9%)
20	0.3730	0.3120 (-16.4%)	0.3690 (+4.1%)	0.3740	0.3430 (-8.3%)	0.3890 (+4.1%)
30	0.3413	0.2780 (-18.6%)	0.3367 (+6.9%)	0.3520	0.3227 (-8.3%)	0.3647 (+6.9%)

Table 8.4: Precision on the representative testbed when 3 or 5 sources were selected. All search engines were assigned effective retrieval algorithms.

Document Rank	**3 Sources Selected**			**5 Sources Selected**		
	UUM	**CORI**	**RUM**	**UUM**	**CORI**	**RUM**
5	0.4560	0.3720 (-18.4%)	0.4600 (+1.0%)	0.4280	0.3960 (-7.0%)	0.4400 (+2.8%)
10	0.4540	0.3400 (-25.1%)	0.4640 (+2.2%)	0.4460	0.3900 (-12.6%)	0.4520 (+1.3%)
15	0.4240	0.3140 (-25.9%)	0.4493 (+6.0%)	0.4440	0.3533 (-20.4%)	0.4440 (+0.0%)
20	0.4050	0.3020 (-25.4%)	0.4260 (+5.2%)	0.4310	0.3340 (-22.5%)	0.4280 (-0.7%)
30	0.3747	0.2533 (-32.5%)	0.3900 (+4.1%)	0.3993	0.2967 (-25.7%)	0.3973 (-0.5%)

Table 8.5: Precision on the WT10g testbed when 10 or 20 sources were selected. All search engines were assigned effective retrieval algorithms.

Document Rank	**10 Sources Selected**			**20 Sources Selected**		
	UUM	**CORI**	**RUM**	**UUM**	**CORI**	**RUM**
5	0.2082	0.1583 (-24.0%)	0.2289 (+9.9%)	0.2000	0.1812 (-9.0%)	0.2201 (+10.1%)
10	0.1763	0.1323 (-25.0%)	0.1814 (+2.9%)	0.1763	0.1427 (-19.1%)	0.1901 (+7.8%)
15	0.1464	0.1132(-22.7%)	0.1560 (+6.6%)	0.1546	0.1264 (-18.2%)	0.1601 (+4.3%)
20	0.1314	0.1021 (-22.3%)	0.1397 (+6.3%)	0.1428	0.1104 (-22.7%)	0.1475 (+3.3%)
30	0.1107	0.0833 (-24.8%)	0.1137 (+2.7%)	0.1223	0.0913 (-25.4%)	0.1286 (+5.2%)

effective retrieval algorithms acquire much more accurate results than the three ineffective algorithms. The extended Boolean retrieval algorithm seems to be even less effective than the other two ineffective algorithms.

8.3.2 Retrieval results with ineffective search engines

The experiments in this section answer the question: How does the new returned utility maximization (RUM) method work in the federated search environments where exist different proportions of ineffective search engines? Experiments were conducted to compare the RUM method with the CORI and unified utility maximization (UUM) methods that do not consider retrieval effectiveness of search engines.

The first set of experiments was conducted when all available information sources on the three testbeds use three types of effective retrieval algorithms (i.e., INQUERY, LM and SMART). The three types of search engines were assigned to the information sources on different testbeds in a round-robin manner. The experimental setting is similar as that in prior research [Si & Callan, 2004b]. The results are shown in Tables 8.3-8.5. It can be seen that the CORI algorithm was not as effective as the UUM algorithm, which is consistent with previous research [Si & Callan, 2004b]. This demonstrates the fact that when the search engines of information sources are effective to return their relevant documents, the UUM resource selection algorithm has an advantage over the CORI algorithm. The results in Tables 8.3-8.5 also show that the RUM algorithm is at least as good as the UUM algorithm in all of the configurations of the experiments. This indicates the effectiveness of the RUM method when all of the search engines in federated search environments are of high quality.

As many information sources in real world federated search applications use ineffective engines, more experiments were conducted to simulate federated search environments where some search engines are of low qualities. In the second set of experiments, one third of the information sources on the three testbeds of Trec123_100Col, representative and WT10g were assigned ineffective search engines in a round-robin manner. Specifically, one large information source of the representative testbed was assigned the extended Boolean retrieval algorithm, while the other large source was assigned the SMART search engine (more detail is in Chapter 2). The results on these three testbeds are shown in Tables 8.6, 8.7 and 8.8 respectively.

The experimental results indicate that the RUM method outperformed the UUM method in all configurations, which is consistent with our expectation that RUM explicitly models search engine retrieval effectiveness and thus can select information sources that return more relevant documents. Particularly, it can be seen from Table 8.6 that the RUM method acquired a substantial improvement over the UUM method on the Trec123_100Col testbed. For the representative testbed, the results in Table 8.7 show that the RUM method was better than the UUM method. However, the advantage of the RUM method on the representative testbed was smaller than that on the Trec123_100Col testbed when relatively more information sources (i.e., 5) were selected. Detailed analysis indicates that as the two large information sources on the representative testbed contain more relevant documents than small sources, the RUM method favors the two large sources more than the other small sources, which is similar to the behavior of the UUM method. Therefore, when relatively more information sources (e.g., 5) were selected, the overlap in the sources selected by RUM and UUM was getting larger and thus these two methods tend to generate similar results.

Table 8.6: Precision on the Trec123_100Col testbed when 3 or 5 sources were selected. One third of the search engines were assigned ineffective retrieval algorithms.

Document Rank	3 Sources Selected			5 Sources Selected		
	UUM	**CORI**	**RUM**	**UUM**	**CORI**	**RUM**
5	0.3640	0.3200 (-12.1%)	0.4160 (+14.3%)	0.3800	0.3400 (-10.5%)	0.4400 (+15.8%)
10	0.3260	0.3000 (-8.0%)	0.3840 (+17.8%)	0.3400	0.3200 (-5.9%)	0.3860 (+13.5%)
15	0.3133	0.2760 (-11.9%)	0.3587 (+14.5%)	0.3173	0.3067 (-3.3%)	0.3680 (+16.0%)
20	0.2960	0.2630 (-11.2%)	0.3380 (+14.2%)	0.3050	0.2990 (-2.0%)	0.3520 (+15.4%)
30	0.2727	0.2327 (-14.7%)	0.2993 (+9.8%)	0.2840	0.2733 (-3.8%)	0.3240 (+14.1%)

Table 8.7: Precision on the representative testbed when 3 or 5 sources were selected. One third of the search engines were assigned ineffective retrieval algorithms.

Document Rank	3 Sources Selected			5 Sources Selected		
	UUM	**CORI**	**RUM**	**UUM**	**CORI**	**RUM**
5	0.3840	0.3440 (-10.4%)	0.4320 (+12.5%)	0.3880	0.3400 (-2.4%)	0.4280 (+10.3%)
10	0.3260	0.2940 (-9.8%)	0.3760 (+15.3%)	0.3580	0.3280 (-8.4%)	0.3880 (+8.4%)
15	0.3027	0.2787 (-7.9%)	0.3333 (+10.1%)	0.3373	0.3240 (-3.9%)	0.3600 (+6.7%)
20	0.2800	0.2620 (-6.4%)	0.3150 (+12.5%)	0.3150	0.3030 (-3.8%)	0.3280 (+4.3%)
30	0.2413	0.2253 (-6.6%)	0.2693 (+11.6%)	0.2867	0.2827 (-1.4%)	0.2867 (+0.0%)

Table 8.8: Precision on the WT10g testbed when 10 or 20 sources were selected. One third of the search engines were assigned ineffective retrieval algorithms.

Document Rank	10 Sources Selected			20 Sources Selected		
	UUM	**CORI**	**RUM**	**UUM**	**CORI**	**RUM**
5	0.1691	0.1381 (-18.3%)	0.2000 (+18.3%)	0.1753	0.1649 (-5.9%)	0.1918 (+9.3%)
10	0.1443	0.1082 (-25.0%)	0.1567 (+8.6%)	0.1598	0.1371 (-14.2%)	0.1784 (+11.6%)
15	0.1203	0.0955 (-20.6%)	0.1278 (+6.2%)	0.1402	0.1127 (-19.6%)	0.1505 (+7.4%)
20	0.1093	0.0861 (-21.2%)	0.1129 (+3.3%)	0.1211	0.0974 (-19.6%)	0.1320 (+9.0%)
30	0.0907	0.0698 (-23.4%)	0.0983 (+8.4%)	0.1041	0.0804 (-22.8%)	0.1117 (+7.2%)

The results on the WT10g testbed are shown in Table 8.8. Again, the RUM method generates more accurate results than the UUM method. More careful observation by comparing Tables 8.6 and 8.8 indicates that on average the advantage of the RUM method over UUM is lower on the WT10g testbed than that on the Trec123_100Col testbed. This can be explained by the different characteristics of the two testbeds. WT10g contains many more small information sources (e.g., 625 out of 934 resources contain less than 1,000 documents) than Trec123_100Col (e.g., 3 out of 100 resources contain less than 1,000 documents). Small sources tend to return shorter ranked lists than large sources. This makes the method of measuring search engine retrieval effectiveness by investigating rank consistency less effective.

Table 8.9: Precision on the Trec123_100Col testbed when 3 or 5 sources were selected. Two thirds of the search engines were assigned ineffective retrieval algorithms.

Document Rank	3 Sources Selected			5 Sources Selected		
	UUM	CORI	RUM	UUM	CORI	RUM
5	0.3240	0.2920 (-9.9%)	0.3880 (+19.8%)	0.3400	0.3120 (-8.2%)	0.4240 (+24.7%)
10	0.2840	0.2760 (-2.8%)	0.3480 (+22.5%)	0.3040	0.3120 (-2.6%)	0.3740 (+23.0%)
15	0.2600	0.2520 (-3.1%)	0.3293 (+26.7%)	0.2893	0.2880 (-0.5%)	0.3413 (+18.0%)
20	0.2470	0.2290 (-7.3%)	0.3130 (+26.7%)	0.2680	0.2680 (+0.0%)	0.3170 (+18.2%)
30	0.2160	0.2040 (-5.6%)	0.2687 (+24.4%)	0.2413	0.2467 (+2.2%)	0.2780 (+15.2%)

Table 8.10: Precision on the representative testbed when 3 or 5 sources were selected. Two thirds of the search engines were assigned ineffective retrieval algorithms.

Document Rank	3 Sources Selected			5 Sources Selected		
	UUM	CORI	RUM	UUM	CORI	RUM
5	0.3320	0.3160 (-4.8%)	0.3880 (+16.9%)	0.3720	0.3240 (-2.9%)	0.4160 (+11.8%)
10	0.2840	0.2880 (-1.4%)	0.3440 (+21.1%)	0.3360	0.3040 (-9.5%)	0.3680 (+9.5%)
15	0.2520	0.2427 (-3.7%)	0.3240 (+28.6%)	0.3053	0.2867 (-6.1%)	0.3360 (+10.0%)
20	0.2330	0.2230 (-4.3%)	0.3020 (+29.6%)	0.2690	0.2600 (-3.4%)	0.3100 (+15.2%)
30	0.2007	0.1900 (-5.3%)	0.2527 (+26.3%)	0.2320	0.2300 (-0.9%)	0.2727 (+17.5%)

Table 8.11: Precision on the WT10g testbed when 10 or 20 sources were selected. Two thirds of the search engines were assigned ineffective retrieval algorithms.

Document Rank	10 Sources Selected			20 Sources Selected		
	UUM	CORI	RUM	UUM	CORI	RUM
5	0.1443	0.1175 (-18.6%)	0.1649 (+14.3%)	0.1464	0.1320 (-9.8%)	0.1629 (+11.3%)
10	0.1289	0.1041 (-19.2%)	0.1464 (+13.6%)	0.1216	0.1082(-11.0%)	0.1361 (+11.9%)
15	0.1038	0.0893 (-14.0%)	0.1162 (+11.9%)	0.1107	0.1003 (-9.4%)	0.1223 (+10.5%)
20	0.0897	0.0799(-10.9%)	0.1026 (+14.9%)	0.0990	0.0871(-12.0%)	0.1129 (+14.0%)
30	0.0753	0.0643 (-14.6%)	0.0835 (+10.9%)	0.0883	0.0739(-16.3%)	0.0959 (+8.6%)

Specifically, the RUM method ranks information sources by their estimated returned utilities as described in Equation 8.5. To calculate the returned utility of a top ranked document in each information source, the estimated centralized ranks are first obtained from the rank patterns of training queries; then the utility of this document is calculated by averaging the probabilities of relevance of documents with the estimated centralized ranks. Small size sources return short ranked lists and the rank transformations often map top ranked documents in the sources to high centralized ranks. Therefore, the rank transformation may be less effective in estimating the retrieval effectiveness of search engines for small sources. However, in real world applications, most information sources contain reasonably large amount of documents (e.g., several thousands), as observed in the FedLemur project. These federated search environments are more like the Trec123_100Col testbed in this aspect and the RUM method can be expected to be effective. It is also possible to design a variant of the returned utility maximization model that associates different weights with the documents at the top of the ranked lists. This approach may better distinguish the retrieval effectiveness of search engines for small sources and is an interesting future research topic.

In the third set of experiments, two thirds of the search engines were assigned ineffective retrieval algorithms. The results are shown in Tables 8.9-8.11. It can be observed that the RUM method outperformed the UUM method substantially on all the three testbeds. The advantage of the RUM method over the UUM method is larger than that when none or one third search engines were ineffective, which is consistent with our expectation that the power of the RUM method of detecting ineffective search engines is more substantial in federated search environments with more ineffective search engines.

8.3.3 Which search engines are selected?

The experiments in Sections 8.3.2 demonstrate the advantage of the RUM method in the federated search environments where there exist different amounts of ineffective search engines. To provide more insight of the behavior of the RUM method, a set of experiments was conducted to investigate the percentage of selected information sources with different retrieval algorithms.

The experiments were conducted on the Trec123_100Col and the WT10g testbeds, where one third of the search engines were assigned ineffective retrieval algorithms and 5 or 20 information sources were selected respectively. The results are shown in Tables 8.12 and 8.13. The effective search engines were assigned to twice as many information sources as ineffective search engines. The UUM resource selection method does not consider search engines effectiveness. Therefore, the UUM algorithm selected effective search engines roughly twice as frequent as ineffective search engines (not exactly due to data variance). In contrast, the RUM method really favored effective search engines and disfavored ineffective search engines, which is consistent with our expectation. The resource selection choice made by the RUM method considers both relevance information and search engine retrieval effectiveness, so some ineffective search engines are still selected on both the Trec123_100Col and the WT10g testbeds because they contain more relevant information. Note that the difference between RUM and UUM method is smaller on the WT10g testbed than that on the TREC123_100Col testbed, which is consistent with the discussion in Section 8.3.2.

Table 8.12: The percentage of selected resources using different retrieval algorithms on Trec123_100Col testbed. 5 sources were selected. One third of the search engines were assigned ineffective retrieval algorithms. Therefore, INQUERY, LM and SMART have been assigned to about 22% sources respectively, while INQUERY_RAND, LM with Bad Parameters and Extended Boolean have been assigned to about 11% sources respectively.

Algorithms	**INQUERY**	**LM**	**SMART**	**INQUERY RAND**	**LM Bad Parameter**	**Extended Boolean**
UUM	17.6%	24.9%	23.3%	10.2%	10.2%	13.9%
RUM	24.9%	35.9%	26.1%	5.7%	4.9%	2.4%

Table 8.13: The Percentage of selected resources using different retrieval algorithms on WT10g testbed. 20 sources were selected. One third of the search engines were assigned ineffective retrieval algorithms. Therefore, INQUERY, LM and SMART have been assigned to about 22% sources respectively, while INQUERY_RAND, LM with Bad Parameters and Extended Boolean have been assigned to about 11% sources respectively.

Algorithms	**INQUERY**	**LM**	**SMART**	**INQUERY RAND**	**LM Bad Parameter**	**Extended Boolean**
UUM	19.4%	23.9%	21.1%	13.8%	11.7%	10.3%
RUM	22.5%	27.6%	23.0%	10.5%	9.0%	7.5%

8.4 Summary

One important issue for real world federated search applications in uncooperative environments is that some information sources may be associated with ineffective search engines and are not effective at returning their relevant documents. Our experience with the FedStats portal suggests that many U.S. government information sources on the Web use ineffective search engines that return unranked or randomly ranked results, or return many documents that do not exist, as in the case of broken links.

Most previous resource selection algorithms do not consider the effectiveness of search engines. This may cause a serious problem. If a selected information source has a bad search engine, this information source may not return as many relevant documents as expected by the resource selection algorithm, so the overall quality of the final ranked list is degraded.

A new algorithm is proposed in this chapter to extend the unified utility maximization framework for considering the retrieval effectiveness of search engines in uncooperative federated search environments. The new algorithm sorts information sources by estimating how many relevant documents they return instead of how many relevant documents they contain. The new framework is called the returned utility maximization framework. It measures the effectiveness of a particular search engine by investigating the consistency between the ranked lists of the search engine and the corresponding lists generated by an effective centralized retrieval algorithm on the same sets of documents. The retrieval effectiveness profiles of available search engines can be built by this approach. During the resource selection phase, the returned utility maximization framework incorporates the effectiveness profiles with the evidence of relevance to select information sources that can return the largest amount of relevant documents.

Empirical studies have been conducted on a range of testbeds with different characteristics to show the advantage of the new research. Particularly, the results suggest that returned utility maximization method provides more accurate results than prior methods that do not consider the factor of search engine effectiveness when some search engines are not effective. More detailed analysis shows that the power of the return utility maximization method comes from selecting more effective search engines and avoiding ineffective search engines.

This chapter presents a specific example to show the power and flexibility of the unified utility maximization framework for considering other search engine characteristics, such as retrieval effectiveness. Besides this specific example, the unified framework opens a door for considering a set of non relevance-based evidence such as search engine delay or information authority.

In the current framework of the returned utility maximization, all training queries are associated with equal weights. A possibly better approach is to assign training queries with various weights according to their similarities with a user text query. This method offers the possibility of distinguishing query-specific/topic-specific retrieval effectiveness of individual search engines. It is an interesting future research topic.

Furthermore, the returned utility maximization framework in this chapter uses a single centralized retrieval algorithm to measure the effectiveness of source-specific search engines. However, this may introduce implicit bias towards retrieval algorithms that share similar behavior as the centralized retrieval algorithm (e.g., favor short documents). It is an interesting topic in future research to use a set of effective centralized retrieval algorithms to reduce the possible bias.

Chapter 9: Conclusion and Future Work

Conventional search engines like Google or MSN have provided effective search solutions for some information on the Web that can be easily acquired by crawling Web links. These conventional search engines copy Web pages into a single centralized database, index the contents, and make them searchable. However, a large amount of valuable information cannot be copied into a single centralized database due to reasons such as intellectual property protection and frequent information update. The information is hidden behind source-specific search interfaces. There are multiple types of hidden information such as unstructured data behind text search interfaces and semi-structured or structured data behind more complicated search interfaces. The information sources that contain this type of hidden information exist on the Web as well as in enterprise search environments, and they are often created and maintained by domain experts.

Federated search has been proposed as the search solution for the valuable hidden information. It provides a unified search interface that connects the source-specific search engines of available information sources that contain hidden information. This dissertation focuses on the research problems of federated search applications for text data behind simple text search engines within uncooperative environments where only most rudimentary cooperation can be assumed from available information sources.

This chapter concludes the dissertation by summarizing the contributions and then proposes several directions for future research. Specifically, it first summarizes a range of new algorithms proposed for different main research problems of federated search. Section 9.2 summaries the general contributions of the dissertation that advance the state-of-the-art of federated search. Finally, several possible future topics are discussed to extend the research in this dissertation.

9.1 Summarization of dissertation results

There are three main research problems in federated search. First, the contents as well as many other properties of each information source should be acquired (resource representation). Second, given a user query, a decision should be made about which sources to search (resource selection). Third, the results returned from all selected sources may be integrated into a single final list (results merging). This dissertation proposes a full range of algorithms for all three main research problems. Furthermore, a unified utility maximization framework is proposed to integrate and adjust the individual solutions into a single framework for different federated search applications.

Information source size (i.e., number of documents) is an important type of resource representation for different federated search applications like resource selection. However, most previous federated search algorithms did not consider source sizes because it was difficult to obtain reasonably good source size estimates efficiently. This dissertation proposes a Sample-Resample algorithm to effectively and efficiently acquire source size estimates. It views the sampled documents in the centralized sample database as a set of representative documents of those in the complete information sources. The Sample-Resample method analyzes the statistics of the document frequencies of a set of resample queries in both the centralized sample database and the complete information sources to acquire the source size estimates. Different variants of the Sample-Resample method have been proposed to utilize different types of support from individual search engines. This new method has been shown to be more effective and efficient than the alternative Capture-Recapture method for different sets of information sources. It is not perfect and there is room for improvement in future research, but it is good enough to support rather accurate resource selection.

There has been considerable prior research for resource selection. Most of the previous algorithms were tied with the “big document” approach. The “Big document” approach treats information sources as single documents and does not explicitly consider individual documents. The deficiency of this approach is carefully discussed in the dissertation. Some prior methods tried to turn away from the “big document” approach. However, they generally make impractical assumptions that limit their effectiveness and efficiency. The relevant document distribution estimation (ReDDE) resource selection algorithm is proposed in this work. It views information sources as document repositories instead of single big documents and it explicitly estimates the distributions of relevant documents across available sources to optimize for the high-recall goal of the information source recommendation application. The ReDDE algorithm makes full use of the information source size estimates and the content descriptions from the resource representation component [Si & Callan, 2003a]. It is not only more theoretically solid but also provides more accurate empirical results than several other alternatives within different types of federated search environments.

For a federated document retrieval system, the final step is results merging, where the individual ranked lists from selected information sources are merged into a single final ranked list. It is a difficult task especially in uncooperative environments as the diverse retrieval algorithms and the heterogeneous corpus statistics make it difficult to compare the document scores in the source-specific ranked lists. Some previous results merging methods download all returned documents and recompute comparable document scores at the search client. They are effective but are associated with a large amount of communication and computation costs. Other previous methods tried to approximate comparable document scores heuristically and they are not very effective. The regression based method (i.e., semi-supervised learning) is proposed in this work to estimate centralized document scores. This method first identifies a set of representative documents that contain both source-specific document scores and centralized document scores. These documents serve as the training data to estimate query-specific and source-specific linear models for transforming source-specific document scores to centralized comparable document scores. The models are applied on all returned documents to acquire comparable document scores and to create the final ranked list. An extensive set of experiments was conducted under different operating conditions to show the effectiveness of the new regression based results merging algorithm.

Past federated search research mainly dealt with individual components separately. The field has started to realize that effective solutions of individual components may optimize different criteria (e.g., high recall for information source recommendation, high precision for federated document retrieval). It is a better choice to integrate and adjust different components for different federated search applications. Based on this observation, this dissertation proposes a unified probabilistic framework to integrate effective solutions of different components together.

This approach first learns a single logistic model with a small number of training queries to map centralized document scores on the centralized sample database to the corresponding probabilities of relevance. For a user query, the centralized document scores of sampled documents are calculated and their probabilities of relevance are obtained through the logistic model to further estimate the probabilities of relevance for all available documents. With this information, the unified utility framework formulates the resource selection problem as an optimization problem. It allows a system to explicitly model and compensate for the inconsistencies between different goals of different federated search applications. For information source recommendation, the goal of the framework is to optimize the utility as high recall, and for federated document retrieval, the goal is to optimize for high precision.

Furthermore, a specific variant of the unified utility maximization framework, the returned utility maximization framework, is proposed to incorporate the factor of search engine retrieval effectiveness into resource selection. It investigates the consistency between two sets of ranked lists as the source-specific ranked lists and the corresponding ranked lists by an effective centralized retrieval algorithm for constructing the search engine effectiveness profiles. All of the estimated relevance information and retrieval effectiveness information can be formulated in a returned unified optimization framework to select information sources that can return many relevant documents.

An extensive set of empirical studies has been conducted to show the effectiveness of different variants of the unified utility maximization framework. The methods not only provide more effective resource selection results for information source recommendation but also enable more accurate document ranked lists for federated document retrieval. Furthermore, it has been shown that the returned utility maximization method outperforms several other alternatives when some information sources in the federated search environments are associated with ineffective search engines.

9.2 Significance of the dissertation results

Besides the specific algorithms, at least three major contributions in this dissertation advance the field of federated search. The new research turns away from the previous state-of-the-art "big document" approach. It successfully utilizes the centralized sample database for many federated search applications. Furthermore, it proposes a utility maximization model that provides more theoretically solid foundation for federated search and opens a door for many new applications.

Most previous federated search algorithms were associated with the "big document" approach. This dissertation provides a different approach that turns away from tweaking parameters in "big document" algorithms to view information sources as document repositories and build accurate models of the contents and other important characteristics of information sources. The "big document" resource selection methods were the previous state-of-the-art algorithms and they work well in federated search environments with uniform source size distributions. This dissertation points out the deficiency of the "big document" approach for treating available sources as single big documents and not considering individual documents. On the other hand, our strategy is to model available information sources as document repositories and simulate the environment of a centralized complete database as if all available documents were in a single centralized database.

Specifically, a centralized sample database (CSDB) is created by collecting all the sampled documents from query-based sampling into a single centralized database. It is highly utilized in this dissertation to simulate the centralized complete database for effective and efficient pseudo-centralized solutions. This strategy opens a door for many new opportunities. In resource selection, the centralized document scores of all available documents (mostly unseen) are estimated from the scores of documents in the centralized sample database to infer the distribution of relevant documents. In results merging, the documents in the centralized sample database serve as training data to build query-specific and source-specific models for estimating source-independent document scores from source-specific scores. The search engine retrieval effectiveness is also measured by comparing source-specific results with the results by an effective centralized retrieval algorithm on the centralized sample database.

The centralized sample database is systematically used by the unified utility maximization framework to provide an integrated view of federated search applications. This framework builds solutions for different federated search applications with different goals (i.e., high-recall goal for information source recommendation, high-precision for federated document retrieval). It utilizes the sampled documents within centralized sample database to estimate the probabilities of relevance for all available documents. Furthermore, the shift to modeling federated solutions within a single unified framework also opens a door for considering a broader set of evidence than just relevance, e.g., search engine retrieval effectiveness, search delay and information authority. Particularly, this dissertation shows an extension of the unified utility maximization framework to incorporate search engine effectiveness into resource selection algorithms for choosing information sources that can return the largest amount of relevant documents. The ability of modeling multiple types of evidence within the unified framework is very important for federated search applications in a wide range of real world environments.

In summary, the new research is supported by a more theoretically solid foundation, more empirically effective results and a better modeling ability for real world applications. It is promising to turn federated search from a cool research topic to a much more practical tool.

9.3 Future research topics

Federated search has a broad set of applications. This section describes several directions to extend the research in this dissertation.

The federated search algorithms in this dissertation mainly use the static information acquired by query-based sampling in the offline phase. On the other hand, a large amount of valuable information may be accumulated by the results from past queries and it has not been fully utilized. The new research in this work enables several possibilities for using this type of information. One specific application is to update the centralized sample database on the fly. The centralized sample database has been used extensively for many tasks such as resource selection and result merging. In our research, the centralized sample database did not change over time, but in an operational environment the centralized sample database can change by adding more documents whenever it is necessary to download documents for generating additional training data. This approach provides more training data for the semi-supervised results merging algorithm or enables the use of more sophisticated models for approximating source-independent document scores. It can also track the changing query patterns. However, it is not clear yet whether it is helpful for resource selection to add downloaded documents from past queries as this method may introduce content bias into the centralized sample database. This is an interesting problem to be explored in future research.

An alternative method to utilize results from past queries for resource selection is to model the content topics of available information sources without inserting downloaded documents into the centralized sample database. Basically, from the merged results of a particular past query, the information sources that have more documents ranked at the top of the final list tend to be more related with the query. This can be formally modeled by either supervised or unsupervised methods. Query clusters can be constructed from past queries for representing more general content topics. With this information, the unified utility maximization framework can be extended to estimate the utility of an information source not only with the relevance information estimated from centralized sample database but also with the information acquired from past queries. This can be modeled as a mixture model where one model estimates the relevance with information from centralized sample database and the other model estimates the relevance with information from past queries. It can be seen as an integration of the unified utility maximization method, which considers static information from resource representations, and the query clustering/RDD methods [Voorhees et al., 1995], which consider results from past queries.

Federated search is one solution for searching distributed information. A federated search system still has a centralized agent that does resource selection decision and merges individual ranked lists. A more decentralized solution is the (hybrid) peer to peer full text information retrieval architecture, which includes multiple directory nodes that provide regionally centralized directory services to the network for improving the routing of user queries. Leaf nodes are connected to the directory nodes and they can provide their own information as well as post queries. Many algorithms proposed in this dissertation such as the resource selection and results merging algorithms can be extended for peer to peer information retrieval applications. For example, the unified utility maximization framework can be extended for the resource ranking problem in peer to peer retrieval. Most current resource ranking algorithms only consider relevance information. However, many other factors like search delay due to the unbalanced workload strongly affect the search quality. The unified utility maximization method can be applied on each peer to provide the desired resource ranking with a good trade-off between selecting the most relevant information sources and keeping a good load balance.

The current research of federated search focuses on text based information. However, there are multiple types of information distributed in many information sources. For example, biological science research is now often conducted by teams of biologists analyzing data sets that are too large to publish in journals and sometimes collected independently by other scientists. Biologists often create and maintain their own information sources for sharing biological experimental data, biomedical image data, etc. It is generally difficult for biologists to realize the existence of many relevant biological information sources.

A realistic next step is to design an information source recommendation system that can recommend different types of biological information sources. The "big document" approach can only handle text data, which is not sufficient for this task. The new research in this dissertation turns away from the "big document" approach, which enables consideration of a broader set of evidence. The unified utility maximization model works in the framework of estimating probabilities of relevance. It is a promising method that could be utilized to transform different types of representations (e.g., text data, structured data) into the same representation (e.g., probability of relevance). Particularly, it can be used to combine evidence such as relevance and information authority. A new biological text retrieval system could be incorporated into the model for bridging the vocabulary mismatch between different biologists, which is a serious problem for biological literature [Hersh & Bhupatiraju, 2003]. Web link information could be utilized to generate authority information. All of the information could be considered in an extended utility maximization model for recommending the desired biological information sources.

Reference:

Aslam, J. A. & Montague, M. (2001). Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Avrahami, T., Yau, L., Si, L. & Callan, J. (2006). The FedLemur project: Federated search in the real world. *Journal of the American Society for Information Science and Technology,* 57(3) (pp. 347-358).

Baeza-Yates, R. & Ribeiro-Neto, B. (1999). Modern information retrieval. ACM Press / Addison Wesley.

Bergman, M. (2001). The deep web: surfacing the hidden value. http://www.completeplanet.com/Tutorials/DeepWeb/index.asp. BrightPlanet.

Bergholz, A. & Chidlovskii, B. (2004). Using query probing to identify query language features on the Web. In *Distributed Multimedia Information Retrieval, LNCS 2924,* Springer.

Broglio, J., Callan, J., Croft, W. B. & Nachbar, D. W. (1995). Document retrieval and routing using the INQUERY System. In *Proceedings of 1995 Text REtrieval Conference (TREC-3).* National Institute of Standards and Technology, special publication 500-225.

Buckley, C., Singhal, A., Mitra, M. & Salton, G. (1995). New retrieval approaches using SMART. In *Proceedings of 1995 Text REtrieval Conference (TREC-3).* National Institute of Standards and Technology, special publication 500-225.

Callan, J., Croft, W. B. & Harding, S. M. (1992). The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, (pp. 78-83). ACM.

Callan, J., Croft, W. B. & Broglio, J. (1995). TREC and TIPSTER experiments with INQUERY. *Information Processing and Management*, 31(3). (pp. 327-343).

Callan, J., Lu, Z. & Croft, W. B. (1995). Searching distributed collection with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

Callan, J., Connell, M. & Du, A. (1999). Automatic discovery of language models for text databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*. ACM.

Callan, J. (2000). Distributed information retrieval. In W.B. Croft, editor, *Advances in Information Retrieval*. (pp. 127-150). Kluwer Academic Publishers.

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

Chen, A. & Gey, F. C. (2003). Combining query translation and document translation in cross-language retrieval. In C. Peters(Ed.), *Results of the CLEF2003 cross-language evaluation forum*.

Conrad, J. G., Guo, X. S., Jackson, P. & Meziou, M. (2002). Database selection using actual physical and acquired logical collection resources in a massive domain-specific operational environment. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB).*

Craswell, N., Hawking, D., & Thistlewaite, P. (1999). Merging results from isolated search engines. In *Proceedings of the 10th Australasian Database Conference.* (pp. 189-200).

Craswell, N. (2000). Methods for distributed information retrieval. Ph. D. thesis, Department of Computer Science, The Australian Nation University.

Craswell, N., Bailey, P. & Hawking, D. (2000). Server selection on the World Wide Web. In *Proceedings of the 5th ACM Conference on Digital Libraries*. (pp. 37-46). ACM.

Duda, R., Hart, P. & Stork, D. (2000). Pattern classification. Wiley-Interscience.

Fox, E. A., Koushik M. P., Shaw J., Modlin, R. & Rao, D. (1992). Combining evidence from multiple searches. In D.K. Harman, editor, The *First Text REtrieval Conference (TREC-1).* National Institute of Standards and Technology, special publication 500-207.

French, J. C., Powell, A. L. & Callan, J. (1998). Evaluating database selection techniques: A testbed and experiment comparing the performance of database selection algorithms. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

French, J. C., Powell, A. L., Callan, J., Viles, C. L., Emmitt, T., Prey, K. J., & Mou, Y. (1999). Comparing the performance of database selection algorithms. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Fuhr, N. (1996). Optimum database selection in networked IR. In *Proceedings of the SIGIR'96 Workshop on Networked Information Retrieval.* ACM.

Fuhr, N. (1999). A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems,* 17(3) (pp. 229-249). ACM.

Gazen, B. and Minton, S. (2005). AutoFeed: an unsupervised learning system for generating webfeeds. In *Proceedings of the third International Conference on Knowledge Capture,* ACM.

Gravano, L., García-Molina, H., & Tomasic. A. (1994). The effectiveness of GlOSS for the text database discovery problem. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data.* ACM.

Gravano, L. & García-Molina, H. (1995). Generalizing GLOSS to vector-space databases and broker hierarchies. In *Proceedings of the 21st International Conference on Very Large Databases (VLDB).*

Gravano, L., Chang, C., García-Molina, H., & Paepcke, A. (1997). STARTS: Stanford proposal for internet meta-searching. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data.* ACM.

Gravano, L., García-Molina, H., & Tomasic. A. (1999). GlOSS: Text-Source discovery over the Internet. *ACM Transactions on Database Systems,* 24(2). ACM.

Harman, D. (1995). Overview of the fourth text retrieval conference (TREC-4). In *Proceedings of the Third Text Retrieval Conference (TREC-3).* National Institute of Standards ad Technology Special Publication 500-236.

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

He, B. and Chang, C.-C., K. (2006). Automatic complex schema matching across Web query interfaces: a correlation mining approach. *ACM Transactions on Database Systems (TODS)*, 31(1), March 2006, ACM.

Hersh, W. & Bhupatiraju, R. T. (2003). TREC Genomics Track Overview. Experiments using the Lemur toolkit. In *Proceedings of 2003 Text REtrieval Conference (TREC 2003).* National Institute of Standards and Technology, special publication 500-255.

Hosanagar, K. (2005). A utility theoretic approach to determining optimal wait times in distributed information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Ipeirotis, P. & Gravano, L. (2002). Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB).*

Ipeirotis, P. & Gravano, L. (2004). When one sample is not enough: improving text database selection using shrinkage. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data.* ACM.

Ipeirotis, P., Ntoulas, A. & Gravano, L. (2005). Modeling and managing content changes in text databases. In *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE).*

Jansen, M., Spink, A. & 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).

Kamps, J., Monz, C., de Rijke, M. & Börkur, S. (2003). The University of Amsterdam at CLEF 2003. In C. Peters(Ed.), *Results of the CLEF2003 cross-language evaluation forum.*

Kirsch, S. T. (1997). Document retrieval over networks wherein ranking and relevance scores are computed at the client for multiple database documents. U.S. Patent 5,659,732.

Lafferty, J. & Zhai, C. X. (2001) Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Lam-Adesina, A. M. & Jones, G. J. (2003). Exeter at CLEF 2003: Experiments with machine translation for monolingual, bilingual and multilingual retrieval. In C. Peters(Ed.), *Results of the CLEF2003 cross-language evaluation forum.*

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

Lee, J. H. (1997). Analyses of multiple evidence combination. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Le Calv, A. & Savoy, J. (2000). Database merging strategy based on logistic regression. *Information Processing & Management,* 36(3).

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

(CIKM). ACM.

Lu, J. & Callan, J. (2003). Content-based retrieval in hybrid peer-to-peer networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management (CIKM)*. ACM.

Lu, Z., Callan, J. & Croft, W. B. (1996). Measures in collection ranking evaluation. Technical Report, Department of Computer Science, University of Massachusetts.

Manmatha, R., Rath, T. & Feng, F. (2001). Modeling score distributions for combining the outputs of search engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Martinez-Santiago, F., Martin, M. & Urena, A. (2002). SINAI on CLEF 2002: Experiments with merging strategies. In C. Peters(Ed.), *Results of the CLEF2002 cross-language evaluation forum*.

Nottelmann, H. & Fuhr, N. (2003a). From uncertain inference to probability of relevance for advanced IR applications. In *Proceedings of the 25th* European Conference on Information Retrieval Research.

Nottelmann, H. & Fuhr, N. (2003b). Evaluating different methods of estimating retrieval quality for resource selection. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

Di Nunzio, G. M., Ferro, N., Jones, G. J.F. & Peters, C. (2005). CLEF 2005: Ad hoc track overview. In *C. Peters(Ed.), Results of the CLEF2005 cross-language evaluation forum.*

Och, F. J. & Hermann N. (2000). Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

OCLC (Online Computer Library Center). (2001). Statistics. Online Computer Library Center, Inc. Retrieved August 15, 2001, from the World Wide Web: http://wcp.oclc.org/stats.html.

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

Ogilvie, P. & Callan, J. (2001b). Experiments using the Lemur toolkit. In *Proceedings of 2001 Text REtrieval Conference (TREC 2001).* National Institute of Standards and Technology, special publication 500-250.

Ponte, J. M & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

Press, W. H., Flannery, B. P., Teukolsky, S. A. & Vetterling, W. T. (1992). Numerical recipes in C: The art of scientific computing. Cambridge University Press.

Roberson, S. E. & Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

Rogati, M. & Yang, Y. (2003). CONTROL: CLEF-2003 with open, transparent resources off-line. experiments with merging strategies. In C. Peters(Ed.), *Results of the CLEF2003. cross-language evaluation forum*.

Savoy, J. & Rasolofo, Y. (2000). Report on the TREC-9 experiment: link-based retrieval and distributed collections. In *Proceedings of 9 th Text REtrieval Conference (TREC-9).* National Institute of Standards and Technology, special publication 500-249.

Savoy, J. (2002). Report on CLEF-2002 experiments: combining multiple sources of evidence. In *C. Peters(Ed.), Results of the CLEF2002 cross-language evaluation forum.*

Savoy, J. (2003). Report on CLEF-2003 experiments. In *C. Peters(Ed.), Results of the CLEF2003 cross-language evaluation forum.*

Sherman, C. (2001). Search for the invisible web. Guardian Unlimited.

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

Si, L., Jin, R., Callan, J. & Ogilvie, P. (2002b). A language model framework for resource selection and results merging. In *Proceedings of the 11th International Conference on Information and Knowledge Management.* ACM.

Si, L. & Callan., J. (2003a). Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

Si, L. & Callan, J. (2003b). A Semi-Supervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 21(4). (pp. 457-491). ACM.

Si, L. & Callan, J. (2003c). Distributed information retrieval with skewed database size distributions. *In Proceedings of the NSF's National Conference on Digital Government Research (dg.o2003.)*

Si, L. & Callan, J. (2004a). The effect of database size distribution on resource selection algorithms. In *Distributed Multimedia Information Retrieval*, LNCS 2924, Springer.

Si, L. & Callan, J. (2004b). Unified utility maximization framework for resource selection. In *Proceedings of 13th ACM International Conference on Information and Knowledge Management (CIKM).* ACM.

Si, L. & Callan, J. (2005a). Modeling search engine effectiveness for federated search. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Si, L. & Callan, J. (2005b). CLEF2005: Multilingual retrieval by combining multiple multilingual ranked lists. In *C. Peters(Ed.), Results of the CLEF2005 cross-language evaluation forum.*

Turtle, H. (1990). Inference networks for document retrieval. Technical Report COINS Report 90-7, Computer and Information Science Department, University of Massachusetts, Amherst.

Xu, J. & Croft, W. B. (1996). Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

Xu, J. & Callan, J. (1998). Effective retrieval with distributed collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Xu, J. & Croft., W. B. (1999). Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Yuwono, B. & Lee, D. L. (1997). Server ranking for distributed text retrieval systems on the Internet. *In Proceedings of the 5th Annual International Conference on Database Systems for Advanced Applications.* (pp. 41-49). World Scientific Press.

Viles, C. L. & French, J. C. (1995). Dissemination of collection wide information in a distributed information retrieval system. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

Voorhees, E., Gupta, N. K., & Johnson-Laird, B. (1995). Learning collection fusion strategies. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Yang Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*. (1) (pp 69-90).

Zhai, C. X. & Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Zhai, C. X. & Lafferty, J. (2003). A risk minimization framework for information retrieval. In *Proceedings of the ACM SIGIR 2003 Workshop on Mathematical/Formal Methods in IR*. ACM.

Zhu, X. J. (2005). Semi-Supervised learning with graphs. Ph. D. Thesis, Language Technology Institute, Carnegie Mellon University.