ALGORITHMS AND STRATEGIES FOR SOURCE SELECTION AND RESULTS MERGING (COLLECTION FUSION ALGORITHMS) IN DISTRIBUTED INFORMATION RETRIEVAL SYSTEMS

GEORGIOS PALTOGLOU

PHD DISSERTATION

Supervisor Maria Satratzemi

Three-Member Commitee Michail Salampasis
Georgios Evagelidis

Department of Applied Informatics

University of Macedonia

Thessaloniki

January 2009

Acknowledgments

Reaching the end of the study for this PhD dissertation, I would like to thank the people, without the help of which this work wouldn't be possible.

First of all, I would like to thank my mentor, Michail Salampasis for giving me the opportunity to discover the world of scientific research. He taught me how to think and act independently and how to view everything with a critical eye. With careful steps, he guided me through the obstacles and showed me how to properly conduct research. I wouldn't be here, if it weren't for his support and confidence, even when things looked difficult.

I would also like to express my sincere gratitude to Maria Satratzemi for accepting me in this PhD programme and for providing me with guidance and assistance through this long journey. Her comments throughout this study and her contribution to this dissertation were unparalleled. I would also like to express my gratitude towards the third member of my committee, Georgios Evagelidis for his assistance throughout this PhD programme.

I would also like to thank all of my colleagues at the Alexander Technological Educational Institute of Thessaloniki. It has been a pleasure working and interacting with them during this period. I would like to acknowledge Christos Kouroupetroglou, Periklis Chatzimisios, Christos Tokalidis and Stefanos Harhalakis. I would also like to thank all the stuff at the University of Macedonia for their support during this programme.

This work wouldn't be possible without the continuous support of my family. I owe them my deepest gratitude for their continuous guidance, encouragement and support during my life.

ABSTRACT

General purpose search engines, such as Google and Yahoo!, provide an easy mechanism for users to discover information on the Web. Despite their obvious advantages, they have a number of significant limitations, because they cannot reach or analyze a significant part of the information that is available.

Distributed Information Retrieval systems, employing collection fusion algorithms, offer a solution to the above problem, by allowing users to submit queries to multiple information sources simultaneously through a single interface, offering a much wider coverage of the available information.

This thesis deals with two of the main issues of designing and implementing efficient and effective Distributed Information Retrieval systems: *source selection* and *results merging.* The former deals with the ability of the system to select the most appropriate information sources to delegate the user query and the latter aims to produce the best possible final document list by merging to individual retrieved documents lists from the selected sources.

The new algorithms that are presented in this thesis are designed to function effectively in settings where information sources provide no cooperation at all, thus making them applicable in the widest possible set of environments and domains. The source selection algorithm that is put forth provides a novel modeling of information sources as regions in a space created by the documents that they contain. It provides a full theoretical framework for addressing the source selection problem, while at the same time effectively captures real-world observations and widely accepted notions in Information Retrieval. Extensive experiments demonstrate that it is able to obtain a performance that is at least as good as other state-of-the-art approaches and more often better.

The novel result merging algorithms that are presented are based on the supposition that search engines return only ranked lists of documents, without relevance scores, a scenario which is standard practice in current retrieval systems. They are both able to address the lack of information very effectively, demonstrating significant performance gains over

other state-of-the-art approaches. Additionally, the second algorithm unites the two general directions that the results merging problem has been approached in research, combining their advantages while minimizing their drawbacks.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 The Size of the World Wide Web

The increase in the creation of information is unprecedented. A study by Lyman and Varian (2003) reports that the amount of information created and stored in print, film, magnetic and optical media has doubled between the years 2000 and 2002, noting an increase of about 30% per year. In 2002 alone, the last year that the study analyzed, the amount of information that was produced, was estimated to be about 5 exabytes, 92% of which was digital information stored in magnetic media, mostly hard disks.

The same study reports that the size of the indexable part of the World Wide Web, i.e. the web that is discoverable by general-purpose search engines (more on that below), has tripled in size between the years 2000 and 2002, from $20-50$ terabytes to 167 terabytes, about seventeen times the size of the Library of Congress, the largest library collection in the world. Gulli and Signorini (2005) report that in 2005 there were more than 11.5 billion pages in the indexable Web, and a more recent survey[1], which took place in April 2008, brings that number closer to 45.2 billion. In July 2008, Google, a leading general-purpose search engine[2], reported that the number of unique URLS it has discovered is more than one trillion (Alpert and Hajaj 2008), although not every one of those web pages had been indexed and therefore their contents were and remain unknown. The above facts indicate that we live in a society that creates significant volumes of data, a significant part of which is digital and is stored and accessed through the web.

It is generally believed that the rate of creation of digital information is much more

[1]http://dev.opera.com/articles/view/mama-the-url-set/

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

intense nowadays than the 2000 - 2002 period that the study of Lyman and Varian (2003) considered. A number of reasons are contributing to the above phenomenon. Computing facilities, such as personal computers and portable devices have become mainstream and the number of people that access the Internet is steadily and rapidly increasing. The "always online and available" need of today's information workers also results in a demand for a ubiquitous Internet access.

Also, significantly increasing amounts of information become available and are stored on the Internet, as the popularity of online streaming video platforms increase and blogging is becoming a mainstream practice. Additionally, the number of online web-based applications and storage repositories, also known as "cloud computing" (Wang, Tao, Kunze, Castellanos, Kramer, and Karl 2008, Hayes 2008), have significantly increased the last years resulting in a new form of user interaction with the web, not only for browsing already existing information but also for easily and rapidly creating new content, effectively promoting a paradigm shift of computer usage. Book digitization projects are also becoming more widespread (Coyle 2006) gaining significant momentum and their results are also becoming available online at a increasingly fast rate (see for example Google's Book Search[3]). Although the Internet is the newest medium of information flow, compared to other more traditional means, such as radio, telephone and TV, it is the "fastest growing medium of all times" (Lyman and Varian 2003), becoming the information medium of preference for a significant part of the population.

The rapid and unprecedented proliferation of the Web has resulted in creating an environment where users have to search multiple online information sources (such as news sites, corporate sites, blogs, online encyclopedias, wikis[4] etc) to find information relevant to their information needs. Conventional search engines, such as Google and Ask[5], provide a solution to the above problem by indexing multiple sources and providing a single point of search. These search engines work by discovering content with the aid of *web crawlers*, software agents that traverse the web following links connecting webpages, downloading and indexing them to a centralized index, thus making them discoverable and searchable by search engines users.

[3]http://books.google.com/

[4]Such as wikipedia: http://www.wikipedia.com

[5]http://www.ask.com

General purpose search engines are have to face nonetheless a number of significant problems, limiting their ability to provide to their users the best available information. The prohibitive size and rate of growth of the web make it impossible to be indexed completely. A study by Gulli and Signorini (2005) reports that the more comprehensive general purpose search engine appears to have indexed about 76.2% of the visible web, closely followed by Yahoo at 69.3%. The actual intersection of the indexes of the individual search engines amongst themselves, is at about 68.2% at best, while the majority of search engines seem to have only 50% of their indexes in common. More recent work by Bar-Yossef and Gurevich (2006) suggests that the overlap between the three biggest general purpose search engines (Google, Yahoo! and MSN) is 55% at best and as low as 22%.

Usually, general-purpose search engines give some preference to crawling websites based on a set of pre-specified rules and metrics that measure the importance of specific web-pages, such as the number of inbound links (links that point to the particular page) (Cho, Garcia-Molina, and Page 1998), the Pagerank of the webpage (Page, Brin, Motwani, and Winograd 1998), url-based rules (such as the domain of the page) (Cho, Garcia-Molina, and Page 1998) or breadth-first search order based on the web graph (Najork and Wiener 2001), etc. The above studies clearly demonstrate that the web is indexed only partially and a significant part that can potentially be reached and discovered by search engines, remains outside their indexes and thus cannot be found by their users.

1.2 Invisible Web

There is also a significant part of the web that is altogether not reachable by search engines. That part of the web is collectively known as *invisible*, *hidden*, *deep* or *dark web* (Bailey, Craswell, and Hawking 2000, Bergman 2001, Raghavan and Garcia-Molina 2001, He, Patel, Zhang, and Chang 2007).

Web sites are not crawlable by search engines, for a number of reasons. As already stated, search engines use web crawlers to discover and index new content, which are based on following links, therefore if there isn't a link pointing to a website, that site cannot be discovered by the crawler. Certain sites also explicitly do not allow their content to be indexed by search engines, by implementing the robots exclusion protocol[6], because of

[6]http://www.robotstxt.org/

financial, intellectual property rights or other reasons, offering their own search capabilities.

Also, search engines cannot, sufficiently well, index web pages that contain primarily content in flash, shockwave, images or audio, unless the textual information on that page is sufficient to provide information about the contents of the objects, although recently, a search engine (Adler and Stipins 2008) announced that it is able to index the textual information in flash content. Most importantly, search engine crawlers cannot reach web pages that are dynamically created and populated with information which resides in relational databases in response to queries submitted by users to forms, such as text boxes, drop down lists, etc. Ironically, general purpose search engines are themselves part of the invisible web, since they do not provide access to their contents by means other than through a search interface.

Also, a number of sites and webpages are password protected, i.e. are reachable only by user who have registered. Examples include highly authoritative news agencies, such as "The New York Times"[7]. Significant and authoritative information usually resides within those websites, usually produced by professional reporters.

Even publicly available, up-to-date and authoritative government information is often not indexable by search engines. Miller (2007) reports that 2,000 USA federal government Web sites are not included in commercial search engine results and are therefore part of the invisible web. Even significant parts of the website of the Library of Congress[8], are reported to be unreachable by general purpose search engines.

Studies by Bergman (2001) have indicated that the size of the invisible web may be $400 - 450$ times the size of the web indexable by search engines, reaching a size of 66,800 to 91,850 terabytes according to Lyman and Varian (2003). Thus, a user posing a query to a general purpose search engine may be missing highly qualitative and relevant information.

Overall, it can be concluded that a significant part of information that is available on the Web isn't included in the indexes of general-purpose search engines, either due to the ever increasing size of the web or because it is unreachable by web crawlers and thus cannot be retrieved in response to user queries. This "unretrievable" information is of significant quality, often higher than the average webpage that is available to search engine users. Therefore, vast amounts of authoritative knowledge and data remain outside the reach of

[7]http://www.nytimes.com

[8]http://www.loc.gov/

web users.

1.3 Distributed Information Retrieval

Distributed Information Retrieval (DIR) offers a prospective solution to the above issues. The DIR process (Callan 2000), also known as Collection Fusion (Voorhees, Gupta, and Laird 1994) or Federated Search (Si and Callan 2003b)), offers users the capability of simultaneously searching multiple remote document collections through a single interface. The terms “information sources” and “remote document collections” will be used interchangeably in this thesis to denote a set of documents with a retrieval system capable of retrieving a subset of them in response to a query. The definition is deliberately general in order to include a wide variety of environments and settings. Therefore, an information source may be a hidden web portal, a government site, a corporate vortal, a general-purpose search engine, a relational database management system etc as long as the source offers some sort of content and a search facility. An example of a DIR system is depicted in figure 1.1.

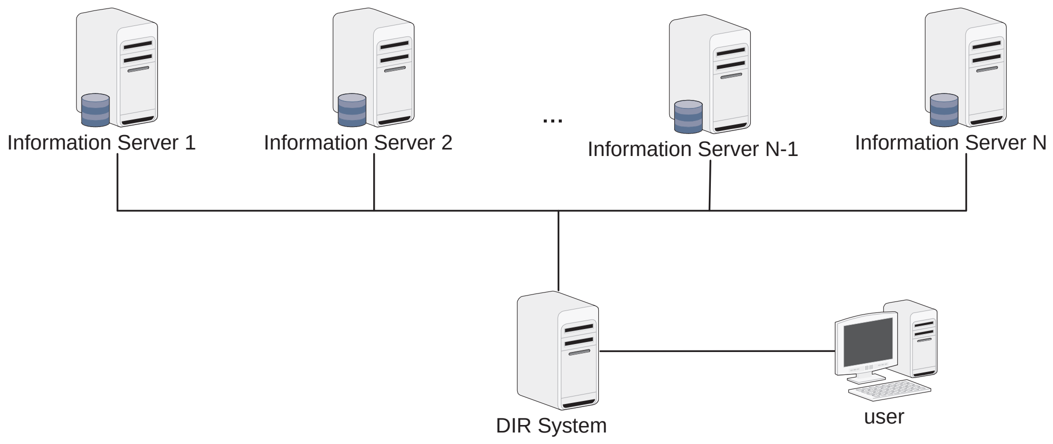


Figure 1.1 A Distributed Information Retrieval System.

The user interacts solely with the DIR system, without concerning him/herself with the intricacies of the underlying information sources. In effect, a DIR system functions as a intermediary between the user and multiple information sources. From the user perspective, the experience of utilizing a DIR system is similar to that of using any other centralized IR system, as the DIR system in principle acts as a complete interface to the underlying information sources providing to its users a holistic, unified view of the available retrieval space. Of course, personalization and parameterization of the system according to individual needs and specific applications (such as search within academic digital libraries or corporate intranets) is possible through the more advanced options usually offered by such systems.

The DIR process can be perceived mainly as three separate but interleaved sub-processes. The *Source Representation* phase (Callan and Connell 2001, Si and Callan 2003a), takes place before the user submits a query to the DIR system (figure 1.2). During this phase, surrogates of the available remote collections are created. The aim of this stage is to provide the DIR system with an, as good as possible, indication about the contents of the underlying information sources, such as their thematic topicality (i.e. politics, sports etc), if any, and potentially an indication of the effectiveness of the retrieval approach employed at each. A thorough review of the approaches that have been presented in research concerning the source representation stage is provided in chapter 2.2.

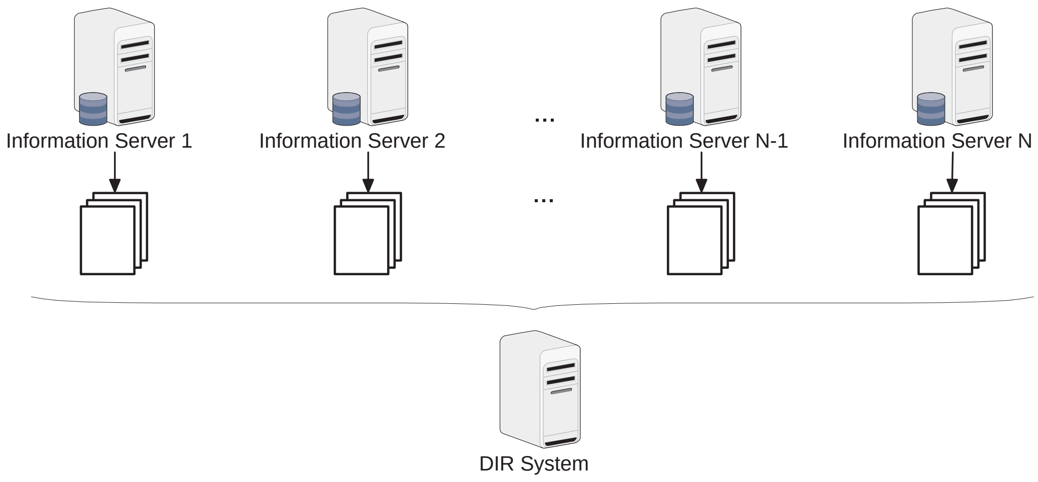


Figure 1.2 Figure depicting the Source Representation phase.

Once the user submits a query to the DIR System, a subset of the available information sources is chosen to process it during the *Source Selection* phase (Callan, Lu, and Croft 1995, Powell, French, Callan, Connell, and Viles 2000, Si and Callan 2003a) (figure 1.3). Delegating the query to all the available information sources would result in a significantly increased bandwidth and time overhead and is both inefficient as the number of the underlying remote collections may be in the order of hundreds or thousands, and doesn't necessarily result in an optimal effectiveness, as extensive experiments and real world applications by Avrahami, Yau, Si, and Callan (2006), Rasolofo, Hawking, and Savoy (2003) and others have demonstrated.

Source Selection has been the main focus of research in the context of Distributed Information Retrieval. A complete overview of the proposed approaches in research is provided in chapter 2.3 and our novel, integral-based approach for effective source selection is presented in chapter 4.

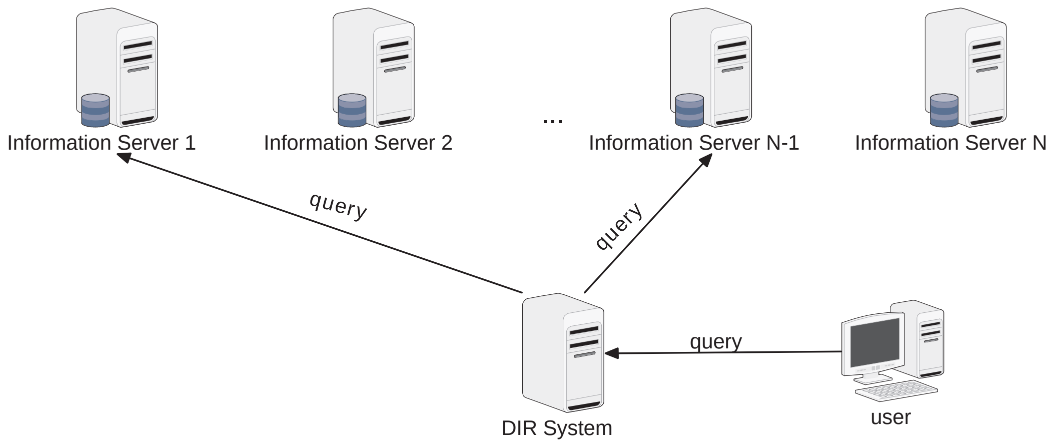


Figure 1.3 Figure depicting the Source Selection phase.

Lastly, the individual results of the selected sources are combined into a single merged result list which is returned to the user during the *Results Merging* phase (Craswell, Hawking, and Thistlewaite 1999, Si and Callan 2003b) (figure 1.4). Previous research and experiments conducted and presented as part of this dissertation in chapter 6 indicate that the results merging phase is of vital importance to the overall quality of the retrieval outcome in a significant number of retrieval scenarios and domains, as it is the last stage of the DIR process, before the system's response to the initial user query is returned. An overview of the approaches that have been presented for results merging is provided in chapter 2.4 and our two novel approaches, which are applicable at different environments each posing different constraints to the DIR process, are presented at chapters 5 and 6.

Additional stages, have also been introduced in research. *Source Discovery* (Cope, Craswell, and Hawking 2003) deals with the problem of automatically locating search interfaces of hidden web sites. Alternatively, these must be provided manually. Also, some research has also been conducted on the issue of *Query Translation* (Calmet, Jekutsch, and Schü 1997, Haas, Kossmann, Wimmers, and Yang 1997), which refers to the the modification of the initial user query to the specific query syntax used by the underlying remote

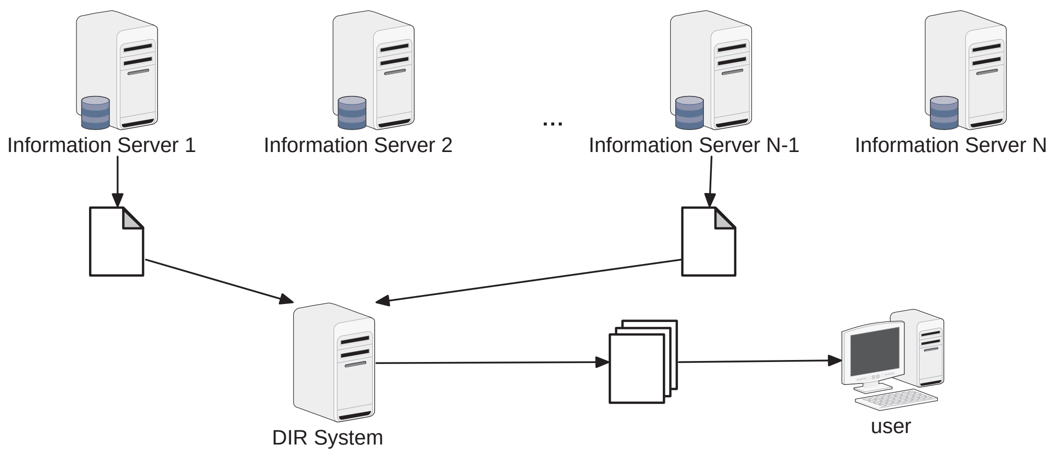


Figure 1.4 Figure depicting the Results Merging phase.

collections and lastly, on the problem of *Result Page Parcing*, extracting the results from the returned result page of the remote search engine (Ashish and Knoblock 1997, Zhao, Meng, Wu, Raghavan, and Yu 2005). Most of the research however has focused on the three stages initially reported, therefore the analysis provided here will be limited to them.

DIR is not to be confused with data fusion, which is used to define the aggregation of the results of multiple search engines over significantly overlapping document collections for the purpose of providing a wider coverage and potentially a more effective retrieval result to the user. On the contrary, DIR usually refers to the process of combining the results from multiple, typically non-intersecting, information sources. The difference is significant for a number of reasons, primarily because most data fusion techniques rely on the documents which belong to the intersection of the collections in order to estimate their final ranking, thus making them incompatible to the DIR scenario of non-intersecting collections which is the focus of this thesis.

1.3.1 Other solutions to the problem of accessing the hidden web

Ntoulas, Zerfos, and Cho (2005) report their experiments on designing and implementing an effective web crawler that can autonomously discover and download pages from hidden websites by automatically generating meaningful queries directed to their search interfaces. They experimented with generating queries in a random, generic or site-specific manner and discovered that occasionally they were able to discover and download a significant part of the hidden website. More recently, Álvarez, Raposo, Pan, Cacheda, Bellas, and

Carneiro (2007) have built a prototype hidden web crawler, named DeepBot, that receives a set of domain definitions as an input, each one describing a specific data-collecting task and automatically identifies and learns to execute queries on the forms relevant to them. Both approaches aim to make available to general-purpose search engines content that is otherwise unavailable and accessed only through hidden web resources.

They both nonetheless face serious limitations. First of all, they would have to overcome the legal issues regarding crawling sites that explicitly prohibit crawling. It has already been stated that a number of highly authoritative sites, such as MEDLINE and SAGE Publications, specifically do not allow search engine crawlers to index their content, by implementing the *robot exclusion* protocol. Failure to conform with the protocol, has often resulted in litigation from the content-provider[9]. Additionally, as hidden websites offer content that is often an order of magnitude larger than most sites, it can be conceived that even if their contents were made available for crawling, most general-purpose search engines could not afford to do so, taking into consideration that already a significant part of the visible web remains outside their indexes.More importantly, crawling a hidden website is only part of the overall aim of making its contents available to the public through an effective search facility. A second, more important issue is the retrieval approach that is applied on the crawled content. Most hidden websites have a specific thematic topicality, i.e. deal with limited and specific number of issues, such a health, state etc. For the above reason, the retrieval algorithms that have been applied on their contents have often been specifically tailored to suite their particular domain of expertise and their individual needs. An example will make that above clearer. Suppose a hidden website, dealing primarily with legal issues. Under this context, the terms "action", "case", "suit" and "lawsuit" have the same meaning, therefore a user searching for legal cases using any of those terms will retrieve similar documents. On the contrary, for a general-purpose search engine such topicality doesn't exist, therefore the user would have to adjust his/her query, potentially trying a number of combinations and variations, in order to receive the maximum number of relevant documents.

A recent study by He, Patel, Zhang, and Chang (2007) points out that while the *crawl-*

[9] See a recent example where a number of Belgian newspapers proceeded to legal action against a major general-purpose search engine at http://www.guardian.co.uk/technology /2007/feb/13/copyright.news

and-index technique used by most search engines has been quite successful for accessing the visible Web, such an model may not be appropriate for the deep Web. That is because the inherit structural heterogeneity across the wide range of deep Web data requires special considerations in order to be made useful and accessible to the public. Additionally, the simple keyword-based indexing which is utilized by most search engines will miss the schematic "structure" available in most Web databases.

Distributed Information Retrieval isn't hindered by the above limitations as it utilizes existing resources, delegating the user query to the most appropriate sources and merging the list of documents that is produced by the retrieval facility of the individual resource. In the above example, for instance, the DIR system would only have to identify that the particular website contains information that is pertinent to legal issues and delegate the user query to it, subsequently receiving as input for merging the retrieval result from the actual information source. Therefore, a DIR system provides a threefold function: firstly, it provides an efficient way of seeking information on the web, as it doesn't need to maintain its own indexes. As a result, it is potentially able to provide a single interface for thousands of individual websites. Secondly, it offers a much wider coverage of the available information than general purpose search engines as it is able to include results from multiple, heterogeneous and diverse sources. Thirdly, it is able to provide effective retrieval as all queries that are submitted to the DIR system are delegated to the most appropriate information sources making use of any specifically-tailored retrieval facilities available, guarantying that the retrieval result will be of optimal quality.

1.4 Domains of Application of Distributed Information Retrieval

1.4.1 Enterprise Search

Apart from the aforementioned application of DIR for accessing the hidden web, another very important functionality of federated search systems and one that currently knows significant development in the IT industry is in the context of *Enterprise Search.*

Vast amounts of corporate knowledge is accumulated in privately owned and managed networks that remain outside the reach of general purpose search engines for privacy reasons. That information is stored in DBM systems, legacy systems, fileshares, portal servers, Content Management Systems (CMS), email systems, corporate wikis and in a variety of file formats, such as Portable Document Format (.pdf), Microsoft Office documents (.doc), spreadsheets (.xls) etc.

That information is usually very difficult to impossible to be indexed into a single repository, as there are occasions where appropriate *wrappers* (software that provides an interface to the contents of a system that can be accessed through otherwise incompatible software) do not exist. Additionally, for large or medium sized corporations, with document repositories over various networks and topologies, it is impractical to create a single index for all the digital content spread over their private network.

The IDC report by Feldman and Sherman (2003) demonstrates the importance of effective enterprise search, by quantifying the significant economic penalties caused by poor quality search within enterprises, both in lost productivity and opportunities. Hawking (2004) presents an overview of the challenges that relate to Enterprise Search for an information driven corporation, citing several scenarios under which an effective enterprise search solution would be able to provide adequate solutions. Enterprise search still remains a very open and unresolved information retrieval issue, as demonstrated by recent empirical studies[10] by AIIM (the Association for Information and Image Management, also known as the Enterprise Content Management Association) that observe that poor enterprise search is the norm rather than the exception in the corporate world and that 49% of survey respondents "agree" or "strongly agree" that it is a difficult and time consuming process to find the information they need to do their job. A significant number of private vendors, such as Vivisimo[11], Fast[12] (now a subsidiary of Microsoft), Endeca[13] and Autonomy[14], exclusively deal with enterprise search environments, each promoting their own platforms and solutions.

Distributed Information Retrieval presents a promising solution to the challenges posed by Enterprise Search by providing a single point of search, that is capable of uniting the dispersed information repositories under a common, unified portal. Under the above con-

[10]http://www.aiim.org/ResourceCenter/AIIMNews/PressReleases/Article.aspx?ID=34834

[11]http://vivisimo.com/

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

[13]http://www.endeca.com/

[14]http://www.autonomy.com/

text, given an user information need, expressed as a query, a DIR Enterprise system would select the most appropriate information sources, delegate the query and return to the user a single, unified result list, regardless of the location, format or underlying storage medium (database or filesystem) that the information is stored.

1.4.2 Peer-to-Peer

Peer-to-peer (P2P) networks, have become increasingly popular in the context of file-sharing systems such as Napster, Gnutella or KaZaA by allowing the handling and exchanging of significant amounts of data in a distributed and self-organizing way. These characteristics offer enormous potential benefits for search capabilities powerful in terms of scalability, efficiency, and resilience to failures and dynamics. Additionally, such a search engine can potentially benefit from the intellectual input (e.g., bookmarks, query logs, etc.) of a large user community.

P2P systems can overall be divided in two general categories; In *Pure* p2p architectures, all peers are equal and all of the functionality is shared among them so that there is no single point of failure and the load is evenly balanced across a large number of peers. *Hybrid* p2p architectures includes two types of nodes: *leaf nodes* which provide the content of the p2p system and are used to represent actual users that share, exchange and search for digital information and *directory nodes*, also referred to as "hubs", "supernodes" or "ultrapeers", that provide regionally centralized directory services for a segment of the network, joining subparts of the network together in order to improve the scalability and the efficiency of the system. Napster, on the contrary, which was the first widely utilized p2p network, relied on a single central server in order to provide a search interface for the available digital content that was available to its users.

Pure p2p approaches include the Gnutella 0.4 protocol[15], Minerva (Bender, Michel, Triantafillou, Weikum, and Zimmer 2005), the system presented by Zhu and Hu (2007) and others. Hybrid solutions include the Gnutella 0.6 protocol[16], JXTA Search by Waterhouse, Doolin, Kan, and Faybishenko (2002), the Odissea network (Suel, Mathur, wen Wu, Zhang, Delis, Kharrazi, Long, and Shanmugasundaram 2003) and others. The case of the Gnutella network is of particular interest. The earlier protocol, although completely decentralized,

[15]Available at http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[16]Still in development and available at http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

was based on sending the user request to every node in the users neightborhood, until a pre-specified number of hops had been accomplished (flooding technique), an approach which is robust but also very inefficient and not scalable. On the contrary, the Gnutella 0.6 protocol adopts a hybrid p2p architecture in order to overcome the weaknesses of the previous approach. Nonetheless, both approaches only support simple name-based (filename) search.

Earlier p2p systems, and some still operational p2p system, are capable of conducting only *filename* search. For example, a user would submit the query "Information Retrieval" and only files that contained the query to the filename (or in a limited metatag information) would be returned. As p2p systems developed, content-aware applications also became available. Those systems, function much in the same function as modern search engines, analyzing and indexing not only the filenames but also the contents of unstructured or semi-structured files (such as html, doc, pdf, txt etc), thus providing powerful means of discovering digital information on a p2p network. The transition from filename search to content-aware search also resulted in a significant increase of the information that would have to be available to the p2p system in order to provide the necessary application framework. Under the above scenario, one of the key elements for the scalability, effectiveness and efficiency of the p2p systems depends on its ability to correctly select promising peers for any particular information need.

Distributed Information Retrieval techniques have been extensively utilized in emerging p2p networks that provide full-text search capabilities. Given a initial user query, the query is delegated only to the nodes which are more likely to contain relevant information for the specific query (source selection), thus providing a time efficient and quality effective solution to the problem of query propagation. After the selected peers have returned their retrieved results, these are merged, usually at the node that initialized the query in order to provide a single result list (results merging). P2P systems that provide source selection and results merging capabilities have been presented by Bender, Michel, Triantafillou, Weikum, and Zimmer (2005), Lu and Callan (2003) and others.

1.4.3 News Metasearch

Another field were federated search techniques have been successfully applied is in the context of News Metasearch engines.

The publication of information and news articles on the Internet is continuous. The biggest news portals, such as Reuters or CNN, have a very frequent rate of producing information and publishing new articles. Even more importantly, when a significant event takes place (such as an significant natural event, or an important economic or political development), the rate of article creation and publication increases significantly, as journalists "hurry" to report and comment on the events.

Current centralized search engines are not well suited for indexing volatile or ephemeral content, such as news stories, as they often cannot keep up with the continuous flow of information from news agencies, because they need to download, analyze and index the contents of their websites at very short intervals. It is true that usually, the websites of important news agencies are indexed at more often rates than other, more static websites, still it is often hours after an event has occurred that the site of an news agency is visited by the search engine's crawlers.

A user searching for current information from a diverse set of available sources has two solutions. Either he/she has to visit and search every one of the sources of interest in order to receive the most updated news or submit a query to a general search engine. The former solution will provide him/her with the most up-to-date news but requires a significant cognitive and time overhead, while the latter approach offers the easiness of a single point of search but outdated results.

A third solution, that combines the best of the above approaches, the freshness of the news as they are published on a news site and the convenience of searching from a single point is provided by News Metasearch systems.

These systems are designed to provide a single interface to multiple news sites, thus making the task of discovering up-to-date information from a diversity of sources fast and easy. The systems provide complete DIR capabilities as they are able to select the most appropriate news agencies given a user query and at a later stage merge the retrieved results into a unified list that is returned to the user. The information that is provided by these systems is up-to-date since the user query is propagated to the actual news sites and not retrieved from an outdated centralized index, thus providing up-to-the-minute relevant information. Such systems have been described in extend by Rasolofo, Hawking, and Savoy (2003) and by Liu, Meng, Qiu, Yu, Raghavan, Wu, Lu, He, and Zhao (2007), while a prototype of such a system is provided by AllInOneNews[17].

[17]http://www.allinonenews.com/

1.4.4 Personal Distributed Information Retrieval

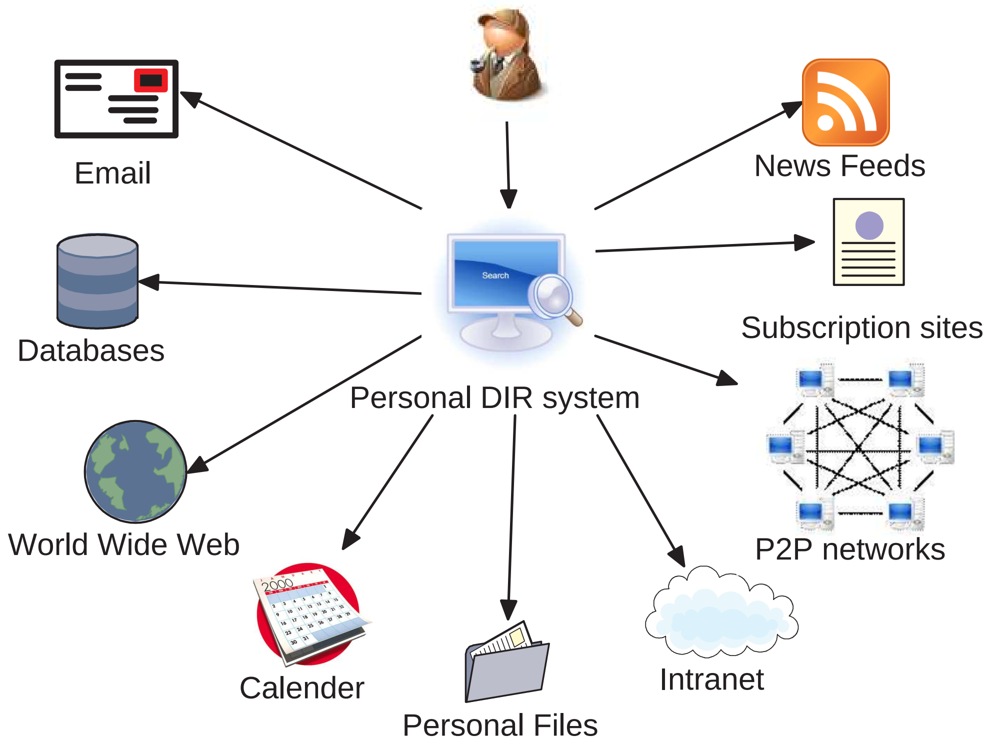


Figure 1.5 A Personal DIR tool.

Figure 1.5 illustrates an ambitious potential application of Distributed Information Retrieval: a Personal DIR tool, capable of searching all of the digital information that may be available to the user, regardless of the medium, the format or whether it is stored locally, on the web or it is being shared through a p2p network.

A Personal Distributed Information Retrieval tool would combine the capabilities of:

- Desktop Search: providing a retrieval facility for locally stored digital information, such as documents, spread sheets, multimedia files etc.
- Web Search: providing full internet search, with the aid of one or more general-purpose search engines in order to cover the largest possible part of the visible web.
- Intranet Search: making use of any available corporate retrieval facilities, in order to aid the user in obtaining work-related digital information, such as corporate presentation templates, department reports etc.

- Peer-2-peer search: providing a search facility for p2p networks at which the user has subscribed thus providing him/her with access to digital information that is shared by other p2p users.
- Website-restraint search: providing a search interface to multiple subscription-based, password-protected and potentially *hidden* websites and online services, such as repository services, fora etc.
- Application-specific search: providing a search facility for desktop applications such as calender, newsfeeds and contacts.

Despite the fact that search in the above contexts already exist as separate applications, each provides only a narrow view of the potential retrieval space, failing to capture the distribution of information over multiple mediums and formats, thus limiting its applicability to one or few more domains.

Desktop Search in particular, which at first glance somewhat assembles the capabilities of the above personal DIR tool, has received increased interest in recent years, with most general-purpose search engine vendors providing their own solutions. Examples of such products include Google Desktop Search[18], Windows Desktop Search[19], X1 Desktop Search[20] and others. Each of the above applications is able to provide advanced (more than simple boolean filename) information retrieval facilities on locally stored information in addition to internet search, usually tied to the provider internet search engine. Research on desktop search products has been done by Dumais, Cutrell, Cadiz, Jancke, Sarin, and Robbins (2003) and Cutrell, Robbins, Dumais, and Sarin (2006), but they mostly focus on usability issues and human-computer interaction concerning the domain of desktop search. They all nonetheless have in common two significant drawbacks: First of all, they are limited by the retrieval capabilities of the respective internet search engine, failing to combine the retrieval effectiveness of multiple engines and therefore excluding retrieval from subscription-based, corporate or hidden websites. Secondly, they fail to capture and represent the above retrieval space in its completeness, but rather explicitly separate the multiple facets of retrieval (local or internet), into different domains.

[18]http://desktop.google.com/

[19]http://www.microsoft.com/windows/products/winfamily/desktopsearch/default.mspx

[20]http://us.config.toolbar.yahoo.com/yds

On the contrary, the aim of a Personal DIR tool would be to provide a single, universal and extensible point of search on the whole retrieval spectrum, giving to its users a complete view of the information that can potentially assist him/her in satisfying the underlying information need.

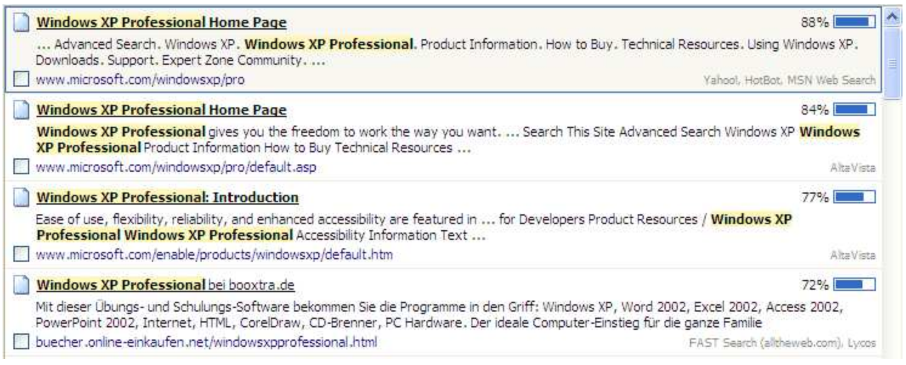
The application of DIR methodologies in the above context is rather natural and intuitive. Given a information need, in the form of a user query, the personal DIR tool would select the most appropriate sources (*source selection*), delegate the query to the selected sources, merge the returned results (*results merging*) and present them to the user, in a unified form, therefore providing a holistic, single point of search for all digital information.

The benefits of such an application are numerous; first of all it provides a single point of search in order to satisfy the information need of the user, thus reducing the cognitive load of the user that would be expended in order to learn and familiarize him/herself with multiple search interfaces and most importantly would reduce to a minimum the retrieval effort in order to find relevant information regardless of format or medium of storage (local or online).

1.5 The motivations and contributions of this dissertation

Distributed Information Retrieval has received significant attention both in academic and industrial research in recent years. The potential applications of DIR extend, as already shown, into a vast array of domains and environments, from harnessing the authoritative content that exists in the hidden web to fully decentralized content-aware peer-to-peer applications. As more digital information is created, the need and potential for retrieving data stored in distributed repositories becomes more urgent and prominent.

Prior approaches which addressed the Distributed Information Retrieval problem relied heavily on the existence of *semi-cooperative* information sources. Under that context, information sources do not provide direct access or information about their contents, that would make them *fully-cooperative*, but do provide information upon query time about the retrieved documents, reporting the relevance score of each document in regard to the submitted query. Figure 1.6 provides an example of such an information source that reports the relevance score of the retrieved documents in percentile form. This hypothesis has been extensively used in research, even though it is rarely encountered in realistic information retrieval environments.



Windows XP Professional Home Page 88%
... Advanced Search. Windows XP. **Windows XP Professional.** Product Information. How to Buy. Technical Resources. Using Windows XP. Downloads. Support. Expert Zone Community. ...
www.microsoft.com/windowsxp/pro Yahoo!, HotBot, MSN Web Search

Windows XP Professional Home Page 84%
Windows XP Professional gives you the freedom to work the way you want. ... Search This Site Advanced Search Windows XP **Windows XP Professional** Product Information How to Buy Technical Resources ...
www.microsoft.com/windowsxp/pro/default.asp AltaVista

Windows XP Professional: Introduction 77%
Ease of use, flexibility, reliability, and enhanced accessibility are featured in ... for Developers Product Resources / **Windows XP Professional Windows XP Professional** Accessibility Information Text ...
www.microsoft.com/enable/products/windowsxp/default.htm AltaVista

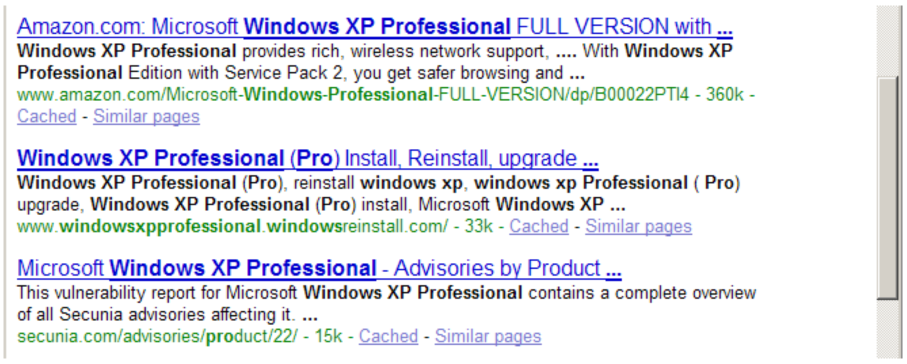
Windows XP Professional bei booxtra.de 72%
Mit dieser Übungs- und Schulungs-Software bekommen Sie die Programme in den Griff: Windows XP, Word 2002, Excel 2002, Access 2002, PowerPoint 2002, Internet, HTML, CorelDraw, CD-Brenner, PC Hardware. Der ideale Computer-Einstieg für die ganze Familie
buecher.online-einkaufen.net/windowsxpprofessional.html FAST Search (alltheweb.com), Lycos

Figure 1.6: Example of an Information Source that reports relevance scores for each retrieved document. In this particular interface the score is in percentile form.

The fact is indeed that most retrieval systems, whether search engines or database management systems, do not report relevance scores of returned documents. Users are usually satisfied receiving only a ranked list of documents, with the most relevant documents appearing on the top positions at a decreasing expectation of relevance as their ranks decrease. Figure 1.7 shows the result list of a typical information source. The lack of any indication of relevancy other than the ranking of each document is observable.

Personal experience has shown that the reporting of relevance scores next to the documents, usually results in increased user confusion rather than transparency and satisfaction. For example, in figure 1.6 what does the 88% score near the top ranking document imply? Is there a potentially better document missing that could reach 100% and if it exists why hasn't it been retrieved by the system? Users are usually able to judge to some extend the relevance or not of a retrieved document by studying its title, as well as its snippet, the text accompanying each retrieved document highlighting the terms of the query that can be found in it. As a result, most information sources do not report any numerical indication (decimal or percentile) about the estimated appropriateness of each document. The applicability of DIR algorithms that rely on the report of document relevance scores becomes at least problematic in a such environments.

One of the motivations of the work that is presented in this thesis for solving the Distributed Information Retrieval problem, is the applicability of the proposed approaches to *completely uncooperative* environments, as those described above.


Amazon.com: Microsoft **Windows XP Professional** FULL VERSION with ...
Windows XP Professional provides rich, wireless network support, ... With **Windows XP Professional** Edition with Service Pack 2, you get safer browsing and ...
www.amazon.com/Microsoft-**Windows-Professional**-FULL-VERSION/dp/B00022PTI4 - 360k - Cached - Similar pages

Windows XP Professional (**Pro**) Install, Reinstall, upgrade ...
Windows XP Professional (**Pro**), reinstall **windows xp**, **windows xp Professional** (**Pro**) upgrade, **Windows XP Professional** (**Pro**) install, Microsoft **Windows XP** ...
www.**windowsxpprofessional**.**windows**reinstall.com/ - 33k - Cached - Similar pages

Microsoft **Windows XP Professional** - Advisories by Product ...
This vulnerability report for Microsoft **Windows XP Professional** contains a complete overview of all Secunia advisories affecting it. ...
secunia.com/advisories/**product**/22/ - 15k - Cached - Similar pages

Figure 1.7: Example of an typical Information Source that doesn't report relevance scores for retrieved documents. Higher ranking documents are expected to be more relevant than lower ranking documents.

In this context, two algorithms for results merging are presented. The first algorithm is designed to explicitly function in environments where remote collections provide only ranked lists of documents with no additional data at all, not even snippet information, making it applicable in the widest possible set of environments. The *Multiple-Regression Results Merging* (MRRM) algorithm is based on maximizing the utility of the local resources already available to the DIR system from the source representation phase in order to dynamically map the collection-dependent *ranks* of the retrieved documents to collection-dependent *relevance scores*.

The notion of collection-dependent document ranking and scoring is analyzed in great depth in chapter 2.4.2, suffice to say at this point that a document's relevance score in regard to a submitted query is very much dependent on the collection in which it resides. This collection-dependence has been studied in extent in the past and is the main reason why the problem of results merging is characterized as a non-trivial one.

On the second phase, the algorithm maps the estimated collection-dependent relevance scores of the retrieved documents to collection-independent scores, thus producing the final merged list of documents that is returned to the user. MRRM requires the minimum cooperation from remote information sources: the retrieval of documents in regard to a submitted query. Therefore the algorithm is completely independent of the cooperation of the individual information source and is able to function regardless of the environment or the domain of the particular application.

Nonetheless, in order for the algorithm to function optimally, a non-trivial number of common documents between the remote collections and their locally stored representatives must be located for each submitted query. Although the experiments that will be presented later in chapter 5.6.1 consistently show that a significant number of documents, more than enough to produce an accurate model, are discovered even when a non-excessive number of documents is requested from the remote collection, the above hypothesis cannot be guaranteed in every setting and environment in regard to every possible submitted query.

The second algorithm for results merging that is presented in this thesis is based exactly on that assumption and extends the previous approach by hypothesizing that it is feasible to obtain near optimal effectiveness (i.e. retrieval quality) with only a limited compromise of efficiency (i.e. retrieval speed and bandwidth consumption). Previous research has shown that downloading all the retrieved documents and locally re-ranking them, usually results in the best performance that can be obtained in DIR settings. This methodology nonetheless incurs great costs, in processing power, bandwidth and time in order to download, index and rank all the returned documents during query time. The new algorithm that is presented in chapter 6 provides a prosperous solution to the above problems, by introducing a novel *hybrid* approach that downloads a careful selection of a limited number of documents during query time and estimates the probability of relevance of the rest solely by their ranks. Thus it is able to attain the two aforementioned goals of effectiveness and efficiency with a minimal occurring cost.

Overall, the two novel results merging algorithms that are presented in this thesis are motivated by the need of DIR systems to be able to attain the best possible performance both in terms of retrieval quality and speed, regardless of the underlying cooperation, environment and domain of the retrieval space.

Two were the basic motivations behind the conception of the source selection algorithm that is presented in this thesis in chapter 4. Previous approaches to source selection required that the number of collections to be selected as well as the number of documents that would be retrieved from each is pre-specified by the user, usually before query time. As, in most cases, information about the distribution of relevant document in remote document collections is unknown, such decisions were based on arbitrary rules. The algorithm that is presented here departs from this tradition and requires that the user only defines the number of documents that he/she wishes to view in the final merged list. The algorithm itself estimates the number of information sources that will be selected as well as the number of documents that will be requested from each, during query time, through a dynamic programming phase.

The above functionality is derived from a novel approach of modeling information sources as integrals in the rank - relevance space produced by their sampled documents using their relevance scores and the intra-collection ranking in regard to the submitted query. Extending the applications of the above modeling, the algorithm specifically addresses the two subproblems of source selection: attaining *high-recall* or *high-precision* depending on the specific application, domain and user preference in which the DIR process takes place.

The two goals are defined and elaborated in extend in chapter 4, but they are based on an observable lack of consistent optimal performance when collections that contain the largest number of relevant documents fail to retrieve them in order to make them available for merging. In order to address the above phenomenon and provide effective solutions to both of the aforementioned goals of source selection, the algorithm utilizes and extends the newly proposed modeling of information sources as integrals. Specifically, in order to solve the high-recall problem, the algorithm selects the collections that cover the largest area in the rank - relevance space, while for the high-precision goal, the algorithm divides the area covered by the remote collections into segments, each representing an estimation of the relevance of the returned lists of the collections. Based on that segmentation, the optimal distribution of documents is calculated in order to maximize the area that is covered by the final document list that will be returned to the user. It must be noted here that the algorithm doesn't require any training phase before it can be successfully utilized in DIR environments, therefore it is suitable for rapidly evolving and volatile settings such as the web or p2p networks.

A significant advantage of the aforementioned approach for source selection is its extensibility. Although the algorithm as it is presented in this dissertation only considers the documents of the remote collections as indicators of appropriateness for collection selection and thus produces a 2-dimensional integral, the above modeling can easily be extended to n-dimensional space in order to support additional evidence for selection, such as occurring costs from querying a specific collection, the speed with which documents are retrieved, a source's popularity and authoritativeness etc.

Overall, the source selection that is proposed aims to present a new framework for addressing the source selection problem, that attains not only increased effectiveness in comparison to other state-of-the-art approaches but is also extensible to support other criteria of selection depending on the particular domain of the application.

1.6 Outline

This dissertation is structured as follows. Chapter 2 offers an thorough analysis of the already proposed solutions to the DIR problem. A relatively brief description of the algorithms that solve the *source representation* problem is offered at chapter 2.2, as they are not the immediate focus of the dissertation, but will prove very useful in understand the environments that the ensuing algorithms are designed to function and the information that is available to the solutions that are proposed in this thesis.

On the contrary, a thorough analysis of *source selection* and *results merging* algorithms is provided at the rest of the chapter. A significant number of approaches have been presented in the past and in more recent years regarding the former problem and the literature review in chapter 2.3 provides a thorough survey of the most prominent approaches relevant to the context of this dissertation. Results merging has received less attention as it was considered of secondary importance in achieving optimal retrieval quality, therefore less focus to this problem have been given in research. A thorough analysis of what makes this subproblem of DIR particular difficult is presented in chapter 2.4, as well as a survey of the different aims and goals of the proposed approaches.

Serious experimental evaluation has been an indispensable part of Information Retrieval, therefore in chapter 3 an extensive analysis of the available testbeds and evaluation metrics that are utilized in this thesis to compare the performance of the newly proposed approaches to other state-of-the-art solutions are presented. An explanation and justification as to the choice of the particular testbeds and metrics used is also provided.

Chapter 4 presents the novel modeling of information sources as integrals that is proposed in this thesis to address the source selection problem. The novelty and intuition behind the approach is presented, as well as a thorough analysis of experimental results that demonstrate the effectiveness of the algorithm.

Chapter 5 presents the first of the two results merging algorithm that are explicitly designed to function in completely uncooperative environments where information sources return only ranked lists of documents without relevance scores. Chapter 6 presents the second, hybrid approach on the results merging problem, that provides a solution which attempts to attain the optimal effectiveness in a distributed setting while in parallel limiting the efficiency costs that unavoidably occur.

Finally, chapter 8 concludes the dissertation by summarizing the findings, setting the present and future vision of DIR and presenting several directions for potential future research.

CHAPTER 2

Literature Review

2.1 Introduction

In this section of the dissertation, a thorough review of the approaches that have already have been suggested in the past in order to provide solutions the Distributed Information Retrieval process will be presented.

The analysis will deal with the three main components of a DIR process, namely: *source representation*, *source selection* and *results merging*. Although this thesis focuses on the two latter problems of DIR, an overview of the source representation stage is also presented for a number of reasons: First of all, it was deemed necessary to explain the goals of source representation and the methodologies that have been presented in the past to introduce the reader into the actual environment in which most source selection and results merging algorithms have been designed to function. Secondly, source representation is an essential stage of the DIR process, creating the "initial conditions" for the ensuing stages of selection and merging and affecting the quality of the retrieval result in a non-trivial manner, therefore an analytical description of its aims and challenges could not be omitted. Lastly, the experiments that will be presented in chapters 4 to 6 assume the existence of a representative for each available source, therefore it was considered vital for the completeness of the work to analyze how the source representatives are created.

Source selection and results merging are of course analyzed in much more depth as they are the focus of this dissertation. Source selection, in particular has been a very active problem in the past and a significant number of approaches have been presented. In this chapter, we aim to take an in depth overview of the proposed approaches, analyze their strong points and weaknesses in an attempt to prepare the ground for the presentation of the novel source selection algorithm that is presented in chapter 4 and the specific motivations behind its conception and implementation.

Results merging on the other hand, has received less attention in late years. Nonetheless, it is considered a vital part of the process as it is the final stage before the DIR system returns to the user a list of documents in response to his/her initial query. Experiments presented later, in chapter 6 clearly demonstrate its importance for the overall effectiveness of the retrieval process. The actual differences in performance between the various approaches in results merging are not only non-trivial, may outweigh the differences between source selection algorithms, under the supposition that a somewhat effective source selection algorithm is utilized. The extensive analysis of the proposed results merging algorithms that will be presented in this chapter, along with an examination of their drawbacks, will provide the motivation which spurred the need for the design and implementation of the two new algorithms that will be presented in chapters 5 and 6.

2.2 Source Representation

2.2.1 Introduction

Resource selection cannot be achieved unless some information about the contents of the remote search engines is known and available to the DIR system. Information that is usually considered in order to capture the content and the thematic topicality of a remote document collection is the number of documents that are contained in a collection (the *size* of the collection), the terms that appear in it (its *vocabulary*), the number of documents that contain each term (hereafter referred to as *document frequency*) and potentially the number of times each term appears in each document (hereafter referred to as *term frequency*) etc. Usually, such information is extracted or provided from the remote information sources in advance, i.e. before the user submits a query to the DIR system (Liu, Santoso, Yu, and Meng 2001, Callan and Connell 2001, Si and Callan 2003a). There are also methodologies, which gather this information "on-the-fly", during query time (Hawking and Thistlewaite 1999).

Both approaches have their advantages and disadvantages. The former are able to guarantee a quick response time and a limited bandwidth overhead during query time, since the representatives of the remote sources are usually stored locally and are readily accessible by the DIR system while the latter create representatives of the available sources for each submitted query usually by issuing lightweight probe queries, therefore imposing an increased response time and increased bandwidth consumption. Nonetheless, the former require an increased storage capability although usually the representative of a remote collection is less that 1% of the actual collection, while the latter can function in an environment where no information needs to be stored about the remote collections, therefore are much more economical in storage requirements. Additionally, probe queries have the advantage of presenting current as well as query-specific information about information sources and may be seen as more appropriate for extremely volatile and dynamic environments (such as p2p networks), while offline methodologies usually have a single general, query-agnostic representative for each source, which may become outdated over time, although approaches which remedy the above characteristic have been presented by Shokouhi, Baillie, and Azzopardi (2007).

Regardless of the approach followed, the importance of resource representation for an effective and efficient distributed retrieval cannot be underestimated. This first stage of distributed information retrieval provides the “initial conditions” for resource selection and results merging. An accurate representation of the underlying remote engines, their contents and potentially their expected effectiveness are of critical importance.

In this section of the dissertation, a number of methodologies that are available for resource representation are explored, both for environments where this information is provided willingly by the remote collections and for environments where the above information has to be extracted by the DIR system.

2.2.2 Cooperative Environments

Although the main focus of this dissertation is on environments where remote search engines do not provide any kind of cooperation (*uncooperative* or *isolated* environments), the analysis will commence with a brief review of approaches that are designed for environments where the remote collections provide cooperation. Such environments may be found within enterprise intranets or academic digital libraries, where a single point of authority can be established and appropriate interfaces between the DIR system and the underlying information sources are available.

Stanford Proposal for Internet Retrieval and Search (STARTS)

The STARTS (STAnford Proposal for Internet ReTrieval and Search) initiative by Gravano, Chang, Garcia-Molina, and Paepcke (1997) is an attempt to facilitate the task of querying multiple document sources through a commonly agreed protocol. It provides a solution for acquiring resource descriptions in a cooperative environment, where remote collections provide important statistics about their contents. A first draft of the proposal was introduced in 1996 with the support of a significant number of organizations and companies, such as Infoseek, Verity, Microsoft, etc. The initiative defines a protocol on the *type* of information that can be exchanged between search engines and DIR systems, deliberatively leaving open the way that exchange is to take place (i.e. HTTP, XML etc). The initiative provides a set of three main features. The *source metadata* feature through which search engines can export information about themselves. The kind of information covers a wide spectrum and includes attributes such as whether the search engines uses stemming, some elements of its ranking function (such as the range of scores that can be attributed), whether or not it uses linkage information amongst webpages, the terms that appear in the index of the search engine and their document frequency etc. Such information can be very valuable for the correct selection of the appropriate sources, given a user query.

The protocol also provides a *query language* with which complex queries can be submitted to the underlying search engines such as boolean operators (AND, NOT etc), proximity filters, specifications on where the query terms appear (title or body of pages) etc. Lastly, it also provides a *merging* component that provides information about the returned documents, such a term frequencies, document sizes etc for the documents that are retrieved by the remote information sources. The usage of the above information greatly facilitates the distributed information retrieval process, effectively granting the DIR system information that is primarily available only to the actual remote engine. Unfortunately, the adoption of the proposal has been limited in practice.

Lightweight Probes

Hawking and Thistlewaite (1999) proposed a Lightweight Probe method for cooperative environments that creates query-specific representatives during query time. The method functions by broadcasting a small number p of terms to all available servers, each of which responds with a small packet of term information. The statistics that are requested by the probe are: a) the total number of documents in the server, b) the number of documents that contain each probed query term, c) the number of documents in which a specific number of the terms occur and d) the number of documents containing a specified number of probed query terms within a specific proximity of each other. The transmitted data is then used by the source selection algorithm to rank the likely utility of the servers. One of the basic assumptions of the algorithm is that a *p*-term probe is cheaper to be processed by the remote information source than a *p*-term query because no ranking needs to be actually returned, therefore the information sources will be willing to provide such information. The approach requires that probes are transmitted to every available source, therefore its applicability in DIR settings with hundreds or thousands remote collections is doubtful. Additionally, as it creates source representatives during query-time, rather than in advance, it has all the limitations of the "on-the-fly" approaches, analyzed in 2.2.1.

OpenSearch

OpenSearch[1] is a collection of technologies that allow publishing of search results in a format suitable for syndication and aggregation, original proposed by Amazon.com's subsidiary A9[2]. The technology consists of multiple parts for defining, querying and retrieving results from a search engine in XML format, either RSS or Atom.

The OpenSearch *description document* protocol provides information about search engines and how they can be queried. For example the format for imaginary search engine XML.com[3] would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <ShortName>XML.com search</ShortName>
  <dc:relation href="http://www.xml.com"/>
  <Description>Search XML.com articles and Weblogs</Description>
  <Tags>xml web</Tags>
```

[1]http://www.opensearch.org
[2]http://www.a9.com/
[3]http://www.xml.com

```
  <Contact>admin@xml.com</Contact>
  <Url type="application/atom+xml"
   template="http://search.xml.com/?q={searchTerms}&
                              p={startPage?}&format=atom"/>
  <Query role="example" searchTerms="test+xml"/>
  <Attribution>All content Copyright 2007,
                      O'Reilly and Associates</Attribution>
</OpenSearchDescription>
```

Most of the information provided in the xml, such as the name, description etc of the search engine, is straightforward. Of particular interest is the *url* element which provides information on how to submit queries to the search engine. The *template* attribute of the element provides the general format in which queries must be submitted to the system, by replacing the parameters "searchTerms" with the appropriate query terms.

Results from the cooperative search engines are provided in XML format. An example of a page of search results in RSS 2.0 format is provided below.

```
 <?xml version="1.0" encoding="UTF-8"?>
 ...
  <description>Search results for "New York history"
                                 at Example.com</description>
  <opensearch:totalResults>4230000</opensearch:totalResults>
...
  <item>
    <title>New York History</title>
    <link>http://www.columbia.edu/cu/lweb/
                           eguids/amerihist/nyc.html</link>
    <description>
      ... Harlem.NYC - A virtual tour and information on
      businesses ...  with historic photos of Columbia's own New York
      neighborhood ... Internet Resources for the City's History. ...
    </description>
  </item>
```

```
</channel>
</rss>
```

The *description* element provides information about the submitted query that produced the returned results, in this case the query being "New York history". Likewise, *totalResults* reports the number of total results retrieved for the specific query. Each *item* element is a search result, reporting the title of the document, its link and a small description suitable for snippet information.

Overall, OpenSearch defines a number of simple formats for sharing search results among information retrieval systems. Although the standard was promoted as an industry effort to aggregate results from multiple search engines, it hasn't been widely adopted and is currently implemented only by a limited number of search engines.

2.2.3 Uncooperative Environments

Remote Collection Vocabulary Estimation

When cooperation from search engines is not available or software interfaces between the DIR system and the information sources do not exist (i.e. *isolated* environments), a number of techniques have been developed that allow for an estimation of their contents as well as their sizes (number of indexed documents). Strategies that attempt to measure the effectiveness of the remote search engines have also been devised, but mainly as part of the source selection process (Si and Callan 2005).

Query-based sampling (Callan and Connell 2001) is such a technique that creates collection samples through multiple one-term queries. The approach is based on the simple idea that sampling of the remote collection infers some information about its contents and of the topicality of the underlying document collections, if any actually exists. Since most remote engines provide only a search interface and no mechanisms of browsing through the documents that they contain, that sampling may be achieved by sending multiple one-term queries and downloading the first N documents that are returned. Usually, after each response, the language model (i.e. the set of terms) from which new terms are chosen to be send to the remote collection is updated so that ensuing queries contain terms that are at least in principle guaranteed to return some documents. Terms from that language model can be chosen through various approaches: randomly, by term frequency, by document frequency etc. Usually the best results were reported to be obtained through a random selection of terms.

The above process may be repeated as many times necessary until a stopping criterion has been satisfied. Many criteria have been devised and their effectiveness examined in research (Azzopardi, Baillie, and Crestani 2006, Shokouhi, Scholer, and Zobel 2006, Caverlee, Liu, and Bae 2006). For example, the sampling may continue until a percentage of the remote documents (i.e. 10%) have been sampled. The above criterion of course requires that information about the total number of indexed documents be known, something that has been the subject of study (more information below). Alternatively, the same number of documents (i.e. 300) may be requested from each remote search engine.

Various measures have also been proposed in order to measure the quality of the produced sample. For example Baillie, Azzopardi, and Crestani (2006) suggest that the usual measures of *Collection Term Frequency* ration (CTF), which measures the proportion of terms contained in the sample of a document collection and the *Spearman Rank Correlation Coefficient* (SRCC) which account for the relative position of term rank shared between the actual and sample language model, are problematic in measuring sample quality and instead propose a new measure based on the Kullback-Leibler Divergence (KL) measure, which measure the divergence between the probabilities of terms appearing in the sample and the original document collection.

Thomas and Hawking (2007), influenced by the work of Bar-Yossef and Gurevich (2006) presented a number of novel approaches for sampling in distributed information retrieval environments. They moved away from the usual approach of query-based sampling of remote collections and presented methodologies based on random walks, term-pools and hyperlink methodologies and compared them with the standard query-based methodologies. They demonstrated that actual random sampling of remote collections cannot be easily achieved and that in most cases of documents collections and sampling methodologies some sort of bias exists.

In the context of the experiments that are reported in this dissertation, it was decided to adopt the initial query-based sampling methodology by Callan and Connell (2001), sending one-term queries to the remote search engines until a pre-specified number of documents (usually 300) had been downloaded. The reason for this choice of sampling is that the above methodology has become standard practice when evaluating source selection and results merging algorithms, so it was selected in order to make the experiments and the results presented here reproducible by others in the future. Naturally, the same setting was applied to all the algorithms tested.

Remote Collection Size Estimation

Besides an estimation of the terms that appear in the remote search engines, the actual number of documents that are indexed is also of significant importance in Distributed Information Retrieval approaches. Most state-of-the-art source selection algorithms take into consideration the size of the remote collections in order to compensate for the skewedness of the collection sizes in most realistic DIR environments.

The first methodology was proposed by Liu, Santoso, Yu, and Meng (2001) and was based on a simple *capture-recapture* approach. The algorithm assumes that there are two (or more) samples from each remote document collection. Let N be the actual size of the remote collection, A_1 the event that a document is included in the first sample, which is of size $|A_1|$, A_2 the event that a document is included in the second sample, which is of size $|A_2|$ and $|A_1 \cap A_2|$ is the number of documents that appear in both samples. Therefore:

$$P(A_1) = \frac{|A_1|}{N} \tag{2.1}$$

$$P(A_2) = \frac{|A_2|}{N} \tag{2.2}$$

$$P(A_1|A_2) = \frac{|A_1 \cap A_2|}{|A_2|} \tag{2.3}$$

Supposing that the two samples are independent:

$$P(A_1|A_2) = P(A_1) \tag{2.4}$$

Therefore, the estimated document size $\hat{N}$ of the remote collection is:

$$\hat{N} = \frac{|A_1| * |A_2|}{|A_1 \cap A_2|} \tag{2.5}$$

The samplings A_1 and A_2 are produced by sending multiple randomly generated queries to the remote collection. The methodology provides accurate estimations of document sizes but its implementation is considered unrealistic for modern environments. For example, experiments reported by Si and Callan (2003a) showed that in order to effectively estimate the size of a collection of 300,000 documents, a total number of 2,000 queries have to be issued for a total of 2,000,000 (non-unique) documents ids to be examined. Note that the algorithm doesn't require that the documents are downloaded, only that their ids (the url in a web environment) are stored.

A second, more economical and yet sufficiently accurate methodology, called *sample-resample* is presented by Si and Callan (2003a). The approach assumes that the remote search engines report the number of documents that match a query, at least for one-term queries. Let again N be the size of the remote document collection, N_{sample} the number documents sampled from the collection, df the document frequency of query q in the remote collection (the number of documents that match the query) and df_{sample} the number of sampled documents that match the query.

The event that a randomly chosen document from the remote collection would contain term q is denoted as A and the event that a randomly chosen sampled document contains query term q is denoted as A_{sample}. The probabilities for these events are estimated as follows:

$$P(A_{sample}) = \frac{df_{sample}}{N_{sample}} \tag{2.6}$$

$$P(A) = \frac{df}{N} \tag{2.7}$$

Assuming that the sample is a good representation of the actual remote document collection, we have $P(A) \approx P(A_{sample})$ and therefore the document size $\hat{N}$ of the remote collection can be estimated as:

$$\hat{N} = \frac{df * N}{N_{sample}} \tag{2.8}$$

Again, a number of queries need to be send to the remote collection and the mean of the individual estimates is used to produce the final estimation. Si and Callan (2003a) report that an average of 5 queries per search engine are sufficient to produce accurate enough estimations.

Additional methodologies for accurate estimation of collection sizes are proposed by Shokouhi, Zobel, Scholer, and Tahaghoghi (2006), inspired by the *mark-remark* techniques used in ecology to estimate the population of a particular species of animal in a region (Sutherland 1996).

In the context of the experiments that are reported in this dissertation, the *sample-resample* methodology reported by Si and Callan (2003a) was utilized, sending 5 queries per remote engine and averaging the estimated collection sizes to produce a final collection size estimation. Again, the choice of methodology was dictated by the fact that the aforementioned method is the standard methodology used when comparing source selection algorithms (Si, Jin, Callan, and Ogilvie 2002, Shokouhi 2007, Hawking and Thomas 2005).

2.3 Source Selection

2.3.1 Introduction

Source selection has received considerable attention in research. In fact, it may be argued that it has been the main focus of distributed information retrieval for a number of years. A significant number of algorithms and approaches have been presented in research.

Overall, the algorithms that have been proposed can be divided into two main categories. The first approach, sometimes referred to as "*super-document approach*", which represents the initial direction adopted, considers the remote information sources as simple *aggregations of documents*. Under that context, every remote collection is a simple *super-document*, comprised of all its documents (i.e. the borders of the constituting documents are omitted and a single document per collection is constructed). The advantages of this approach is that the source selection problem can be viewed as a special case of classical document retrieval for centralized IR, with the sole difference being in the sizes of the underlying documents. One of the most widely used source selection algorithms, namely CORI, is based on such an assertion. The disadvantage of this approach is that it fails to calculate or give an estimation of the actual individual documents that belong to the remote collection that may be relevant to the query and it only provides a general notion of collection relevance in regard to the user query.

Consider the following, simplified example. Let there be 2 document collections, collection C_1 and collection C_2, each one having four documents of equal length: $C_1 = \{d_{11}, d_{12}, d_{13}, d_{14}\}$ and $C_2 = \{d_{21}, d_{22}, d_{23}, d_{24}\}$. For simplicity reasons, suppose that a one-term query is posed to the DIR system and that the frequency of the term in the documents is: $tf_{d_{11}} = 8, tf_{d_{12}} = tf_{d_{13}} = tf_{d_{14}} = 0$ and $tf_{d_{21}} = tf_{d_{22}} = tf_{d_{23}} = tf_{d_{24}} = 2$. Despite the significant difference of the distribution of the query term in the two collections, the above approach of viewing the collections as simple aggregations of documents will rank both collections equally, since the term frequency of the query term in both collections will be $tf_{C_1} = tf_{C_2} = 8$.

The second general approach is based on explicitly regarding each collection as a repository of documents and selecting the sources that are generally most likely to return a largest number of relevant documents. The different approaches that have been adopted in estimating the number of relevant documents in each collection, have resulted in the different algorithms that have been presented in research for source selection (Si and Callan 2003a, Shokouhi 2007, Si and Callan 2004).

The approach presented in this thesis for source selection is based on the latter assertion and remote collections are not regarded as super-documents but as repositories of individual documents. The task of source selection is consequently explicitly differentiated between two different tasks: the *high-recall* and the *high-precision* goals.

The former aims at *source recommendation* applications, where it is important to direct the user to information sources with the largest number of relevant documents. A likely scenario of usage of such an application would be to direct users to hidden web sites that are most likely to contain significant amounts of information relevant to their information need. A recent paper by Seo and Croft (2009) views the Blog Distillation task, under which the goal is to direct users to blogs that have a central and recurring interest in topic X, as a special case of source selection for source recommendation applications, achieving one of the best performances in the corresponding task in TREC 2007 (Macdonald, Ounis, and Soboroff 2007). The high-recall task may in some ways be related to Broder's *navigational queries* under his taxonomy of web search (Broder 2002), according to which a large percentage (approx. 68%) of users are mainly interested in reaching "a site that provides good information" on a specific topic. The above hypothesis remains an open research topic since no user-oriented studies have been done in DIR environments to verify it.

The *high-precision* goal aims at providing the user a final merged result list with the most relevant documents appearing in the top ranks and is related to classic IR precision. Although the two tasks are highly associated, they are implicitly distinct. Algorithms that perform adequately on one, are not guarantied to perform similarly on the other. The apparent discord is based on the ability (or lack of) of the remote collections to actually return their most relevant documents for merging, having been selected in the source selection phase. Therefore, for example, a collection with a significant number of relevant documents may not be able to retrieve them for merging, while another collection containing fewer relevant documents may retrieve most of them, thus significantly contributing to the quality of the final merged document list. The observation was originally made by Craswell (2000) and has been further explored by Si and Callan (2004) where the two tasks were explicitly defined.

In the example provided above with collections C_1 and C_2, it could be observed that collection C_1 would most likely be more appropriate for the high-precision task, under the assumption that the the relatively high frequency of the query term in document d_{11} provides an strong indication of its increased probability of being relevant to the query. On the contrary, collection C_2 is most likely to be better suited for the high-recall task, as all of its documents contain the query term thus suggesting that its contents are overall more pertinent to the query than C_1, which has only one document containing the query term.

2.3.2 Prior Approaches

Glossary of Servers Server (GlOSS)

Gravano, Garcia-Molina, and Tomasic (1994) originally presented Glossary of Servers Server (GlOSS) to operate in environments where remote collections made use only of Boolean information retrieval systems. Briefly, Boolean IR systems provide rudimentary retrieval capabilities, returning unranked lists of documents that contain any, all or none of the query terms, with the usage of the boolean operators OR, AND, NOT respectively. Later, they extended the algorithm to gGlOSS (Gravano, Garcia-Molina, and Tomasic 1999) to handle vector space and other more advanced information retrieval models, which are able to provide ranked retrieval. The study here will focus on the extended gGlOSS, which is more appropriate to environments where DIR systems may be utilized.

gGlOSS assumes that remote collections can be characterized and ranked based on their *Goodness* in respect to queries submitted. The *Goodness* of remote collection C_i is defined as the sum of similarities between a query Q and documents $d_k \in C_i$, whose individual similarities are above a threshold l:

$$Goodness(l, Q, C_i) = \sum_{d_k \in C_i, sim(Q,d_k) > l} sim(Q, d_k) \tag{2.9}$$

The estimation of $sim(Q, d_k)$, the similarity between a query Q and document d_k, has been the focus of a significant number of vector, probabilistic and other models in Information Retrieval and is not examined further in this thesis. For a quick overview of some approaches presented in research the reader is directed to section 3.4 and for a thorough analysis at the works of Rijsbergen (1979), Baeza-Yates and Ribeiro-Neto (1999) and Manning, Raghavan, and Schütze (2008).

Based on the calculated $Goodness(l, Q, C_i)$ for each collection, the *ideal* ranking at threshold l, $Ideal(l)$ can be formed by sorting the collections in a descending order of goodness. DIR systems that utilize gGlOSS cannot directly compute $Ideal(l)$, as the calculation of $sim(Q, d_k)$ by the DIR system requires a significant amount of information, such as term frequencies for all the documents in the collection where query terms $q_j \in Q$ appear, which is usually not available or very expensive, in terms of bandwidth and time, to acquire. Rather, $Ideal(l)$ is advanced as a goal to which gGlOSS uses estimators $Max(l)$ and $Sum(l)$, which require only the document frequency df_j of each query term q_j and the sum of term weights $w_{j,k}$ over all documents d_k in the remote collections that contain it.

For example, suppose that the document frequency at a collection for query term q_1 is $df_{q_1} = 10$ and the respective document frequency for term q_2 is $df_{q_2} = 6$. Let the sum of term weights $w_{1,10}$ for query term q_1 be 7.5, the sum of term weights $w_{2,6}$ for query term q_2 be 5.4. We can firstly calculate the weight that each document receives from the query terms it contains. Both estimators assume that the sum of weights is spread equally among the containing document, therefore each of the documents that contain term q_1 receives a score of $7.5/10 = 0.75$ and each of the documents that contain term q_2 receive a score of $5.4/6 = 0.9$.

The $Max(l)$ estimator is based on a *high-correlation* scenario, which considers that terms occur together as often as possible. Under this scenario there are 6 documents that contain *both* q_1 and q_2 and 4 documents that contain only q_1, therefore for a l value of 0.3 we have: $Max(0.3) = 6 * (0.75 + 0.9) + 4 * 0.75 = 12.9$ while for a l value of 0.8 we get: $Max(0.8) = 6 * (0.75 + 0.9) = 9.9$. In the second case, documents that have a similarity value less than l are disregarded, therefore only documents that contain both q_1 and q_2 that have a similarity value of $0.75 + 0.9 = 1.65$ are considered.

On the contrary, the $Sum(l)$ estimator is based on a *highly-disjoint* scenario, according to which terms occurs in different documents as much as possible. Therefore, under the disjoint scenario, 6 documents contain only query term q_1 and 4 documents only term q_2. Therefore, for a l value of 0.3 we have: $Sum(0.3) = 6 * 0.9 + 4 * 0.75 = 8.4$ while for a l value of 0.8 we get: $Sum(0.8) = 6 * 0.9 = 3.6$.

For $l = 0$ all three produce the same ranking, so $Max(0) = Sum(0) = Ideal(0)$. The score attained by collection C_i when $l = 0$ is then defined as:

$$Score(C_i) = \sum_{q_j \in Q} \sum_{d_k \in C_i} wt_{j,k} \tag{2.10}$$

where $wt_{j,k}$ is the weight of query term q_j in document d_k. Usually weights $wt_{j,k}$ are proportional to the frequency of the term in the document and inversely proportional to the document frequency of the term in the collection. Again, the reader is referenced to the works of Manning, Raghavan, and Schütze (2008) and others for more information.

French, Powell, Viles, Emmitt, and Prey (1998) showed that $Max(l)$ and $Sum(l)$ are valid estimators of $Ideal(l)$, but $Ideal(l)$ is not well-correlated to relevance. On the contrary, gGlOSS was found to favor collections with a large number of documents, being strongly influenced by the size of the collection. The above bias is somewhat expected, taking into consideration that the *Goodness* of a collection is estimated as the sum of the similarities of documents above a threshold, therefore a collection with a lot of documents would accumulatively attain a higher *Goodness*, than another with less documents, even if those had higher similarities.

Overall, gGlOSS tends to rank sources with many document with low similarity higher than sources with fewer documents of high similarity, a feature that resulted in degraded performance in respect to other approaches. The above weakness of gGlOSS indicates the importance of a *collection size* smoothing parameter, in order to moderate the effects of large collections containing a significant numbers of average scoring documents, over small, but more relevant, collections.

Cue Validity Variance (CVV)

The CVV algorithm for source selection by Yuwono and Lee (1997) is based on the Cue Validity Variance (CVV) of query terms among remote document collections. The approach is based on the idea that query terms that have a higher CVV have a more unbalanced distribution among collections (i.e. occur frequently at some and rarely at others), can better discriminate between document collections and therefore should influence the final ranking of remote collections more significantly. The goodness score $G_{i,j}$ for collection C_i and query term $q_j \in Q$ is computed as follows:

$$CV_{i,j} = \frac{\frac{df_{i,j}}{N_i}}{\frac{df_{i,j}}{N_i} + \frac{\sum_{n \neq i}^{|N|} df_{n,j}}{\sum_{n \neq i}^{|N|} N_n}} \tag{2.11}$$

$$CVV_j = \frac{\sum_{i=1}^{|N|} \left(CV_{i,j} - \overline{CV_j}\right)^2}{|N|} \tag{2.12}$$

$$G_{i,j} = \sum_{j=1}^{Q} CVV_j \cdot df_{i,j} \tag{2.13}$$

where $df_{i,j}$ is the document frequency of query term q_j in remote collection C_i, N_i is the number of documents in collection C_i, $|N|$ is the number of available remote collections and $\overline{CV_j}$ is the average $CVV_{i,j}$ of query term q_j over all collections.

Equation 2.11 measures the degree to which query term q_j distinguishes collection C_i from others by calculating the variance of df_j over all the collections. Equation 2.12 measures the skewness of the distribution of query term q_j across the collections, in order to estimate its usefulness in distinguishing one collection from the other. The higher the variance, the more useful the term. Finally, equation 2.13 calculates the goodness score $G_{i,j}$ for collection C_i and query term q_j by multiplying the document frequency of query term q_j in the collection with its calculated variance. The rationale behind CVV is that a collection that contains a significant number of times a query term that has a significant variance gets a higher score than another collection that contains the same term less times or a third collection that contains the same number of times a query term that has a smaller variance.

The reasoning behind the algorithm seems to be valid and has proved to be quite effective (Yuwono and Lee 1997) in cooperative environments, outperforming gGlOSS. Nonetheless, in isolated environments where collections do not provide cooperation and source representatives need to be created through sampling, CVV has proved to be significantly less effective than other approaches (Powell and French 2003). A possible explanation of the deterioration of the effectiveness of the algorithm in such environments may lay in the fact that through the samples, important information about the skewedness of the distribution of terms amongst collections may be lost, since the samples provide only partial and incomplete knowledge about the contents of the information sources. As a result, the algorithm fails to correctly identify the terms with the higher CVV, therefore assigning more weight to terms which have a uniform distribution and ranking collections less than optimally.

Kullback-Leibler Divergence

Kullback-Leibler Divergence for source selection was originally presented by Xu and Croft (1999) and was later adapted in a language modeling context by Si, Jin, Callan, and Ogilvie (2002). According to the model, the probability $P(Q|C_i)$ of generating the query Q from collection C_i is calculated as:

$$P(Q|C_i) = \prod_{q_j \in Q} (\lambda P(q_j|C_i) + (1-\lambda)P(q_j|G)) \tag{2.14}$$

where

$$P(q_j|C_i) = \frac{tf_{ij}}{|C_i|} \tag{2.15}$$

and

$$P(q_j|G) = \frac{\sum_{C_i} tf_{ij}}{\sum_{C_i}\sum_{d \in C_i} tf_i^d} \tag{2.16}$$

$P(q_j|C_i)$ is the probability of generating query term q_j from collection C_i and is usually calculated using a maximum likehood estimate (equation 2.15) where tf_{ij} is the number of times query term q_j appears in collection C_i and $|C_i|$ is the number of terms in the collection. Respectively, $P(q_j|G)$ is the probability of generating the query term from the whole corpus vocabulary G (equation 2.16), where tf_i^d is the term frequency of term i in document d. The λ parameter is used in order to smooth the collection-based language model with the global language model and is usually set at 0.5.

Language Models have been extensively used for centralized Information Retrieval (Zhai and Lafferty 2001). Their application to distributed environments is similar in context by viewing remote collections as a concatenation of the documents that constitute them, per the "super-document" approach. An advantage that they have in relation to the aforementioned approaches is that they are not significantly biased towards large collections, as the denominator in equation 2.15 is used to penalize large collections that contain a large number of terms. Additionally, they are able to function effectively even in uncooperative

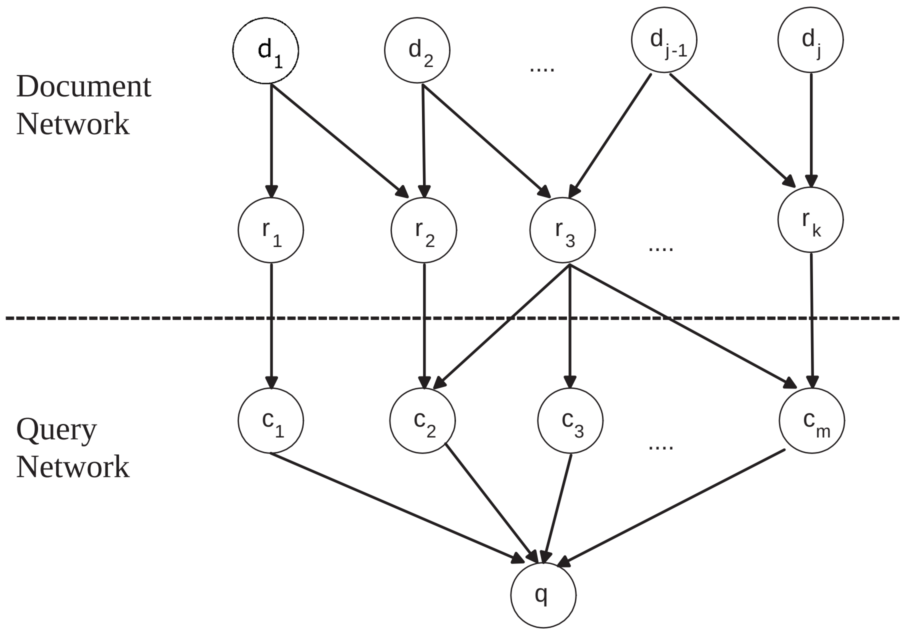


Figure 2.1 A simple document retrieval inference network.

environments where only incomplete information about the contents of the collections is available (Si, Jin, Callan, and Ogilvie 2002). Nonetheless, they suffer the deficiencies of all algorithms that follow this approach, in that they fail to consider the special properties of the source selection task in contrast to document retrieval.

Collection Retrieval Inference network (CORI)

CORI by Callan, Lu, and Croft (1995) is one of the most widely used source selection algorithms. As all algorithms presented as far, CORI considers documents collections as document surrogates, consisting of the concatenation of the collection's documents (*super-documents*).

The algorithm is based on inference networks (figure 2.1) in which the leaves (the d nodes) represent document collections and the representations (r) nodes represent the terms that occur in the collection. The probabilities that flow through the arcs are analogous to tf (term frequency) and *idf* (inverse document frequency) of centralized information retrieval, with the difference that *df* (document frequency) is used instead of tf and icf (inverse collection frequency) instead of *idf*.

Remote collections are ranked based on the belief $p(Q|C_i)$ that the information need represented by query Q is satisfied by remote collection C_i. The belief $p\left(r_k|C_i\right)$ that representation concept r_k, which is usually, but not necessarily, a term of query Q, is observed given collection C_i is estimated as:

$$T = \frac{df}{df + 50 + 150 \cdot \frac{cw}{avg\_cw}} \tag{2.17}$$

$$I = \frac{log\left(\frac{|N|+0.5}{cf}\right)}{|N| + 1.0} \tag{2.18}$$

$$p\left(r_k|C_i\right) = b + (1 - b) \cdot T \cdot I \tag{2.19}$$

where df is the number of docs in collection C_i that contain term r_k, cf is the number of collections that contain term r_k, cw is the number of terms in C_i, avg_cw is the average cw, $|N|$ is the number of available collections and b is the default belief, set to the default value of 0.4.

Finally, the overall belief $p\left(Q|C_i\right)$ in collection C_i for query Q is estimated as the average of the individual believes of the representation concepts:

$$p(Q|C_i) = \frac{\sum_{r_k \in Q} p\left(r_k|C_i\right)}{|Q|} \tag{2.20}$$

Experiments conducted by Powell and French (2003) report that CORI is more effective than both GlOSS and CVV, performing “most accurately and most consistently” across a range of testbeds and settings. Further experiments conducted by Si, Jin, Callan, and Ogilvie (2002) report that it performs similarly to Language Models in most testbeds. The algorithm has been one of the most widely used algorithms for source selection. It has been extensively used as a baseline by Powell and French (2003), Si and Callan (2004), Nottelmann and Fuhr (2003a), Shokouhi (2007), etc so it is also utilized in the line of experiments that are presented in this thesis.

Relevant Document Distribution Estimation (ReDDE)

A first attempt to estimate the number of relevant documents in remote collections by using as reference the documents that were sampled during the source representation phase was presented by Si and Callan (2003a). The algorithm was named ReDDE and gives an estimation of the number of relevant documents $\hat{Rel}_Q(i)$ for query Q in collection C_i as:

$$\hat{Rel}_Q(i) = \sum_{d_k \in C_i} P(rel|d_k) \cdot \frac{\hat{N}_i}{N_{i\_sample}} \tag{2.21}$$

where $\hat{N}_i$ is the estimated size of collection i and N_{i_sample} is the number of documents sampled from the collection. $P(rel|d_k)$ defines the probability that document d_k is relevant to query Q and is of particular importance to the algorithm.

The procedure followed by the algorithm and the estimation of $P(rel|d_k)$ necessitates some explaining. Given a user query, it is submitted to the *centralized sample index*, an index comprised of all the sampled documents indexed together. Under the assumption that the sample of each collection is uniform and unbiased, the centralized sample index therefore functions as an unbiased representative of the single centralized index, that would be created if all the documents were available for indexing. Si and Callan (2003a) use the INQUERY retrieval system (Callan, Croft, and Harding 1992) to retrieve documents from the centralized sample index, but claim that any effective retrieval algorithm would suffice.

Suppose a collection C_i with $\hat{N}_i$ estimated documents, from which N_{i_sample} have been sampled. Each time one of the sampled documents of a collection C_i appears in the retrieved list above a certain rank, the document is considered relevant and the collection it originated from gets scored a small constant C_q, therefore:

$$P(rel|d_k) = \begin{cases} C_q & \text{if } Rank\_central(d_i) < ratio \cdot \hat{N}_{all} \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

where:

$$Rank\_central(d_k) = \sum_{\substack{d_k \\ where\ Rank\_sample(d_k) < \\ Rank\_sample(d_i)}} \frac{\hat{N}_i}{N_{i\_sample}} \tag{2.23}$$

and $Rank\_sample(d_k)$ is the rank that document d_k obtained in the centralized sample index. The value of the parameter *ratio* was reported to be 0.003 in the original paper. The parameter C_q doesn't influence the effectiveness of the algorithm, since it is constant for all collections.

ReDDE therefore attempts to estimate the number of relevant documents in collection C_i $\hat{Rel}_Q(i)$, by summing up the number of documents originating from that collection, that are retrieved above a certain threshold in the centralized sample index, multiplying that number by its *scale factor* SF_i, which is defined as:

$$SF_i = \frac{\hat{N}_i}{N_{i\_sample}} \tag{2.24}$$

SF_i represents the inverse portion of the collection documents that have been sampled from collection i. The rationale behind the above multiplication is that for any such high ranking document ReDDE assumes that there should respectively be SF_i documents in the actual remote collection, that are expected to score similarly and therefore should also be considered relevant. For example, if a document belonging to the collection C_i has a score of 0.7, then there should be approximately SF_i documents in the actual collection that will have about the same score. The same intuition is applied to all the documents returned by the centralized sample index.

However, there have been studies examining whether the samples created through sampling are truly random. Shokouhi, Zobel, Scholer, and Tahaghoghi (2006) demonstrated that the resulting sample is heavily biased. Even if a truly random sample of all the available collections were possible, it is very unlikely that the above assertion will hold true. That is because there are no evidence that an unbiased sampling of documents would result in a uniformity in the distribution of document scores, in regard every query posed to the remote information source. The randomness of sampling refers to the ability of the sampling procedure to retrieve documents in a random fashion (Shokouhi, Zobel, Scholer, and Tahaghoghi 2006), i.e. the distribution of documents over multiple sampling processes in a true random sampling should simulate the Poisson distribution. Potentially, a sampling process that takes place during query time, such as the Lightweight Probe technique by Hawking and Thistlewaite (1999), may provide a solution to the above issue, but that remains an open research question. Therefore the assumption that an accurate estimation of the relevant documents for each collection is produced is severely flawed.

Regardless of that, the algorithm does have a significant number of merits, as it is the first attempt to view the problem of source selection as explicitly different from document retrieval. Additionally, the algorithm, for the first time in research, incorporates the collections sizes into the source selection process, not in a manner that will bias the results towards large collections, but rather as an indicator that large collection that have some high-ranking sampled documents are more likely to contain more relevant documents than smaller collections.

The performance of ReDDE is similar to CORI at testbeds with collections of similar size and better when the sizes vary significantly as demonstrated by Si and Callan (2003a), Hawking and Thomas (2005) and Shokouhi (2007) and it is also used as a baseline in the experiments that are presented later in this thesis.

Central-Rank Source Selection (CRCS)

Two variants of the original ReDDE algorithm, CRCS(l) and CRCS(e), were presented by Shokouhi (2007), assigning different weights to the documents that are retrieved from the centralized sample index depending on their rank, in a linear or exponential fashion instead of a constant value (the C_q parameter in equation 2.22) as ReDDE. They are based on the notion that documents that rank higher in the centralized sample index are mote likely to be relevant and therefore the collections that they originated from, must be rewarded proportionally. Specifically, the final score of remote collection C_i is estimated as:

$$Score(C_i) = \frac{\hat{N}_i}{\hat{N}_{max} \cdot N_{i\_sample}} \cdot \sum_{d_k \in C_i} R(d_k) \tag{2.25}$$

where $\hat{N}_{max}$ is the estimated size of the largest remote collection.

$R(d_k)$ is the impact of document d_k that originated from the sample of collection C_i and is ranked at position $Rank\_sample(d_k)$ in the centralized sample index, using an effective retrieval algorithm. (in the original paper, the INQUERY retrieval system was used) and can be computed either linearly:

$$R(d_k) = \begin{cases} \gamma - Rank\_sample(d_k), & \text{if } Rank\_sample(d_k) < \gamma \\ 0, & \text{otherwise} \end{cases} \tag{2.26}$$

for CRCS(l), or:

$$R(d_k) = \alpha \cdot exp(-\beta \cdot Rank\_sample(d_k)) \tag{2.27}$$

for CRCS(e), using a negative exponential function. The values of parameters α, β and γ are heuristically set to 1.2, 0.28 and 50 respectively.

CRCS(l) and CRCS(e) are particularly effective in producing a final merged list with a significant number of relevant documents in top ranks and thus attained improvements in precision in comparison to CORI and ReDDE over some testbeds, but the ability of the algorithms to correctly identify collections with the largest number of documents (i.e. *recall*) is usually lower. In order to provide a complete and thorough analysis of the algorithm for source selection that is presented later in this thesis, both approaches are also utilized as baselines.

Decision-Theoretic framework (DTF)

The Decision-Theoretic framework (DTF) for source selection was originally presented by Fuhr (1999). The basic assumption behind the framework is that a specific cost $Cost(d_k, Q)$ can be assigned to each collection C_i when document $d_k \in C_i$ is retrieved for query Q. The user specifies the total number n of documents which should be retrieved from the available m remote collections and the task is to compute an optimum solution, a vector $\overline{d} = (d_1, d_2, \ldots, d_m)^T$ which minimizes the overall cost:

$$M(n, q) = min_{|s|=n} \sum_{i=1}^{m} Cost(d_k, Q) \tag{2.28}$$

An important issue at DTF is the definition of "costs". The term is used in a broad way in order to include, besides money, other factors like quality and time. In particular, the quality of the retrieval results, which is measured by the effectiveness of the process, has received special attention. Fuhr (1999) defined the cost C^+ for viewing a relevant document and C^- for viewing an irrelevant document, where $C^- > C^+$. Therefore, if $r_i(d_k, Q)$ denotes the number of relevant documents in the result set of collection C_i for query Q, the cost function is:

$$Cost_i^{rel} = r_i(d_k, Q) \cdot C^+ + (d_k - r_i(d_k, Q)) \cdot C^- \tag{2.29}$$

The Decision-Theoretic framework is one of the first attempts to approach the problem of source selection from a theoretical point of view. However, it requires extensive training making use of human relevance judgments in order to built a model of probability for relevance for each available remote collection making it highly impractical for realistic web environments. Implementations of the approach by Nottelmann and Fuhr (2003a) have been reported to perform worse than CORI, particularly on short queries. An attempt to combine DTF with CORI was also presented by Nottelmann and Fuhr (2004) by combining the scores attained by each collection in a linear manner but again the performance improvements were minimal for short queries while more substantial for mid (9.9 terms) and long (87.5 terms) queries.

In contrast to the original theoretical DTF approach, the implementations presented select a pre-specified number of collections and request a pre-defined number of documents from each, in a sense “violating” the original nature of the algorithm of computing an optimal solution that minimizes the overall cost. The source selection algorithm that is presented later in the thesis, can be viewed as a variation of the original theoretical approach, but uses a different modeling of information sources and of the retrieval process, not as a functions of cost, but as integrals in the rank-relevance space. Additionally, in comparison to the implementations of DTF the new algorithm dynamically estimates the number of sources that will be queried as well as the number of documents that will be retrieved from each, without requiring any training phase, in order to maximize the overall area covered by the retrieved documents.

Hybrid Source Selection

A hybrid approach to source selection, which combines centralized and distributed information retrieval in web-based environments was presented by Hawking and Thomas (2005). The work focuses at environments where not all of the available remote collections offer search capabilities and additionally some of them are crawlable. It was inspired by a survey taken of the GOV testbed provided by NIST[4] (more information on the testbed is provided in chapter 3) which reports that only 31% of the servers appear to be searchable (i.e. offer a search option), therefore applying a true DIR approach would be unfeasible.

As an alternative, a hybrid system is considered, whereby some or all of the servers with a search interface are considered candidates for selection and the remainder of servers are crawled into a centralized index. To answer a user’s query, a source selection algorithms selects a number of searchable servers to delegate the user query, including the central index.

Two novel, anchor-based approaches are presented and are tested in the environment presented above. The first, named HARP (Highest Anchor Ranked Page), creates surrogates for documents (web pages) through the links that point to them, found at the central index. So if, for example, two links were found on the central index with anchor texts *government* and *energy*, pointing to a particular document, then the surrogate for that document would

[4]http://ir.dcs.gla.ac.uk/test_collections/govinfo.html

be $\{government,\ energy\}$. Note that the actual content of the document may be unknown and only the anchor text that points to it is utilized in order to create its representative. The set of surrogates is indexed together and given a user query a score is assigned to each surrogate based on the following formula:

$$w_t = \alpha log(tf_d + 1) log\left(\frac{|N_i| - n + 0.5}{n + 0.5}\right) \tag{2.30}$$

where w_t is the weight contribution of (Porter stemmed) query term t, N_i is the size of the remote collection and n is the number of documents whose incoming anchor text contains t.

A ranked list of 1000 surrogate documents is returned and they are grouped, based on the server they reside. A list of servers is thus formed. The ranking that they have is based on the ranking of their highest ranking document. In the end of the list, the central index is always added in order to avoid returning an empty list to the user.

The second approach, named AWSUM (Anchor Weighted Sum), is a variant of HARP in which each page in the original ranking contributes to the score of its server. In order to prevent a server with many low-value pages overwhelming another with fewer but higher-value answers, each page's contribution is its score divided by its rank. Servers are ranked by descending score, with the central crawled collection at the tail of the list if not otherwise present.

The above approaches were found to be effective in web-oriented tasks, such as homepage finding where the goal of the user is to find the homepage of a company or product. The improvements they recorded weren't significantly better than the performance of ReDDE. Nonetheless, their applicability and adoption are limited to the above considered environment where some of the available remote collections are crawlable. The usage of hyperlinks in DIR environments is a novel approach, compared with previous approaches that base their selection on the actual contents of the remote collections but it is difficult to imagine how the approach could be implemented in a true distributed environment where the only information about the sources is provided through their samples and sources do not contain hyperlinks to one another.

Unified Utility Maximization (UUM)

The Unified Utility Maximization (UUM) framework, presented by Si and Callan (2004) is based on estimating the probabilities of relevance of documents in the remote collections. The algorithm selects collections in order to maximize a utility function, which can be defined as the solution to two types of maximization problems, aiming to address and specifically differentiate between the two tasks in source selection, *high-recall* and *high-precision* problems.

The utility function for the high-recall problem can be defined as:

$$U(\bar{d}, \theta^*) = \sum_{i=1}^{|N|} I(C_i) \sum_{j=1}^{\hat{N}_i} \widetilde{R}(d_{ij}) \tag{2.31}$$

where $C_i = \left\{d_{i1}, d_{i2}, \ldots, d_{i\hat{N}_i}\right\}$ is a collection, $|N|$ is the number of available collections, $\hat{N}_i$ is the estimated size of collection C_i and $\bar{d} = \left\{I(C_1), I(C_2), \ldots, I(C_{|N|})\right\}$ is a selection vector where $I(C_i)$ is an indicator function which is 1 if C_i is selected and 0 otherwise. Particular attention must be given to vector θ^*, where:

$$\theta^* = \left\{\left(\widetilde{R}(d_{1j}), j \in \left[1, \hat{N}_1\right]\right), \left(\widetilde{R}(d_{2j}), j \in \left[1, \hat{N}_2\right]\right), \ldots, \left(\widetilde{R}(d_{|N|j}), j \in \left[1, \hat{N}_{|N|}\right]\right)\right\}$$

that contains the estimated relevances of all documents at remote collections, where $\widetilde{R}(d_{ij})$ is the estimated probability of relevance of document d_{ij}. These relevances are calculated through a training process. More on that below.

The goal of the model is to find a selection vector $\bar{d}$ to maximize the utility function. Thus:

$$\bar{d}^* = argmax_{\bar{d}} \sum_{i=1}^{|N|} I(C_i) \sum_{j=1}^{\hat{N}_i} \widetilde{R}(d_{ij}) \tag{2.32}$$

subject to:

$$\sum_{i=1}^{|N|} I(C_i) = N_{\bar{d}} \tag{2.33}$$

where $N_{\bar{d}}$ is the predetermined number of collections for selection.

The utility function for the high precision problem is defined as:

$$U(\bar{d}, \theta^*) = \sum_{i=1}^{|N|} I(C_i) \sum_{j=1}^{\tilde{n}_i} \widetilde{R}(d_{ij}) \tag{2.34}$$

where $\tilde{n}_i$ is the number of requested documents from collection C_i. Note that the only difference between equation 2.34 and 2.31 is the sum of probabilities of relevance of documents. Equation 2.31 sums the probabilities of *all* the documents in remote collections, while equation 2.34 sums the probabilities of documents up to rank $\tilde{n}_i$ and therefore differentiates between the high-recall and the high-precision problems.

The goal of the model is again to calculate a selection vector $\bar{d}$ that maximizes the utility function:

$$\bar{d}^* = argmax_{\bar{d}} \sum_{i=1}^{|N|} I(C_i) \sum_{j=1}^{\tilde{n}_i} \widetilde{R}(d_{ij}) \tag{2.35}$$

subject to:

$$\sum_{i=1}^{|N|} I(C_i) = N_{\bar{d}} \qquad and \qquad \tilde{n}_i = N_{rdoc},\ if\ I(C_i) = 1 \tag{2.36}$$

where N_{rdoc} is the number of requested documents from collection C_i.

One of the key issues with UUM is the estimation of the probabilities of relevance $\widetilde{R}(d_{ij})$. In order for the algorithm to learn them, an extensive training process needs to take place in advance. Specifically, the algorithm requires that a number of training queries be used (i.e. 50 are suggested in the paper), in regard to which CORI is utilized to select the top 10 collections and merge their top 50 documents, using SSL. The top 50 documents in the final merged list need to be downloaded and human judgments be issued for those documents.

Although the training phase is less time and resource consuming than that required for DTF, it is still significant, making the methodology inapplicable in dynamic information retrieval environments, such as the web or p2p systems. A significant contribution of the algorithm is the explicit distinction between the *high-precision* and *high-recall* goals of source selection. Both goals are viewed as different optimization problems, therefore the algorithm is able to present a complete framework under which the source selection problem can be viewed. The performance of the algorithm was found to be similar to ReDDE in some testbeds and better in others.

Returned Utility Maximization (RUM)

A variation of the UUM algorithm was presented by Si and Callan (2005) that also takes into consideration the effectiveness of the retrieval at the remote collections.

For each information source the algorithm builds a retrieval effectiveness profile in the form of a mapping function:

$$\varphi_{ij} : d_{c\_i}(r_1) \rightarrow d_c(r_2) \tag{2.37}$$

where φ_{ij} is the ranked list transformation of the j^{th} training query for the i^{th} collection, $d_{c\_i}(r_1)$ is the r^{th} document in the ranked list from the i^{th} collection and $d_c(r_2)$ is the r_2^{th} document in the ranked list produced by the retrieval system on the centralized sample index.

The approach is again based on finding the selection vector $\bar{d}$ that maximizes the utility function $U(\bar{d}, \theta^*)$:

$$\bar{d}^* = argmax_{\bar{d}}\, U(\bar{d}, \theta^*) \tag{2.38}$$

subject to:

$$\sum_{i=1}^{|N|} I(C_i) = N_{\bar{d}} \qquad and \qquad \tilde{n}_i = N_{rdoc},\ if\ I(C_i) = 1 \tag{2.39}$$

where:

$$U(\bar{d}, \theta^*) = \sum_{i=1}^{|N|} I(C_i) \sum_{j=1}^{||J||} P_i(j) \sum_{j=1}^{\tilde{n}_i} \widetilde{R}(\varphi_{ij}(d_{c\_i}(r))) \tag{2.40}$$

where $P_i(j)$ is the probability that the rank pattern of remote collection i is similar to the rank pattern of the effective retrieval algorithm employed at the centralized sample index for training query j^{th} and is calculated as $P_i(j) = \frac{1}{||J||}$ ($||J||$ being the number of training queries).

Experiments presented (Si and Callan 2005) showed that RUM attains similar performance to UUM when most of the remote information sources utilize effective retrieval approaches and is better when most of the remote informations source use ineffective retrieval algorithms. Nonetheless, the differences in the results are not as pronounced as it might be expected. For example, when two thirds of all the available collections employ ineffective retrieval algorithms, the difference reported on precision in comparison to UUM is on average around 14.6% over all the testbeds, when it would be expected that potentially a more accurate ineffectiveness indicator may have a stronger preference for the effective information sources, thus attaining a clearer performance improvement.

Overall, the results showed that the modeling of the effectiveness of the retrieval with the mapping function φ_{ij} does have some merits, but its applicability in realistic environments would be problematic. As already discussed one of the most significant advantages of DIR in comparison to centralized approaches to accessing the hidden web, is that the former approach delegates the query to the actual information source where potentially "specially-tailored" retrieval approaches are utilized. RUM nonetheless compares the effectiveness of the retrieval algorithm utilized at the remote collection with a generalized approach, in which any thematic topicality of the retrieval algorithm will be viewed as a lack of effectiveness, rather than a benefit. In other words, the algorithm isn't able to distinguish between a truly ineffective information source and another that uses specific algorithms to produce highly relevant results in its specific topicality. Nonetheless, it cannot be argued that the algorithm does present a first attempt of dealing with the problem of retrieval effectiveness in remote collections.

News Metasearch

The work of Wu, Meng, Yu, and Li (2001) also discuss the problem of source selection, although in a different context than the approaches already presented. The original algorithm that is proposed focuses exclusively on remote collections that use the vector space model to retrieve documents and is only applicable in cooperative environments, thus limiting its applicability.

A revised, more general version of the ranking algorithm was presented by Liu, Meng, Qiu, Yu, Raghavan, Wu, Lu, He, and Zhao (2007). The *adjusted maximum normalized weight* $amw(q, C_i)$ for query term q and collection C_i is calculated as:

$$amx(q, C_i) = log\left(\frac{\sum_{C_i} |C_i|}{\sum_{C_i} df_{C_i}}\right) * max\left\{\frac{tf_{d_k}}{|d_k|},\ d_k \in C_i\right\} \tag{2.41}$$

where df_{C_i} is the document frequency of query term q in collection C_i (i.e. the number of documents that contain q) and tf_{d_k} is the term frequency of term q in document d_k (i.e. the number of times q appears in the document).

Collections are rank based on the $amx(q, C_i)$ score that they achieve. Usually, for efficiency reasons only the top 50 $amx(q, C)$ scores are kept for each term and the rest are disregarded.

The algorithm has been employed in realistic environments, resulting in the development of the AllInOneNews metasearch engine[5]. Nonetheless, no comparisons in regard to other source selection algorithms have been conducted therefore no measure of effectiveness in comparison to other already analyzed approaches exist. Additionally, the algorithm hasn't been tested in laboratory settings and IR testbeds, as those that will be described in chapter 3, making it difficult to make final conclusions about its overall effectiveness.

2.4 Results Merging

2.4.1 Introduction

Although most of the focus in distributed information retrieval has been on source selection, significant progress has also been made in results merging.

The results merging phase is the last part of the distributed information retrieval process, where the individual result lists from the remote collections are merged into one single, unified list which is returned by the DIR system to the user submitting the query. Previous research (Callan 2000, Craswell, Hawking, and Thistlewaite 1999), as well as new experiments presented in chapter 6 consistently show that the results merging phase is vital to the overall effectiveness of the retrieval process, especially in precision oriented environments where users expect a significant number of relevant documents in the top ranks of the returned document list. Even if the most appropriate information sources have been chosen in the previous stages, if the merging isn't effective the overall quality of the retrieval process will deteriorate. This importance is augmented particularly in the web environment where users rarely look past the top 20 results and most often do not browser after the top 5 results as observed by Jansen, Spink, and Saracevic (2000).

Merging the result lists from individual collections is a complex problem not only because of the variety of retrieval algorithms that may be utilized at the remote information sources, but also because of the diversity of individual corpus statistics among document collections, resulting in an intensely heterogeneous environment and an inconsistent scoring between collections.

[5]http://www.allinonenews.com

2.4.2 Prior Approaches

One of the first experiments in results merging was conducted by Voorhees, Gupta, and Laird (1994) and was revisited again by Voorhees, Gupta, and Johnson-Laird (1995). In the original work two approaches where tested: one simple interleaving algorithm and a probabilistic biased c-faced die algorithm. The interleaving approach is based on the assumption that all chosen collections have the same number of relevant documents and it works by simply interleaving the documents from the individual collections one by one. It was found to be highly ineffective since the above assumption is rather improbable in most environments. The biased c-faced die approach was based on merging the results from the remote collections in a weighted manner, using information about the number of returned documents. For example a collection that returned twice as many documents than another collection, would have twice as many chances of having its documents in the top ranks. The methodology produced better results and was considered the most sophisticated technique that could be adopted in isolated environments in the absence of a sampling of the remote collections and document relevance scores. The probabilistic nature of the latter algorithm was later re-examined by Yager and Rybalov (1998) and various deterministic approaches were presented.

CORI Results Merging

In environments where the remote collections return not only ranked lists of documents but also a relevance score for each returned document, a variety of approaches have been proposed. *Raw score merging* merges the results as they are returned from the remote collections based on the relevance score that they were given, but it was found to be ineffective (Callan, Lu, and Croft 1995) since it required that the scores are within a common range (i.e. between 0 and 1). The problem of incomparable scores was overcome by normalizing the returned scores at a common range, usually by dividing them by the largest returned score. This approach produced better results, but the problem of different corpus statistics, eventually resulted in incomparable scores. For example, in a collection that is mainly about sports, a document containing the term “computer” will rank high if that term appears in the query, while the same document may rank lower in a computer science related collection. The fact that statistics between collections vary significantly and that therefore simple score merging (raw or normalized) is ineffective was originally made by Dumais (1994).

Weighted score merging overcomes the above issue by assigning each document a final score which is based both on an estimation of the relevance of the document itself, as indicated by the relevance score assigned to the document by the remote collection it originated and an estimation of the appropriateness of collection where it belongs, as calculated by the preceding source selection algorithm. This way, high scoring documents from low scoring collections (as in the above example) rank lower than highly relevant documents from highly relevant collections. The CORI results merging algorithm by Callan (2000) is such a weighted scores merging algorithm and is one of the most widely used algorithms, because of its effectiveness on one hand and its simplicity on the other. The final score $D^{''}$ of a document according to the algorithm is calculated as:

$$C_i^{'} = \frac{C_i - C_{min}}{C_{max} - C_{min}} \tag{2.42}$$

$$D^{'} = \frac{D - D_{min}}{D_{max} - D_{min}} \tag{2.43}$$

$$D^{''} = \frac{D^{'} + 0.4 * D^{'} * C_i^{'}}{1.4} \tag{2.44}$$

Equations 2.42 and 2.43 normalize the collection and document scores respectively to a range of 0 to 1 while equation 2.44 assigns the final relevance score to each document. Specifically, C_i is the relevance score of the collection according to the source selection algorithm, C_{min} and C_{max} are the minimum and maximum scores respectively that any collection can be assigned by the source selection algorithm and $C_i^{'}$ is the normalized collection score. Similarly, D is the relevance score given to a document by the remote collection (the initial document score), D_{max} and D_{min} are the maximum and minimum document scores that could be assigned by the collection and $D^{'}$ is the document normalized score. Note that D_{max} and D_{min} require cooperation from the remote collection to be set. In case when this cooperation is not available, they are set to the relevance score achieved by the most and least relevant document respectively. Finally, $D^{''}$ is the final document score, which again is normalized (thus the division by 1.4) between 0 and 1.

The equation of particular importance to the algorithm is equation 2.44, where the final document score is assigned. It can be observed that two are the important elements that influence $D^{''}$: the normalized original document score $D^{'}$ and the normalized collection score $C^{'}$. A more general formulation of the equation could have been:

$$D^{''} = \frac{D^{'} + (n-1) * D^{'} * C_i^{'}}{n} \tag{2.45}$$

where parameter n serves two causes: First of all, it indicates the total weight shared among the original document score and the collection score and secondly, it sets the weight assigned to the $D^{'} * C_i^{'}$ accumulator, integrating the collection relevancy score with the document score. The formula, although simple in its conception, has proven to be surprisingly robust across a number of testbeds and is still used as a baseline in most results merging experiments.

Semi-Supervised Learning (SSL)

Si and Callan (2003b) presented a results merging algorithm, named Semi-Supervised Learning (SSL), that influenced one of the novel results merging approaches that are presented later in this thesis, in chapter 5.

SSL assumes that the optimal merging of documents that are returned from remote collections and consequently the optimal final result list, is the one that approximates as closely as possible the list that would be returned if all the documents were available for indexing in a single centralized index. The goal therefore of the algorithm is to map *collection-specific* document scores D to *collection-independent* document scores $D^{'}$ and thus to provide a result list that closely approximates a centralized scenario. These two concepts deserve an extended explanation at this point, as they will significantly aid in the understanding of the rationale behind the general aims of results merging, as well as the rationale behind algorithms presented later on.

Collection-specific document scores are the scores that remote collections assign to their documents in response to a query (for example score D from the CORI results merging algorithm, in equation 2.42). *Collection-independent* document scores are the scores that the documents would be assigned in regard to the same query if they were indexed in a single centralized index. These two scores are expected to be different for a number of reasons. First of all, given a query, a document score depends on the retrieval model adopted by the remote collection. A single collection will return different scores for its documents in regard to the same query if it applies a different retrieval model, for example Okapi BM25

(Robertson, Walker, M., and M. 1994) or a Language Model (Zhai and Lafferty 2001). Even though both of this algorithms are considered to be very effective, the rankings they produce are also expected to be different.

Additionally, most retrieval algorithms use collection specific statistics in order to estimate document scores, such as the number of documents in the collection, the document frequency of query terms (the number of documents in which the term appears) etc. These can differ significantly among collections, therefore given the same query, the same document contained in two otherwise different collections using the same retrieval algorithm, would be assigned a different score by each.

Based on the above reasons, *collection-specific* and *collection-independent* scores are expected to be different, firstly because, the retrieval algorithms that are employed at the remote collections are unknown and beyond the control of the DIR system and secondly, the statistics between each remote collection and the hypothetical single index are expected to be significantly different, both in document size as well as term distributions.

In order to achieve the optimal merging, SSL assumes the existence of a *centralized sample index* (CSI) that is comprised of all the sampled documents from the remote collections. The CSI is considered as a representative of the single centralized index that would be created if all the documents were available for indexing. Assuming that the sampling of the remote collections is random, or as close as possible to random, then CSI provides a holistic view of the remote collections such as the terms that are present, their relative frequency compared to other terms etc. These statistics are expected to be close to the statistics of a single central index, therefore the centralized sample index is used as a representative of that single index.

The CSI is also utilized by a significant number of source selection algorithms (see paragraph 2.3.2), but in that context only the ranking of documents is actually directly used and not the underlying corpus statistical information that produces this ranking, while in the context of results merging, the focus is on term information.

The SSL algorithm functions as follows: given a user query and a selection of collections, the query is delegated to the appropriate information sources and is also submitted to the centralized sample index. In the original paper, the INQUERY search engine (Callan, Croft, and Harding 1992) was used to query the centralized sample index, but any effective retrieval algorithm is reported to suffice.

The algorithm therefore receives as input a list of documents with their respective collection-independent scores from the centralized index and a set of lists of documents with their respective collection-dependent scores from the selected collections. It is assumed that at least some documents between the centralized sample index and the result list of every collection are common. These are defined as "overlap" documents.

The algorithm takes advantage of the overlap documents discovered between the result lists of the remote collections and the centralized sample index and uses their corresponding relevance scores to estimate a linear model, that maps the former to the latter. The aim of the algorithm is to use this estimated model, which is usually different for each collection, in order to assign collection-independent scores to the non-overlap documents, thus estimating a score for every returned document.

The existence of a sufficient number of overlap documents has been the focus of research both at the original paper (Si and Callan 2002) and again in (Si and Callan 2003b). Usually, 1,000 documents are requested from each information source, a number which although is in accordance with other experimental settings presented by Voorhees, Gupta, and Laird (1994), Powell and French (2003) and others, is deemed as impractical in most realistic information retrieval environments. A second important assumption of the algorithm is that remote collections return not only simple ranked lists of documents, but also report the scores that the documents were assigned.

Nonetheless, as already discussed in paragraph 1.5, in most modern information retrieval environments, such scores are usually not available since they are not utilized by most users and additionally provide significant information about the internal workings of the search engine. Therefore, the applicability of the algorithm in realistic web environments is somewhat limited.

Based on that observation, two novel results merging algorithms are presented in this thesis, both of which are designed to function in completely uncooperative environments, where remote collections return only rankings of documents. The novel results merging algorithm that is presented in chapter 5 additionally conducts a thorough investigation on the effects on the overall retrieval quality of requesting a more realistic number of documents from each collection and the second algorithm presented in chapter 6 completely alleviates the need for an excessive document request by limiting the number of documents to the possible minimum.

Estimating the relevance of document through Logistic Regression

An attempt to estimate the probability of relevance of documents returned from remote collections by making use only of their ranking has also been made in the past by Calvé and Savoy (2000).

In that work, logistic regression was used in order to estimate the probability of relevance of documents, using as information their ranking. Specifically, given the rank of the retrieved document D_i, the probability of relevance is estimated as:

$$Prob[D_i \text{ } is \text{ } Rel|x_i] = \pi(x_i) = \frac{e^{\alpha+\beta x_i}}{1 + e^{\alpha+\beta x_i}} \tag{2.46}$$

where x_i is the log of the rank of the retrieved document. In this equation, the coefficients α and β are unknown parameters which fit the S-curve of the logistic function. The value estimates for these coefficients are noted as $\hat{\alpha}$ and $\hat{\beta}$, respectively, and are calculated according the maximum likelihood principle using human binary judgments of relevance. The model could be extended in order to take into consideration other indicator of relevance, such as date or relevance scores, nonetheless only the rank is considered in the original paper.

The approach proved to be more effective than raw or normalized score merging but it requires an extensive training phase before it could be utilized successfully. In addition to that, the approach creates a single model for each collection, regardless of the query being posed, thus making its application problematic in volatile and rapidly evolving environments, such as the web.

Feature Distance Ranking Algorithms

Last but not least, a number of approaches download "on-the-fly" (i.e. during query time), partially or fully, the documents that are returned from the remote collections in order to produce a final ranking. The *Feature Distance Ranking Algorithms* by Craswell, Hawking, and Thistlewaite (1999) are two examples of such an approach. The rationale behind them is that a document should be scored lower than others if a query term: a) appears near its end, rather than the beginning, b) isn't near other query terms (supposing a multi-term query), c) has already occurred many times in the document, and d) is commonly found in the remote collection. Based on the above principles, a weight w is assigned to documents for each occurrence of query term q, determined by: a) the character offset of the occurrence l (the number of characters that precede it in the document), b) the distance d between two consecutive occurrences of the term, c) the number of times n the query term has already appeared in the document and d) the total document frequency df of the term.

Two weighting schemes are presented by Craswell, Hawking, and Thistlewaite (1999), w_A and w_B:

$$w_A = \frac{1}{n \cdot \sqrt{d \cdot df} \cdot ln(l)} \tag{2.47}$$

$$w_B = \frac{1}{n^{1.1} \cdot ln(d) \cdot ln(df + 1) \cdot ln(l)} \tag{2.48}$$

The final score of each document is calculated as the sum of the individual weights assigned to it, $score = \sum_Q w$. The advantage of these methods is that they can estimate "first hand" the relevance of a document and do not have to rely on the ranking or the score that it received from the remote collections. Their disadvantage is that they have an increased time overhead and bandwidth requirement in order to download and index all the returned documents, even if the download is only partial.

In the experiments that were presented by Craswell, Hawking, and Thistlewaite (1999) it was demonstrated that w_A weighting scheme is usually more effective than w_B and that its performance is similar to that of downloading the full documents and using the Okapi ranking algorithm (Robertson, Walker, M., and M. 1994).

Results merging algorithms that are based on snippet information

Lastly, a family of results merging algorithms designed primarily for news search engines by Rasolofo, Hawking, and Savoy (2003) but extended for general purpose search engines by Lu, Meng, Shu, Yu, and lup Liu (2005) are based on taking advantage of the snippet information that most search engines provide in their search results. The approaches presented by both works assign documents a final merged score based on various factors, such as the rank and the title of the document in the result list of the search engine, the frequency and the proximity of query terms in the snippet, the usefulness of the search engine (its estimated relevancy to the query) and the publication date of the document (in the case of the news metasearch approaches) and others.

An example of such a scoring algorithm is provided below. The SRRSim scoring function by Lu, Meng, Shu, Yu, and lup Liu (2005) assigns a similarity score to document D in regard to query Q, according to the formula:

$$sim(Q,D) = c_2 * Similarity(Q,Title) + (1 - c_2) * Similarity(Q,Snippet) \tag{2.49}$$

where c_2 is a parameter, set to 0.5, $Similarity(Q,Title)$ is the similarity between the query Q and the title of the document and $Similarity(Q,Snippet)$ is the similarity between the query Q and the snippet. Usually these similarities are calculated using the $Okapi\ BM25$ function (Robertson, Walker, M., and M. 1994), where the score of document D (in our case D is either the $Snippet$ or the $Title$ of the document) for query Q is calculated as:

$$Okapi(Q,D) = \sum_{T \in Q} w \frac{(k_1+1) * tf}{K + tf} * \frac{(k_3+1) * qtf}{k_3 + qtf} \tag{2.50}$$

$$where\ \ w = \frac{N - n + 0.5}{n + 0.5}\ \ and\ \ K = k_1 * \left((1-b) + b * \frac{dl}{avgdl}\right) \tag{2.51}$$

where tf is the frequency of the query term T within document D, qtf is the frequency of T within the query, N is the number of documents in the collection, n is the number of documents containing T, dl is the length of the document and $avgdl$ is the average length of all the documents in the collection. $k1$, $k3$ and b are parameters of the model with values 1.2, 1,000 and 0.75, respectively. Since N, n and $avgdl$ are unknown in the context used here, where information about the actual documents is unavailable, approximations were used to estimate them. For $avgdl$, the average length of the documents that were collected in a sampling was used, setting the value at 1,424.5 (words). Google's size at the time was used in order to simulate $N = 8,058,044,651$. Each query term T is submitted as a single-term query to Google and the number of documents returned as the value of n.

Surprisingly, none of the aforementioned approaches use the *url* of the returned documents in their estimation, a strategy that has proved very effective in tasks such as homepage or known-item finding. Additionally, both of the approaches fail to mention and investigate the importance of the *snippet extraction* algorithm in the overall effectiveness of the merging. Lastly, taken into consideration that most engines present snippets that are focused around query terms and use "..." in order to signify that a part of the documents has been emitted between two occurrences of the query terms, it is very difficult to estimate the distance between two query terms in the actual document.

In the context that the results merging problem is studied in this thesis, the existence of title and snippet information is *not* pressumed. That makes the methodologies presented applicable in a wide set of environments, such as the web, news metasearch applications or p2p networks. Additionally, the approaches proposed are not dependent on the effectiveness of the *snippet extraction* algorithm, a dependence that would otherwise had to be investigated.

The results merging problem is often confused with the metasearch problem, where a number of information retrieval algorithms pose a query to a single document collection or multiple similar collections and subsequently merge the results of the individual algorithms in one final list (Lee 1997). Most of the approaches under that context make use of the common documents returned from the various sources in order to find the most relevant. They are based on the concept that the more a document appears at the returned lists of the individual algorithms, the more relevant it is.

Techniques like COMBSUM or COMBMNZ (Lee 1997) estimate the final score of a document as the sum of the scores obtained by individual document collections, or by multiplying this sum by the number of collections which had non-zero scores. This thesis focuses on environments where remote collections have no documents in common (i.e. are non-intersecting), thus making the utilization of the above algorithms inappropriate.

The term "news metasearch" that has been used above and will be used in this thesis, is only utilized to define a DIR system that functions as an intermediary between the user and various *news* sources. It is assumed that these sources each provide their own articles and that no articles can be found in common among the sources. The term is therefore used only to be in accordance with the terminology used by other researchers in the field, although its usage may be considered somewhat misleading.

2.5 Summary

Distributed Information Retrieval has been a very active field of research, both in academia and in industry. In this chapter, a thorough analysis of the most prominent approaches, relevant to the scope of this dissertation, were presented.

Initially, a definition of cooperative information sources was provided and the protocols and technologies that aim to provide such a functionality were presented. In environments where cooperation is unavailable, techniques have been developed that aim to extract information about the contents of remote collections as well as other characteristics. The utilization of these techniques alleviates a significant part of the problems of DIR systems operating in uncooperative environments, as the extracted information is usually sufficient to generate accurate-enough representatives of the remote collections and therefore facilitate the subsequent phases of the retrieval process. The focus of this thesis is on environments that belong to the latter category, therefore an analytical review of the available approaches was presented and the algorithms that are utilized in the experiments that were conducted and are presented at later parts of this dissertation are discussed.

Source selection has been the main focus of DIR research, resulting in the development of a significant number of algorithms. These fall under two general categories: those who view the source selection problem as a special case of classical centralized IR, per the "super-document" approach and those that explicitly view the source selection problem as a problem with its own characteristics and specific challenges, such as the *high-recall* and *high-precision* goals. Algorithms that belong to both approaches were presented and their advantages and drawbacks indicated and discussed. In the experiments that were conducted, prominent approaches from both categories are utilized in order present a complete coverage of experimental results and evaluations.

Results merging is the last phase of the DIR process and is therefore very significant in a significant number of domains and applications, where the quality of the final, merged document list is of particular importance. Although a significant number of algorithms, such as CORI and SSL, are domain-agnostic and can be readily utilized in almost every setting and environment, a number of domain-specific approaches, such as news metasearch that utilize snippet information, where also presented. The focus of this thesis is on domain-agnostic algorithms, that have the significant advantage of adaptability to a wide set of domains and applications but for completeness reasons, a review of the latter is also provided. In the experiments that will be presented, only the former are discussed as they are most relevant to the work presented here. Additionally, the latter are dependent on a number of factors that are outside the aims of this thesis, such as the snippet-generating algorithm, therefore a thorough experimental analysis would require an overview of those factors.

Overall, it can be observed that a non-trivial number of algorithms have been proposed in research to facilitate every phase of the DIR process. The above may be considered as an indication of the importance of the particular field as well as of the significance of the applications where such approaches may be utilized and adopted, a number of which was cited in chapter 1. As the amount of digital information is increased and the contexts and media under which it becomes available become even more diversified and ubiquitous, DIR algorithms may find applicability in an ever increasing number of domains, therefore the significance of the presented algorithms as well as of the new approaches that are put fourth later in this thesis will increase.

CHAPTER 3

Evaluation Methodology

3.1 Introduction

The field of Information Retrieval has a "strong tradition of serious experimental evaluation" (Callan, Allan, Clarke, Dumais, Evans, Sanderson, and Zhai 2007). Designing and implementing new algorithms and novel approaches to existing problems is very important in IR, but usually there is a need to display some improvement over already existing approaches, in terms of efficiency or effectiveness, in order for the work to know wide acceptance. For that reason, there is an increased need for standard testbeds and invariant evaluation metrics, that provide a common ground for measuring the performance of different approaches.

Although it would be very useful to be able to conduct experiments with real users performing information retrieval tasks at the real web or at environments as those described in chapter 1 and directly measuring the results they receive, doing so is not practical for a number of reasons. First of all, human experiments are expensive both in time and money, as they require the input of a significant number of individual users, often with varying backgrounds in order to simulate the wider possible audience, willing to make a significant effort in assisting the research process. Additionally, user-centered experiments do not offer the kind of verification properties that are important for the rapid advancement of the IR field, such as the ability to easily reproduce the reported results.

Research in DIR systems poses an additional challenge, in that the remote information sources are usually beyond the control of the DIR system, therefore introducing unknown parameters into the evaluation process, which may vary over time, creating an unstable experimental environment.

Although there is a significant number of studies of user behavior and human interaction with IR systems, such as the work done by Jansen, Spink, and Saracevic (2000), Rose and Levinson (2004), Cutrell, Robbins, Dumais, and Sarin (2006) and so fourth, most of the experiments reported by IR researchers focus around some sort of automatic evaluation of retrieval performance. Work by Al-Maskari, Sanderson, and Clough (2007) provides an attempt to unite the above two approaches, giving an indication of the relationship between automatic system evaluation and user satisfaction.

For the above reasons, a mechanism had to be devised that offers researchers the ability to conduct *objective* experiments in *laboratory settings*, that would give them the opportunity to isolate the phenomenon under investigation without worrying about underlying changes and without the necessity of human users.

3.2 Experimental Testbeds

IR testbeds offer the capability of conducting experiments in a completely isolated and controlled environment, giving researchers the ability to easily test new theories and approaches and verify reported results.

An IR testbed is comprised of three components: *a document corpus*, a set of *topics* (i.e. a number of predefined queries) and lastly a set of *relevance judgments* (the "right answers") for each of the topics. Each of the components of a IR testbed serve a specific purpose: The document corpus aims at simulating the real web, or more generally the retrieval space whatever that may be (i.e. an enterprise intranet, files stored on a filesystem etc), by providing a controlled "miniature" of it. The topics' aim is to simulate real users submitting queries to the IR system and lastly, the relevance judgments provide a common ground on defining which documents are considered relevant for each individual information need. The overall aim of the testbed is to provide a common ground of data and evaluation techniques in order to facilitate cross-system evaluation and comparison.

The Text REtrieval Conferences (TREC)[1] (Harman 1993, Harman 1995) that are conducted every year since 1992 and are sponsored by the National Institute of Standards and Technology (NIST) aim to encourage research on text retrieval based on large-scale test collections, by providing large corpora with sufficient queries and relevance judgments.

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

TREC data provides an excellent ground for conducting Information Retrieval experiments both in centralized as well as distributed settings. The database merging track of TREC-4 (Harman 1995) was an early indication of the capabilities offered by the conference for the advancement of distributed information retrieval.

The TREC document corpus contains newswire articles from a wide variety of sources, such as the Wall Street Journal, the Federal Register, the Associated Press etc. The news and government data that is offered by TREC closely resembles the information that is similar to web resources that are the focus of this dissertation, such as hidden web or enterprise resources as reported by Avrahami, Yau, Si, and Callan (2006), Miller (2007). That data is written by professionals focusing on non-trivial issues in a authoritative manner, offering original content.

The document corpus is distributed on CD-ROMs with about 1 gigabyte of data each, compressed to fit. The contents of the first three disks (TREC CDs $1, 2, 3$) are as follows:

- Disk 1
  - *WSJ* - Wall Street Journal (years 1987, 1988, 1989)
  - *AP* - Associated Press newswire (year 1989)
  - *ZIFF* - Articles from "Computer Select disks (Ziff - Davis Publishing)"
  - *FR* - Federal Register (year 1989)
  - *DOE* - Short abstracts from the Department of Energy
- Disk 2
  - *WSJ* - Wall Street Journal (years 1990, 1991, 1992)
  - *AP* - Associated Press newswire (year 1988)
  - *ZIFF* - Articles from "Computer Select disks (Ziff - Davis Publishing)"
  - *FR* - Federal Register (year 1988)
- Disk 3
  - *SJMN* - San Jose Mercury News (year 1991)
  - *AP* - Associated Press (year 1990)

– *ZIFF* - Articles from “Computer Select disks (Ziff - Davis Publishing)”

– *PAT* - U.S. Patents Office (year 1993)

Table **3.1** shows some basic document collection statistics, concerning the sizes of the individual document collections, as well as the average number of terms per article.

Source	**Size (MBs)**	**Number of Records**	**Median number of terms per record**	**Average number of terms per record**
WSJ	517	173,252	200	353
AP	500	164,597	349.5	372.5
ZIFF	423	132,100	174	403
FR	473	45,820	314	1,045
DOE	186	226,087	82	89
SJMN	245	90,257	279	337
PAT	290	6,711	2,896	3,543

Table **3.1** Statistics about document corpus from the TREC 1, 2, 3 CDs

The *documents* constituting the corpus in TREC data are uniformly formated into an SGML-like structure, for example:

```
<DOC>
<DOCNO> WSJ881117-0168 </DOCNO>
<HL> Securities Rules To Be Rewritten In Britain Again </HL>
<AUTHOR> Craig Forman (WSJ Staff) </AUTHOR>
<SO> </SO>
<CO> EUROP </CO>
<IN> SCR </IN>
<DATELINE> LONDON  </DATELINE>
<TEXT>
Just seven months after introducing a sweeping securities rulebook,
Britain plans to overhaul its regulations, replacing a tough regime
of business-conduct rules with a looser, simplified code of good
practice principles.
...
```

```
</TEXT>
</DOC>
```

Each document has a beginning and an end marker, along with a different DOCNO id field in order to uniquely identify it. The main content of the article is contained within the TEXT tags. Some additional tags are also provided, such as the AUTHOR which declares the author of the articles, if one is stated, but they are mostly ignored.

TREC topics are also provided in SGML-like form. A sample topic is provided below:

```
<top>
<head> Tipster Topic Description
<num> Number: 101
<dom> Domain: Science and Technology
<title> Topic: Design of the "Star Wars" Anti-missile Defense System
<desc> Description:
Document will provide information on the proposed configuration,
components, and technology of the U.S.'s "star wars"
anti-missile defense system.
<smry> Summary:
Document will provide information on the proposed configuration,
components, and technology of the U.S.'s "star wars"
anti-missile defense system.
<narr> Narrative:
A relevant document will provide information which aids
description of the design and technology to be used in the
anti-missile defense system advocated by the Reagan administration,
the Strategic Defense Initiative (SDI), also known as "star wars."
...
<con> Concept(s):
1. Strategic Defense Initiative, SDI, star wars, peace shield
...
<fac> Factor(s):
<nat> Nationality: U.S.
```

```
</nat>
<def> Definition(s):
</top>
```

The *num* tag declares the unique id of the query and the *title* tag gives an initial first description of the topic, and it is usually this field that is used in order to simulate the user queries. The *desc* provides a longer description of the topic, more rarely used for user queries. Lastly, the *smmr* and *narr* tags provide instructions to the human assessors on which documents should be considered relevant for the particular query.

Relevance judgments are the last important and indispensable part of a test collection, as they provide the “right documents” for each topic. The goal in creating relevance judgments is to collect a list of relevant documents for each topic and make that list as comprehensive as possible. Clearly, it is nearly impossible to manually inspect every single document in regard to every topic to create complete assessments as the number of avalable documents is $O(1000)$. Therefore, all TRECs have used a *pooling* method (Sparck Jones and Van Rijsbergen 1975) in order to assemble the relevance judgments. According to the method a sample of the available documents, those that are retrieved by the participating systems, are considered as possibly relevant documents and they are shown to human assessors, whose aim is to manually inspect them and judge their relevancy in regard to the respective topic. For the TREC conferences in particular, the sampling method that was used was to take the top 100 documents for each query retrieved by each participating system and merge them into the pool for assessment. Research by Zobel (1998) and Voorhees (1998) have shown that the above pooling method produces valid and stable results in ranking retrieval systems according to their performance despite potential missing relevant documents or disagreeing human assessors.

An example of relevance judgments for TREC Query 101, shown above is provided bellow:

```
101 0 AP880212-0047 1
101 0 AP880330-0014 1
101 0 AP880330-0182 0
...
...
```

The first column of indicates the unique topic number. The third column indicates the unique document id (the DOCID of the document as shown above) and the fourth column indicates whether or not that document has been judged relevant, in which case it is denoted by 1 or 2 or non-relevant, in which case it is denoted by 0. The second column is always zero.

3.3 Distributed Information Retrieval Testbeds

A common and popular approach for creating testbeds for Distributed Information Retrieval experiments from the original TREC CDs is to partition the data by source and date into small collections, each one representing an information source. For example Xu and Callan (1998) partitioned the data from TREC 1 and 2 CDs into 107 collections, Rasolofo, Abbaci, and Savoy (2001) partitioned the documents from TREC-8 into 4 collections based on their source and the documents from TREC-9[2] into eight collections of approximately equal size, French and Powell (2000) partitioned the TREC $1, 2, 3$ CDs into 236 collections and Powell and French (2003) partitioned the data from the TREC Very Large Corpus (VLC) into 921 collections. Compared with the previous divisions, the partition of TREC 1, 2, 3 CDs by source and date has been very popular in DIR experiments and has been extensively used in the past by Xu and Croft (1999), Powell, French, Callan, Connell, and Viles (2000), Callan (2000), Si and Callan (2004), Nottelmann and Fuhr (2003a), Shokouhi (2007), etc resulting in the existence of a significant number of baseline results, therefore it is used in the line of experiments that is presented here.

- **Trec123-100col-bysource ("Uniform"):** The documents in TREC 1, 2, 3 CDs are divided in 100 non-intersecting collections of roughly equal sizes (about 30 MBs each), organized by source and publication date. The contents of the collections are somewhat heterogeneous.

A second approach for partitioning the TREC data is to use a clustering algorithm on the documents, in order to create topically similar groups of documents. Such a partitioning has been provided by the "Trec4-kmeans" partitioning where the documents from TREC-4

[2]For a complete reference of the data used in each of the TREC conferences, see http://trec.nist.gov/data.html

were clustered by a two pass K-means algorithm using the Kullback-Leibler divergence as the distance metric[3]. The partitioning was originally presented by Xu and Croft (1999) and has also been used extensively by Si, Jin, Callan, and Ogilvie (2002), Shokouhi (2007) and others:

- **Trec4-kmeans ("K-means"):** The documents from TREC-4 are divided in 100 non-intersecting collections by a k-means clustering algorithm. The collections are very homogeneous and the word distribution is very skewed.

A potential disadvantage of the above partitions is the somewhat limited size distribution of information sources. In the Uniform partitioning all collections are roughly the same size, although the number of documents contained in each differs significantly. Although the Trec4 offers a somewhat more realistic environment for DIR experiments, where information sources are topically partitioned, the sizes of the collections remain relatively moderately skewed. Real world DIR applications, such as those presented by Avrahami, Yau, Si, and Callan (2006) report that in realistic environments the distribution of sizes of the collections tends to be much more unbalanced with a significant number of small information sources and a limited number of sources an order of magnitude larger. Three more testbeds with asymmetric collection sizes and distribution of relevant documents were created from the Uniform testbed, in order to provide a more complete simulation of realistic environments and have also been extensively used by Si and Callan (2005), Shokouhi (2007) and others:

- **Trec123-2ldb-60col ("Representative"):** The collections in the Uniform testbed are organized in alphabetical order and every fifth collection starting with the first is merged into a single collection. The same process is also applied starting with the second collection, resulting in two large collections of about 650 MBs each and 60 smaller ones of about 30 MBs each. All the collections have the same density of relevant documents. The Representative partitioning aims to simulate environments where a limited number of large collections contain a significant number of relevant documents, and a larger number of smaller collections also contain some relevant documents. It should be noted that the large collections are an order of magnitude bigger in size than the small ones. A likely scenario for such an environments would

[3]A complete definition of the all the testbeds is available at `http://boston.lti.cs.cmu.edu/callan/Data/`.

be a DIR system that uses a set of hidden web sites with heavily unbalanced sizes and the relevant documents are diffused among the information sources.

- **Trec123-AP-WSJ-60col ("Relevant"):** The 24 Associated Press and the 16 Wall Street Journal collections are merged into two large collections. Those two collections are both larger in size than the rest of the collections (the former being about 750 MBs and the latter about 520 MBs) and hold most of the relevant documents. The Relevant testbed simulates an environment where all of the relevant documents are contained within a limited number of significantly larger collections. Such a scenario can be envisioned for a DIR system that uses both general-purpose search engines and hidden web information sources and the relevant documents for a query are all contained within the indexes of the general purpose search engine or the largest hidden web information sources.

- **Trec123-FR-DOE-81col ("NonRelevant"):** The 13 Federal Register and 6 Department of Energy collections are merged into two collections. Although these collections are much larger in size than the rest, they have very few relevant documents. The NonRelevant testbed has a twofold function: First of all, it functions as a baseline to ensure that the designed source selection algorithms are not heavily influenced by the collection sizes (ranking the largest collections first regardless of whether they contain any relevant documents) and secondly to simulate environments where the largest collections contain no relevant documents at all.

The advantages of all of the above testbeds is that they offer qualitative content in a way similar to authoritative web resources, such as hidden web or enterprise resources. In addition to that, each one of the collections in the testbeds contains a significant number of documents, effectively creating nontrivial content-oriented clusters.

Their drawback is that they are artificially made, not web-based and they both offer the same limited degree of distribution of $O(100)$. In order to better evaluate the proposed algorithms presented in this dissertation, another more natural testbed had also to be manufactured, ideally one that is both web-based and presents a more natural separation of collections. The GOV document corpus offered a potential candidate for the above task. The collection is comprised of a crawl in early 2002 of the .gov domain, which is used by government entities residing in the United States of America. It contains 1.25 million documents in textual form, either from .html pages or extracted from .doc, .pdf and .ps files[4].

Despite the apparent suitability of the test collection, there were significant reasons that hinder its usage in DIR experiments. In order to understand those reasons, a definition of a TREC *Task* (also known as *Track*) needs to be provided: Tracks are used in TREC in order to define the general purpose of the retrieval process. For example, the TREC 2006 Legal Track focused on a specific aspect of retrieval in the legal domain, that of meeting the needs of lawyers to engage in effective discovery of digital document.

The GOV collection has been used in two Tasks in TRECs 2002 to 2004. The *topic distillation* task is a retrieval task in which the goal is to retrieve "key resources" rather than relevant documents and the *named-page finding* task is a known-item task where the goal is to find a particular page that has been named by the user. Therefore, the available topics and relevant judgments available for the GOV corpus are suited for the above tasks. Nonetheless, most of the research in DIR, with the exception of the work by Hawking and Thomas (2005), has focused on *adhoc* tasks, whose general purpose is to retrieve relevant documents, and not topic distillation or named page tasks, therefore it was deemed that the use of the GOV test collection would be inappropriate, despite the obvious suitability of the document corpus itself.

The GOV2 collection was also a potential candidate for conducting DIR experiments. The collection is a second, deeper crawl of the .gov domain that took place in early 2004, containing 20 times more documents that the original GOV corpus, to a total of 426 GBs[5]. The GOV2 testbed does have adhoc queries but its focus is more on efficiency issues (such as indexing speed and retrieval response time), rather than effectiveness (i.e. retrieval quality). Other more practical reasons were also present, as the collection is provided at a specific fee[6], therefore posing practical obstacles from incorporating it into the experiments presented here. It should be mentioned here that Shokouhi (2007) engineered a DIR testbed from the GOV2 corpus by selecting the 100 largest servers, in terms of number of crawled

[4]More information on the GOV test collection can be found at http://ir.dcs.gla.ac.uk/test_ collections/govinfo.html

[5]More information on the GOV2 test collection can be found at http://ir.dcs.gla.ac.uk/test_ collections/gov2-summary.htm

[6]For a list of prices for the various test collections, see http://ir.dcs.gla.ac.uk/test_ collections/access_to_data.html

pages. Although the above DIR testbed is an order of magnitude larger than the TREC collections, it still offers the same limited degree of distribution of 100 collections.

An already available test collection was therefore utilized, namely the WT10g test collection (Bailey, Craswell, and Hawking 2003). The WT10g collection is a 10GBs crawl of the Web conducted in 1998, containing approximately 1.7 million documents. Based on the WT10g corpus, a testbed appropriate for Distributed Information Retrieval experiments was engineered:

- **WT10g-1000col-byUrl ("Web"):** The documents from the WT10g collection are divided into 11.653 collections based on their URLs and the 1000 collections with the largest number of documents are selected. The testbed are very diverse both in word distribution as well as in size.

The advantages of the last test collection are numerous. It is web-based, naturally divided into collections as they were created by their authors and offers a much greater distribution than the standard TREC collections, $O(1,000)$. In addition, there already exist appropriate adhoc queries and relevance judgments. Details on the collections are provided in Table **3.2**.

Name	**Number of Collections**	**Size in GB**	**Num of Documents**		
			Min	**Max**	**Avg**
Trec123	100	3.2	752	39,713	10,782
Trec4	100	2.0	300	82,700	5,700
WT10g	1,000	7.5	278	26,505	1,206

Table **3.2** Statistics about the collections

Details about the queries used for each test collection are provided in Table **3.3**. An added advantage of the newly proposed test collection is that it has an average of 2 words per query, which is common for actual web queries, as stated by Jansen, Spink, and Saracevic (2000).

3.4 Simulating Information Sources

Having multiple partitions of documents that are representative of real world applications under different scenarios is only part of the process that is required for building and

Name	**Number of Queries**	**TREC Topic Set**	**TREC Topic Field**	**Average Length of Query Terms**
Trec123	100	51-150	Title	3
Trec4	50	201-250	Description	7
WT10g	100	451-550	Title	2

Table **3.3** Statistic about the queries

experimenting with realistic DIR environments. Information sources are more than repositories of documents. It is assumed that a retrieval system capable of retrieving documents from that repository in response to user queries is also available at each information source. Therefore, a second, just as important factor that had to be considered was the information retrieval algorithm that would be used at the remote collections. Two strategies were possible; it could either be assumed that all the remote collections employ the same algorithm or that those differ.

In order to make the experiments more realistic, the second approach was adopted. Three retrieval algorithms were implemented and were assigned to the remote collections in a round robin fashion. These are: INQUERY (Callan, Croft, and Harding 1992), a language model based on KL divergence (Zhai and Lafferty 2001) and Okapi BM25 (Robertson, Walker, M., and M. 1994). For example, the first, fourth, seventh and so fourth collections, ordered by name were assigned the INQUERY retrieval algorithm, the second, fifth, eighth and so fourth collections were assigned a KL divergence model, etc. The above assignment of retrieval algorithms amongst information sources was adopted for all the aforementioned partitioning of documents. All three algorithms are generally considered very effective, are widely used in IR experiments and are presented in more detail below. They were all implemented using the Lemur Toolkit[7].

The belief function used by INQUERY (Callan, Croft, and Harding 1992) to estimate the belief $p(q|d)$ of query term q within document d is:

$$p(q|d) = b + (1-b) * T * I \tag{3.1}$$

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

where

$$T = \frac{tf_{q,d}}{tf_{q,d} + 0.5 + 1.5 * \frac{length(d)}{avg\_len}} \tag{3.2}$$

and

$$I = \frac{log\frac{N+0.5}{n_q}}{log(N+1)} \tag{3.3}$$

where n_q is the number of documents containing term q, N is the number of documents in the collection, avg_len is the average number of words in documents in the collection, $length(d)$ is the number of words in document d and $tf_{q,d}$ is the number of times term q occurs in document d. Lastly, b is the "default belief", usually set to the value of 0.4. T in equation 3.2 estimates the significance of term q in document d (i.e. how descriptive is the term of the content of the document) and is often referred to as the tf component (in this case, the "Okapi tf" function). I in equation 3.3 estimates the discriminating power of the term in regard to the whole corpus and is often referred to as the *idf* component.

The final belief $p(Q|d)$ of query Q for document d can be estimated using various probabilistic operators. These cover a wide range of Boolean (AND, OR, NOT), proximity and synonym operators. In the context that the INQUERY retrieval engine was used in this dissertation, belief $p(Q|d)$ is calculated as the average of the individual believes $p(q|d)$:

$$p(Q|d) = \frac{\sum_{q \in Q} p(q|d)}{|Q|} \tag{3.4}$$

For more information the reader is pointed to the work of Allan, Connell, Croft, Feng, Fisher, and Li (2000) and Callan, Croft, and Harding (1992). The reader may notice a similarity between equations 3.1 to 3.3 used by INQUERY and equations 2.17 to 2.19 used by the CORI source selection algorithm. As already stated, CORI and INQUERY use the same approach to ranking collections or documents, as CORI considers each collection simply an aggregation of documents. The only noticeable different between the two sets of equations is in the parameters used respectively: INQUERY uses value of 0.5 and 1.5 for the estimation of T, while CORI uses 50 and 150 respectively, proportionate of the increased frequencies of terms within the *super-document* collections.

Language Models for Information Retrieval are based on the idea that each document d is viewed as a multinomial distribution of the words in the vocabulary. The aim of the model therefore is to estimate the probability $p(Q|d)$ of generating query Q given document

d. We used the Kullback-Leibler divergence to measure how well a document d predicts query Q:

$$p(Q|d) = KL(Q,d) = \sum_{q \in Q} p(q|Q) * log\left(\frac{P(q|Q)}{P(q|d)}\right) \tag{3.5}$$

where $p(q|d)$ is the language model for document d, i.e. the probability of generating query term q from document d. Since $p(q|Q)$ is constant for each query and therefore doesn't influence the final ranking of documents, the estimation on $p(q|d)$ is of primary interest for the algorithm. The Lemur Toolkit can estimate $p(q|d)$ using a maximum likehood estimator (MLE), a simple form of Jelinek-Mercer smoothing (Jelinek and Mercer 1985) and others. The former approach will generally underestimate the probability of any word unseen in the document, so the latter estimator, which uses smoothing, was used. The main purpose of smoothing is to assign a non-zero probability to the unseen query terms and therefore improve the accuracy of word probability estimation in general. The Jelinek-Mercer method estimates $p(q|d)$ as:

$$p(q|d) = \lambda * p_{ml}(q|d) + (1-\lambda)_{ml} * p(q|C) \tag{3.6}$$

where

$$p_{ml}(q|d) = \frac{tf_{q,d}}{length(d)} \tag{3.7}$$

and

$$p_{ml}(q|C) = \frac{n_q}{N} \tag{3.8}$$

where λ is the smoothing parameter, set to 0.5 in the experiments that will be presented here, $p_{ml}(q|d)$ is the probability of viewing query term q at document d and is estimated using a maximum likehood estimator and $p_{ml}(q|C)$ is the language model for the whole collection, i.e. the probability of generating query term q from the whole collection C.

The *Okapi BM*25 (Robertson, Walker, M., and M. 1994) retrieval algorithm is considered one of the better models of relevance based upon term occurrences within text documents. According to the model, the relevance weight assigned to document d due to query term q is calculated as:

$$Okapi(Q,d) = \sum_{q \in Q} w \frac{(k_1+1) * tf_{q,d}}{K + tf_{q,d}} * \frac{(k_3+1) * qtf}{k_3 + qtf} \tag{3.9}$$

$$where \;\; w = \frac{N - n_q + 0.5}{n_q + 0.5} \;\; and \;\; K = k_1 * \left((1-b) + b * \frac{length(d)}{avg\_len} \right) \tag{3.10}$$

where as before $tf_{q,d}$ is the frequency of the query term q within document d, qtf is the frequency of q within query Q, N is the number of documents in the collection, n_q is the number of documents containing q, $length(d)$ is the length of the document and avg_len is the average length of all the documents in the collection. $k1$, $k3$ and b are parameters of the model, set to their default values of 1.2, $1,000$ and 0.75, respectively.

3.5 Distributed Information Retrieval Metrics

Various measurements have been proposed in literature for DIR experiments (French, Powell, Viles, Emmitt, and Prey 1998, French and Powell 2000). Recent research in both of the problems of source selection and results merging (Si and Callan 2003b, Rasolofo, Hawking, and Savoy 2003, Si and Callan 2004, Shokouhi 2007, Gravano, Garcia-Molina, and Tomasic 1999) has focused on two particular measures, so they are also used here.

Recall is concerned with the comprehensiveness of the retrieval result and is defined as the fraction of the available relevant documents that have been retrieved. Nonetheless, *recall* as it is defined above is problematic in Distributed Information Retrieval environments since only a subset of the available collection sources is selected and thus it is generally impossible to retrieval all relevant documents.

French and Powell (2000) proposed a new *recall* analogue measure R_n for evaluating source selection algorithms, which has been widely adopted in the research community (Callan, Lu, and Croft 1995, Nottelmann and Fuhr 2003a, Si and Callan 2005). R_n compares a given source selection algorithm at rank n to a *desired* source selection algorithm at the same rank. The desired algorithm, is a retrospective baseline algorithm that ranks information sources according to the number of relevant documents they contain for a query. This ranking is also known as relevance based ranking (RBR) or *optimal* ranking. Formally, R_n can be defined as:

$$R_n = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n} B_i} \tag{3.11}$$

where E is the collection ranking provided by the source selection algorithm under investigation and B is a baseline ranking. For example, suppose that a distributed setting contains four collections, $\{C_i\}_{i=1,2,3,4}$ and for a particular query each contains $5, 20, 0, 10$ relevant documents respectively. The optimal ranking therefore would be $B = \{C_2, C_4, C_1, C_3\}$. Supposing a source selection algorithm that produces the following ranking $E = \{C_4, C_2, C_3, C_1\}$, the value of R_n for various values of n is: $R_1 = \frac{10}{20} = 0.5$, $R_2 = \frac{30}{30} = 1$, $R_3 = \frac{30}{35} = 0.857$, $R_4 = \frac{35}{35} = 1$.

Usually, the goal is to achieve the maximum effectiveness searching only few collections, so in the context of the experiments that will be presented here, where the available collections are $O(100) - O(1000)$, results are reported for selecting up to 20 collections ($n \leq 20$). R_n is a particularly important measure for source selection applications addressing the *high-recall* goal, as it was defined in paragraph 2.3, as it provides a clear indication as to whether they are able to rank collections with a significant number of documents higher than others.

Precision is one of the most widely used measure for adhoc experiments and has been extensively used for measuring the performance of results merging algorithms primarily and source selection algorithms secondly. It can be defined as the faction of the retrieved documents that are relevant. Since users rarely look past the top 20 results (Jansen, Spink, and Saracevic 2000), we report precision measurements up to that rank, at the top 5, 10, 15 and 20 documents (noted as P@5, P@10, P@15 and P@20 respectively). Formally:

$$P@N = \frac{|Relevant\ Documents\ in\ top\ N\ Ranks|}{N} \tag{3.12}$$

Additional measurements like *F-Measure, 11-point Point Precision or Average Precision* (Manning, Raghavan, and Schütze 2008) that combine precision and recall into a single evaluation, giving an overview of the system in regard to both of the above facets of the retrieval, accuracy and comprehensiveness, have also been presented in centralized IR but haven't been adopted to a distributed setting that is the focus of this dissertation. The choice of the above measures has been dictated by their extensive usage in evaluating results merging and source selection algorithms in research (Si and Callan 2003b, Rasolofo, Hawking, and Savoy 2003, Shokouhi 2007, Gravano, Garcia-Molina, and Tomasic 1999, Callan 2000).

CHAPTER 4

Collection-Integral Source Selection

4.1 Introduction

The novel source selection algorithm that is presented in this chapter of the thesis explicitly focuses on both of the tasks of the source selection process, the *high-recall* and *high-precision* goals, without requiring any form of training, making it applicable across a wide setting of applications and environments. The algorithm accomplishes the above objectives utilizing a novel and simple metric for estimating the relevance of information sources by modeling each collection as an integral in the rank - relevance space produced by its documents, using the relevance score and the intra-collection rank of its sampled documents. In order to achieve the former goal, the algorithm selects the collections that cover the largest area in the space. For the latter goal, the algorithm divides the area covered by the remote collections into segments, each representing an estimation of the relevance of the returned lists of document of the collections in regard to the submitted query, and thus estimates the benefit of including them in the merged list. Based on that estimation, the optimal distribution of documents is calculated in order to maximize the relevance of the final document list that will be returned to the user and therefore maximize the overall gain.

An important novelty of the new algorithm is that it relaxes the necessity for a random sampling and a per-query uniform distribution of document scores, in contrast to previous approaches, such as ReDDE, UUM and RUM (see chapter 2.3.2), by utilizing the *scale factor* SF_i (equation 2.24) only once per collection and avoiding making assumptions about the unsampled documents of the collections. Therefore, it is less susceptible to the effectivenesses of the sampling process and is overall more suited to provide estimations about the potential relevance of the unsampled documents of the remote collections.

The algorithm departs from traditional practice in DIR in an additional manner. Previous approaches in DIR have relied on the number of collections that will be queried as well as the number of documents that will be retrieved from each being pre-specified, usually by the user. Since in most cases, information concerning the actual distribution of relevant documents amongst remote collections is unavailable (hence the need for source selection), such decisions are usually based on arbitrary rules. The algorithm that is presented here requires that only one decision is made: the number of documents in the final merged list. The decision on how many collections to query and how many documents to retrieve from each is determined by the algorithm, through a dynamic programming process. As it will be shown, although the system receives the minimum information, the results that are produced are equivalent or better to those produced by more information-demanding systems. Although no user-centered studies were conducted as part of this thesis, it is believed that such a facility would be greatly appreciated by users, since it would alleviate the need to heuristically specify the parameters of the retrieval process.

4.2 Integral based Source Selection

4.2.1 Collection-integral Source Selection for High Recall

In order for the algorithm to function, a *centralized sample index*, which is comprised of all the sampled documents indexed together, is built. As in previous approaches discussed in section 2.3, the centralized sample index is considered as a representative of the single global index that would be created if all the documents were available for indexing, providing estimates of global collection statistics (such as global idf), necessary for the estimation of collection-independent document relevance scores. Briefly, a "collection-independent" score can be defined as the score that a document would obtain if the query was submitted to a centralized index, containing all the documents that are scattered at the remote collections in the distributed environment. For a more thorough definition and analysis, please refer to chapter 2.4.2.

The query is directed to the centralized sample index and a list of documents with relevance scores is returned. The INQUERY retrieval algorithm (Callan, Croft, and Harding 1992) is used to query the centralized sample index, but any effective algorithm can be applied. The returned documents are assigned to the collections from which they were originally sampled, creating a vector of scores for each collection i: $v_i = \{score_1, \ldots, score_{n_i}\}$, where n_i is the number of retrieved documents originating from the sample of collection i.

Each collection is represented as a plot, using the scores and the ranks of the returned documents as points: $\{(1, score_1)$, $(2, score_2)$,..., $(n_i, score_{n_i})\}$. The ranking of the returned documents in the centralized sample index is discarded, in contrast to Shokouhi (2007) and Si and Callan (2003a) and the ranking of documents only in reference to other intra-collection documents is utilized. By making usage of the relevance scores of the documents in reference to the centralized sample index and not of their rankings, the algorithm is able to retain intricacies in scores, which are not apparent in ranking. For example, a difference of one in ranking (i.e. 2 and 3) may be due to a minimal (i.e. 0.63 and 0.62) or more substantial (0.6 and 0.5) difference in scores. Using scores instead of rankings, makes the algorithm more "sensitive" to such subtle differences.

Incorporating collection sizes

At this point, the estimates for the size of the collections are incorporated into the algorithm. Previous research (Si and Callan 2003a, Shokouhi 2007) has indicated that using the estimated size of the collections improves effectiveness, especially in environments with skewed collection sizes.

The approach followed here is based on the following assumption. Supposing that collection i has n_i documents returned in the centralized sample index w.r.t. a query, then it is assumed that there should be about $\frac{n_i}{N_{i\_samp}} * \hat{N}_i = n_i * SF_i$ documents returned if the query is directed to the actual remote collection, where SF_i is the *scale factor* of collection i, originally defined in equation 2.24. An example will make the above clearer. Suppose a collection of estimated size of 3000 documents from which 300 documents have been sampled, therefore $SF_i = 10$. If the sample of the collection returns 50 documents w.r.t a query, it is assumed that if the query is directed to the actual remote collection there would be about $\frac{50}{300} * 3000 = 500$ documents returned. Note that no assumption about the relevance of the unsampled documents or the distribution of their scores is made, only about the overall number of returned documents.

Additionally, the relevance score of the last retrieved document is assumed to be close to zero. Taking into consideration that the SF_i is $O(100) - O(1000)$ in most cases, it is safe to suppose that the last document that is retrieved from a collection would most likely have a very low relevance score, therefore justifying the above score. The exact value of the score doesn't influence the performance of the algorithm.

The accuracy of the above hypothesis, impacts the effectiveness of the algorithm in only a limited manner. Preliminary experiments, using the actual collection sizes and estimations, showed that the algorithm remains robust and exhibits only small variations in performance across a range of errors concerning the estimated collection sizes.

Based on the above, a last tuple is inserted at the tail of each collection plot, as document $n_i + 1$, at point:

$$\left(\frac{n_i}{N_i\_sample} * \hat{N}_i, 0 \right)$$

The above assumptions and coordinates serve a very specific purpose, that of introducing the estimated sizes of the collection into the algorithm, without making any assumptions about the unseen, unsampled documents. Previous approaches in source selection (Si and Callan 2003a, Shokouhi 2007) heuristically consider the top N documents returned from the centralized sample index to be relevant and disregard the rest as irrelevant. The same modeling of relevance is also utilized by most automatic relevance feedback methods. In order to avoid using a step function to simulate the relevance of the documents retrieved from the remote collection, a score close to zero is assigned to the last document, in effect slowly and gradually reducing the probability of relevance of the potentially retrieved documents from the remote collection.

Taking into consideration that, as discussed above, the sampling is usually non-random and the estimates of the collections sizes are less than accurate, with a mean-average error ratio of 0.23 at best, as reported by Si and Callan (2003a), the algorithm is able to function effectively even in environments where the "initial conditions" for source selection, i.e. the sampling of the remote collections and the estimation of their sizes, are less than perfect. The experiments, reported in section 4.5, demonstrate that the hypothesis indeed does hold some merit.

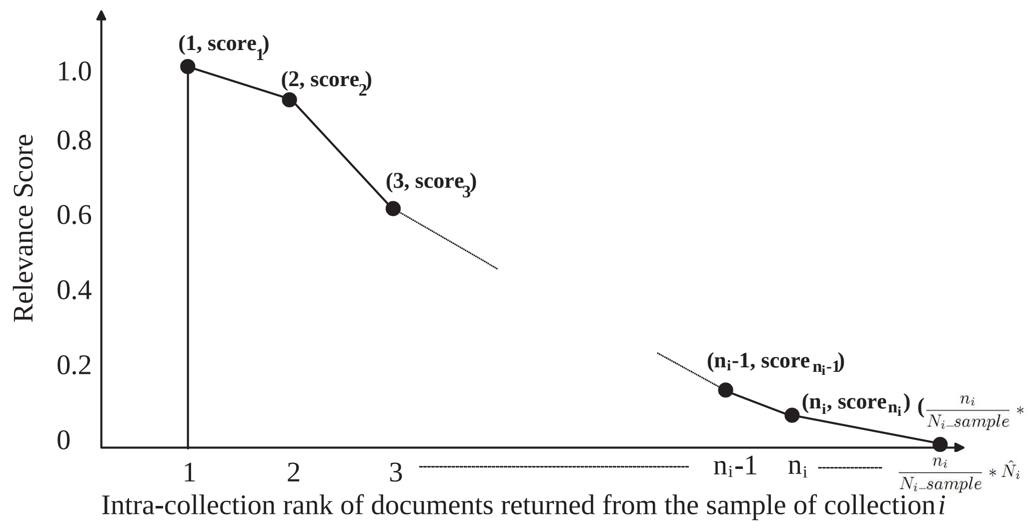


Figure 4.1 Initial plot of documents.

Definition of interpolant line segments

Next, the family of linear interpolant functions for each collection i that pass through the points (Figure 4.1) is defined as follows:

$$F_i\,(j \leq x < j+1) = score_j + (x-j)\,(score_{j+1} - score_j) \tag{4.1}$$

where $j = 1, 2\ldots, n_i$. The above concise form allows the determination of the value of $F(x)$ in each of the $[j, j+1)$ intervals (for example $F(2.5) = score_2 + (2.5-2)(score_3 - score_2)$). Although it is possible to fit a single higher-degree function through the points via interpolation, applying a single curve unavoidably results in diminished accuracy, therefore a family of linear functions was chosen that is guaranteed to pass through all the points in the plot. The idea and motivation behind the use of the plot is to have a first estimate about the relevance of the documents in the sample collections in regard to the particular query. One would expect that a collection that is a good candidate for selection to have certain properties, such as a nontrivial number of documents in the plot and at least some of them with high relevance scores.

Applying modifications to the initial collection plot

The above intuitions are enhanced and quantified by applying two transformations to the initial plot. The transformations involve assigning different weights to the documents depending on the ranking and the relevance score they attained. The process of applying different weights to the returned documents has been a tested practice in DIR (Si and Callan 2003a, Shokouhi 2007) as well as in other fields, such as metasearch (Aslam and Montague 2001) or expert finding (Macdonald and Ounis 2006). Here, this practice is transferred to the appropriate modeling of the collections as plots.

First, the original tuples are transformed in the following manner: $(j, score_j) \rightarrow (log_m j, score_j)$. An example of this transformation is shown on Figure 4.2. As it can be observed, the interval between the top left-most points in the plot is increased, while it is steadily decreased as the values in the x-axis increase. The reason and the effects of the transformation will become apparent when the final collection-ranking function is defined, but the overall purpose is twofold. First of all, it increases the area covered by the top, high-ranking documents and gradually decreases it for the ensuing documents. That is because high-scoring documents mark higher in the y axis and with the above transformation the segments that they cover on the x axis also increase, effectively augmenting the area that they cover in relation to documents with lower scores, thus promoting the collections they belong. This way, collections that have documents with high scores are rewarded in comparison to collections that have average-scoring documents. The property is both important and desirable since collections that have at least some high-scoring documents are expected to be more appropriate for selection than others that have mainly documents with average scores. Additionally, the above transformation dampens the effect of collections returning

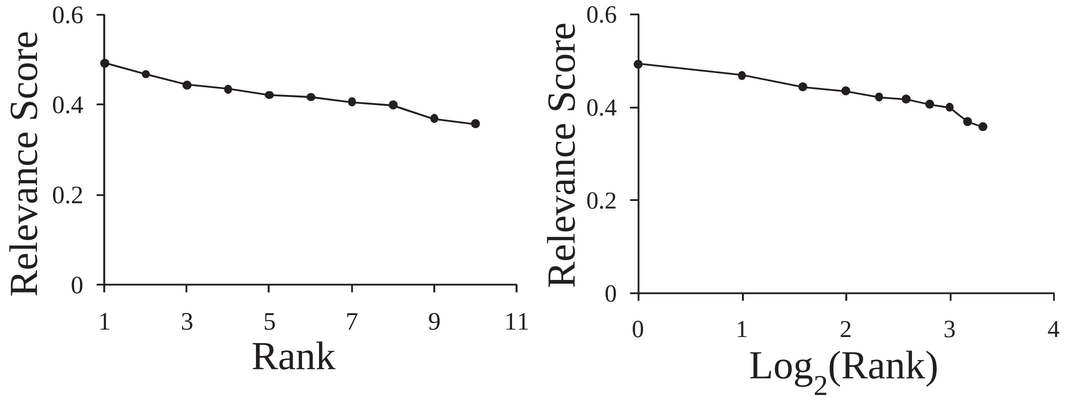


Figure 4.2 An example of $(j, score_j)$ to $(log_2 j, score_j)$ transformation

a large number of documents, effectively moderating their contribution to the area covered by the collection. Although a large number of documents may be indicative of the appropriateness of the collection for selection, the expected correlation isn't proportional (i.e. a collection that returns 30 documents isn't necessarily twice as relevant as one returning only 15[1]).

The level of the reward that is granted to collections with high scoring documents and the degree of the dampening effect both depend on the base of the logarithm (the m value) and is left as a parameter of the model.

Secondly, the scores of the documents are transformed using the exponential function (e^{score_i}), as suggested by Macdonald and Ounis (2006) and Ogilvie and Callan (2003). The Lemur Toolkit[2] that was used for the experiments returns the log of the probability of documents (Ogilvie and Callan 2001), therefore applying the above transformation returns the score to the probability scale. The above transformation has the additional effect of further boosting the scores of high-scoring documents and has been previously used for expert finding tasks (Macdonald and Ounis 2006) and known-item search (Ogilvie and Callan 2003).

Summarizing the two transformations $(j, score_j) \rightarrow (log_m j, e^{score_j})$ and applying them to the initial family of interpolant functions $F_i(x)$ for collection i (equation 4.1), it is being transformed to $D_i(x)$:

$$\begin{aligned} D_i(j \leq x < j+1) &= e^{score_j} + \\ &+ (log_m x - log_k j) * (e^{score_{j+1}} - e^{score_j}) \\ &= e^{score_j} + log_m \frac{x}{j} * (e^{score_{j+1}} - e^{score_j}) \end{aligned} \tag{4.2}$$

It is now possible to define the ranking function $R(i)$ for collection i, on the basis of which collections are ranked:

$$R(i) = \int_1^{n_i+1} D_i(x)dx \tag{4.3}$$

which can intuitively be defined as the area between the transformed function $D_i(x)$ and the horizontal axis x.

[1]The same principal is also applied in classical information retrieval (i.e. Okapi BM25 (Robertson, Walker, M., and M. 1994)), where a document in which a query term appears twice as many times as another document isn't considered to be twice as relevant.

[2]`http://www.lemurproject.org`

An additional innovation of the above modeling is that it moves away from the summative approach that has been followed in research (i.e. Shokouhi (2007), Si and Callan (2003a), etc), where the score of each collection is the *sum* over a function of rank (or relevance) of its sampled documents, instead forming and implementing a new metric, such as the area covered by documents in the rank - relevance space. Of course, it may be argued that integral is also summative in nature, based on continues space instead of discrete as previous approaches, in which case the present approach represents a first attempt of estimating the appropriateness of a source for selection in continuous space, rather than discrete.

The newly proposed algorithm was named *Collection-integral Source Selection* (CiSS). The algorithm as it was presented as far, that selects the collections that cover the largest area in the rank - relevance space, was utilized to address the high-recall problem and was therefore named CiSS-HiRe (CiSS for *High Recall*).

4.2.2 Collection-integral Source Selection for High Precision

Previous research by Craswell (2000) and more recently by Si and Callan (2003a) has shown that the high recall and the high precision goals are often not attainable without special consideration for the specifics of each particular goal, especially in testbeds with skewed collection sizes and distribution of relevant documents. Si and Callan (2004) for the first time applied a clear distinction between the two related but not necessarily equivalent goals and the source selection problem was explicitly differentiated. Recent research on source selection (Si and Callan 2005, Shokouhi 2007) has focused on the high-precision goal, considering the high-recall goal of secondary importance.

Addressing the high-precision goal requires that the CiSS algorithm focuses on maximizing the area that is covered by segments of the collection integrals, each representing an estimation of the relevance of the returned documents lists from the remote collections, thus estimating the potential benefit of including their top ranked documents in the final merged list, rather than straightforward ranking the collections by the area that they cover.

The basis of the above estimation is the modeling of the collections as integrals in the transformed rank - relevance space, as it was presented above. Rather than computing the space covered by the whole plot, as it was done for CiSS-HiRe, the integral is divided into smaller segments, each one representing the benefit of including the top ranked documents from the collection in the merged list. An example of such a segmentation is given on Figure 4.3. For readability reasons, a segmentation of the initial plot is presented, but the algorithm is based on the transformed plot (section 4.2).

In order to simplify calculations, it is assumed that remote collections return documents at lists of ten documents per page, since most commercial search engines by default provide results in lists of ten documents. The methodology presented can also be applied to lists of greater granularity (i.e. three, five documents per result page).

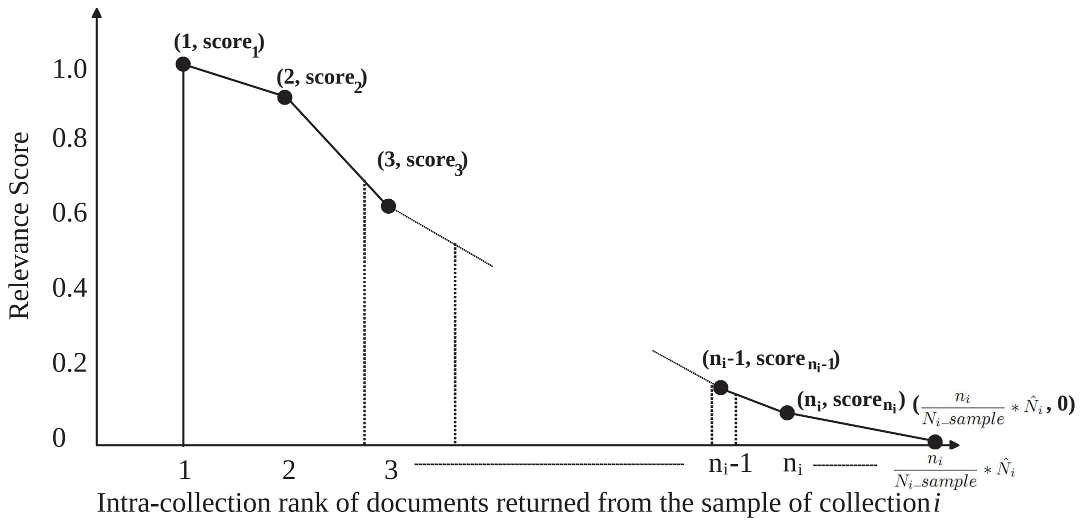

(1, score1)
(2, score2)
(3, score3)
(ni-1, scoreni-1)
(ni, scoreni)
$(\frac{n_i}{N_i\_sample} * \hat{N}_i, 0)$
Relevance Score
1.0
0.8
0.6
0.4
0.2
0
1
2
3
ni-1
ni
$\frac{n_i}{N_i\_sample} * \hat{N}_i$
Intra-collection rank of documents returned from the sample of collection i

Figure 4.3 Splitting the collection plot into segments.

One observation that needs to be made here is that the segments are not of equal length on the x-axis, as one might expect. In fact, the segments are wider at the top ranks and slowly decrease as the x values increase. The fact derives from the observation that most of the relevant and therefore useful documents that a collection will return for merging will be located at the top ranks. Experiments presented in chapter 6.4 indicate that increasing the number of requested documents from the remote collections, doesn't necessarily result in an increase in precision in the final merged list. Accordingly, it is assumed that most of the gain of selecting a particular collection and including its documents in the final merged list, derives from its top-ranking documents.

The actual decrease of gain is open to question and depends on a variety of factors, such as the the actual number of relevant documents the remote collection contains, its size and potentially the effectiveness of the retrieval algorithm employed. Previous studies

(Jansen, Spink, and Saracevic 2000, Shokouhi 2007) have indicated that the probability of relevance of a document and therefore the gain of retrieving it, has an negative exponential or logistic relation with its rank. Based on the above studies, a rough division of the x-axis for collection i is designed based on the following formula:

$$H_i(x) = \frac{e^{\alpha-\beta x}}{1+e^{\alpha-\beta x}} * log_m \left[\frac{n_i}{N_i\_sample} * \hat{N_i} \right] \tag{4.4}$$

where $\alpha = 2$ and $\beta = 0.75$; x is the cardinality of the result page $(1^{st}, 2^{nd}, \dots)$ and represents the general gain of observing/retrieving the documents in the x^{th} result page of the remote collection. The contents of the log_m of $H_i(x)$ is the x-coordinate of the $(n_i + 1)$ document, hereafter noted as x_{n_i+1}. The above function represents a rough estimate of the decrease of relevance of the returned documents from the remote collections.

The above function has the significant drawback that the series of the first part (the sum of its values for $x = 1, 2, \dots$) is greater than 1, meaning that the segments at the x-axis would extend beyond the last point $(n_i + 1)$ of the collection plot. In order to keep the slope of the decrease but remedy the above problematic feature, the original $H_i(x)$ is devided by γ, therefore $H_i^*(x) = \frac{H_i(x)}{\gamma}$. In the experiments that were conducted γ was set to 4.15. More information on how the parameters α, β and γ are set is given in paragraph 4.4.

It is notable that the series of $H_i^*(x)$ is less than $1 * log_m x_{n_i+1}$ indicating that a part of the collection integral will not be considered in the calculation. The above means that some documents belonging to the remote collections will never be considered for retrieval and merging. The conclusion is in accordance with the results presented by Shokouhi, Zobel, Scholer, and Tahaghoghi (2006), where it is reported that approximately 10% of the webpages in the WT10g collection (Bailey, Craswell, and Hawking 2003) are not retrieved during the extraction of one hundred query-based samples, indicating that some documents may be unretrievable, at least by few-term queries. A more recent study by Azzopardi and Vinay (2008), studying the phenomenon of document accessibility in IR systems also suggests that some documents may be unretrievable by search engines, unless the actual query is the document itself.

Note that the produced decline is dependent only on the number of returned sampled document for collection i and its estimated size. A more fine-tuned definition of the above decline for each collection i, i.e. dynamic estimation of parameters α, β and γ, is left as future work and is further discussed in chapter 7.

The function $G_i(x)$, that calculates the area of the x^{th} segment of the i^{th} collection and thus estimates the gain of retrieving it, is therefore defined as:

$$G_i(x) = \int_{H_i^*(x)}^{H_i^*(x+1)} D_i(x)dx \tag{4.5}$$

The vector $gainV_i$ that will contain the estimated gains $G_i(x)$ for each collection i can now be defined as:

$$gainV_i = \{G_i(1), G_i(2), \dots\} \tag{4.6}$$

Each element k of the above vector represents the gain of including the k^{th} result page from the i collection into the final merged list.

Having generated the vector $gainV_i$ for each collection i, the algorithm proceeds to a dynamic programming solution, in order to estimate the optimal distribution of documents from remote collections that maximize the area covered by the collection segments. The process takes as input the $gainV_i$ vectors for all the available collections and the number of requested documents in the final merged list. The purpose of the dynamic programming is to maximize the gain (i.e. overall area covered) of the final merged list. The problem is non-trivial and is viewed as a typical "optimal allotment problem" (Lew and Mauch 2006) with a limitation on the number of users. The implementation of the solution be either be recursive, which is very inefficient in regard to the time necessary to locate the optimal solution or through storing the values of each iteration, until the maximum value and the optimal distribution is calculated under the pre-specified limitation, which was the preferred way in the experiments that were conducted.

The output of the process is a final vector $gain = \{gain_1,\ gain_2, \dots\}$, each element of which represents the number of documents that will be requested from the respective remote collection for merging. For example if $gain = \{10, 0, 20, 0,\ \dots, 0\}$ then the algorithm will request 10 documents from the first collection, 0 from the second, 20 from the third and so fourth.

The version of the algorithm that aims to solve the high-precision problem was named CiSS-HiPre (CiSS for *High Presision*).

4.3 The advantages of using an integral-based metric for source selection

Modeling information sources as integrals in the rank - relevance space of their documents offers a significant number of advantages. First of all, it provides with an intuitive notion of the appropriateness of the remote document collections in respect to a query; the area that is covered by their plots in the rank - relevance space. The larger the area, the more appropriate the collection is to be selected.

Secondly and most importantly, using integrals as indicators of appropriateness has the significant advantage of providing a formal way of explicitly addressing the two tasks of source selection, high-recall and high-precision, within the same framework. The entire integral is considered of each collection for the former task, as that provides an indication of the relevancy of the whole collection in regard to the submitted query. For the latter goal, the integral is divided into parts, each one representing the relevancy of the returned result list of the collection and therefore the gain of including them in the final merged result list. The dynamic programming solution is therefore naturally incorporated into the algorithm in order to maximize the overall gain and thus select the collections that provide most of their gain early in their result pages, as it is desirable in order to achieve a high-precision merged list.

An important aspect of the suggested modeling is that it captures real-world observations and widely accepted notions in IR, i.e. such as the non-uniformity of the sampling, the existence of unretrievable documents and the importance of top ranking documents for precision into a theoretical framework by applying simple modifications to the initial collection plot. Potentially, further modifications may be applied in order to model additional experiments results, thus on the one hand, increasing the effectiveness of the algorithm and on the other, producing an even better modeling of collection appropriateness for the task of source selection.

The modeling of the collections as it has been presented and implemented within the limits of this thesis only considers one aspect regarding the appropriateness of the collection for selection; their contents. In a realistic Distributed Information Retrieval environment, nonetheless, the contents of the remote information sources is only one of many potential indicators of selection. Others may include: average information source time of response (slow responding sources should be penalized), occurring costs from submitting a query to specific sources, authoritativeness of sources etc. Such notions can also be incorporating into the proposed modeling, thus producing an integral not in $2-$dimensional space, as it was done in the context of this thesis, but an integral in n-dimensional space, where $n-1$ would be the number of factors under consideration. Therefore, one additional advantage of the above modeling is that it easily allows for such extensions, as they are considered important, depending on the specific characteristics of the scenario under which the DIR system would be employed.

4.4 Experiment Setup

In order to avoid any interference from the results merging algorithm in precision oriented environments, where the goal is to produce a high-precision merged document list (see section 2.3.1), all the retrieved documents from the remote collections are downloaded locally and re-ranked, using the centralized sample index as a "reference statistics database" (Craswell, Hawking, and Thistlewaite 1999, Si and Callan 2003b). The above methodology may not be the most efficient, but it is the most effective, as indicated by Craswell, Hawking, and Thistlewaite (1999) and is also affirmed later by experiments conducted as part of this thesis, in chapter 6.4. The above decision was made to minimized any "noise" introduced in the process by the merging algorithm and present the source selection algorithms under the best attainable performance. Alternatively, SSL could be utilized but previous research by Si and Callan (2004) and again verified in this thesis in chapter 6.4 have indicated that the algorithm performs optimally only when the collections return a significant number of documents (i.e. 100), which would result in an artificial deteriorated effectiveness of CiSS-HiPre in environments where only few documents are requested from a collection. Note that the downloaded documents can afterward either be kept at the centralized sample index, practically updating it or alternatively be deleted. In the line of experiments that were conducted, the second approach was adopted so that previous queries would not effect the results of subsequent queries. Since users rarely look past the top 20 results (Jansen, Spink, and Saracevic 2000), precision measurements are reported up to that rank.

Although no single value of the parameter m for CiSS is optimal for every testbed and setting, preliminary experiments showed that the algorithm is robust across a range of parameter values ($1.0001 - 10$). The value of m was set to 1.07 which provided with a good enough (though not optimal) performance across all testbeds. Queries $51 - 100$ at the Uniform testbed were used for optimizing parameters α, β and γ using MAP as an optimization measure and the estimated values ($\alpha = 2$, $\beta = 0.75$ and $\gamma = 4.15$) were used at every conducted experiment. As an optimization method, the performance of the algorithm was evaluated on a successively smaller grid over the set of parameters.

A significant difficulty in comparing source selection algorithms in precision-oriented environments is that CiSS-HiPre receives as input only the number of requested documents, dynamically estimating the number of collections that are selected as well as the number of documents that are retrieved from each. In order to present all the algorithms under the best possible light and in as equal terms as possible, the following settings are applied: CiSS-HiPre is set to retrieve 100 documents in total and the rest of the source selection algorithms, including CiSS-HiRe, are set to select seven collections each one returning twenty documents. Even though the above setting results in CiSS-HiPre retrieving less documents in total for merging (100 versus 140), potentially it can contact more information sources for merging, to a maximum of ten. The motivation between the above setting is that when CiSS-HiPre is set to retrieve 100 documents, on average it was found to contact seven collections. Studies by Shokouhi (2007) and Avrahami, Yau, Si, and Callan (2006) have shown that increasing the number of collections doesn't necessarily result in an increase of retrieval quality so the above settings present all the algorithms under the best possible conditions.

As already discussed, query-based sampling (Callan and Connell 2001) was used in order to create representatives for the remote collections, sending 75 one-word queries and downloading the first 4 documents, obtaining approximately 300 documents per collection. The above process may not produce optimal representatives as noted by Thomas and Hawking (2007), but has become standard practice when evaluating source selection algorithms (Shokouhi 2007, Si and Callan 2004, Si and Callan 2005). In order to estimate the size of the remote collections the sample-resample technique by Si and Callan (2003a) was utilized.

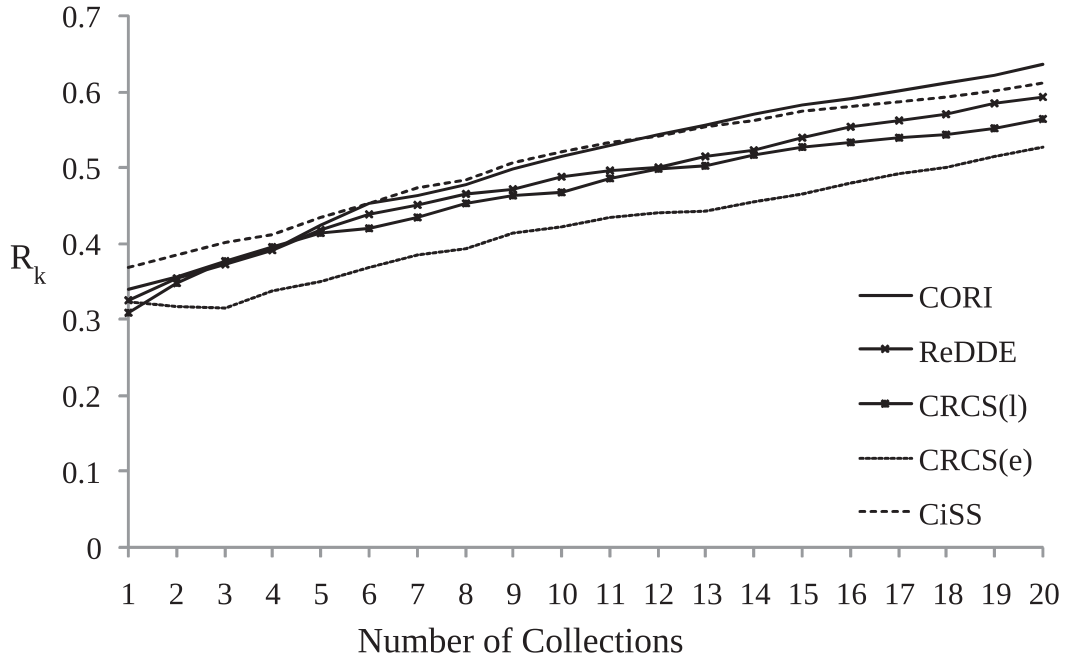


Figure 4.4 R_k values for the Uniform testbed.

4.5 Results

The results of the experiments in recall-oriented settings are presented in figures 4.4 to 4.9 and in precision-oriented settings in tables **4.1** to **4.6**.

In the Uniform testbed (figure 4.4), most of the algorithms perform similarly. The relative uniformity of the distribution of relevant documents at this testbed and the lack of complete information about the collections, make the goal of locating the collections that have the greatest number of relevant documents very difficult. CiSS-HiRe is able to show improvements over other approaches at settings of up to six collections, while performing equally to CORI from that setting and on.

This first testbed and experiment provides an initial indication of the effectiveness of the new modeling of information sources. Despite the inherit difficulty of the testbed for the aforementioned reasons, CiSS-HiRe is able to outperform previous state-of-the-art approaches, especially when a limited number of collections are considered, which is considered as more important in a significant number of settings. The first results therefore are rather encouraging that the above modeling does have some advantages.

The information sources at this testbed are very uniform in byte-size (hence the name of

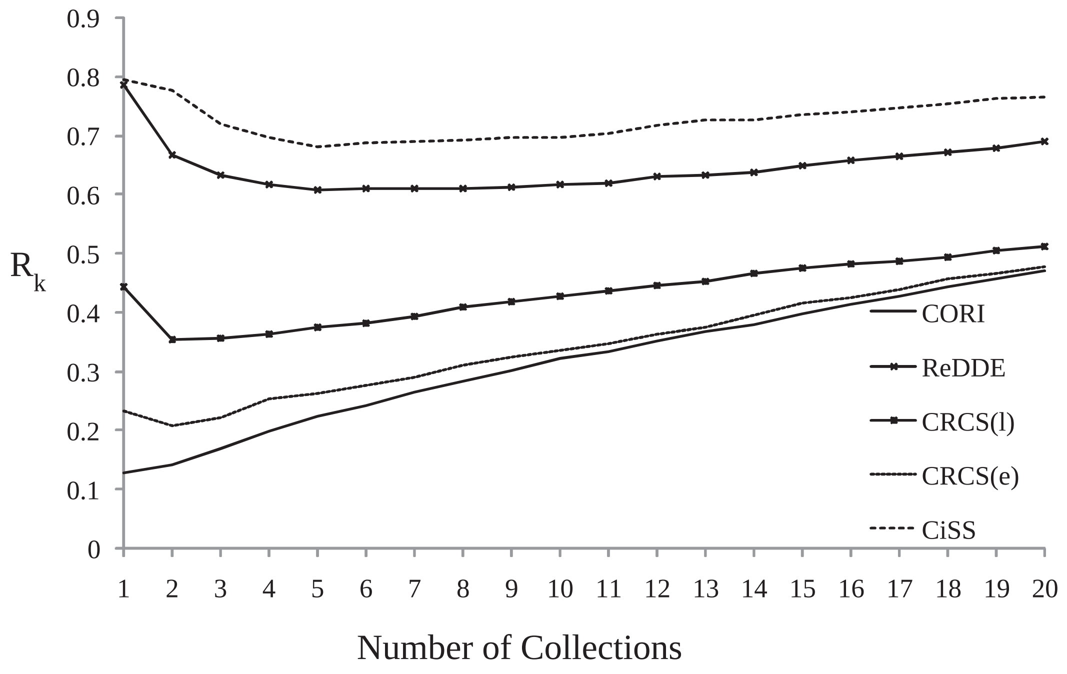


Figure 4.5 R_k values for the Representative testbed.

the testbed) and minimally skewed in terms of number of contained documents. As a result, the incorporated collection size estimator in CiSS provides a relatively similar preference for all collections, therefore no indication of the effectiveness of the utilization of collection sizes as it was introduced in the algorithm is provided at this testbed. The following testbeds serve exactly that purpose.

In the Representative and Relevant testbeds (figures 4.5 and 4.6), CiSS-HiRe locates the collections with the greatest number of relevant documents very effectively and thus attains a clear advantage. The results may be indicative that the new algorithm has a preference for large collections over smaller, but the results from the NonRelevant testbed (figure 4.7), where the two largest collections have almost no relevant documents and all algorithms perform similarly show that this is not the case.

The above results support the utilization of the estimated collection sizes as it was introduced into the algorithm, making it "sensitive" to collections with a significant number of relevant documents, regardless of their size. The Representative and Relevant are testbeds where ReDDE is typically very effective (Si and Callan 2003a, Si and Callan 2004). The experiments suggest that CiSS-HiRe presents an effective solution at collections with skewed

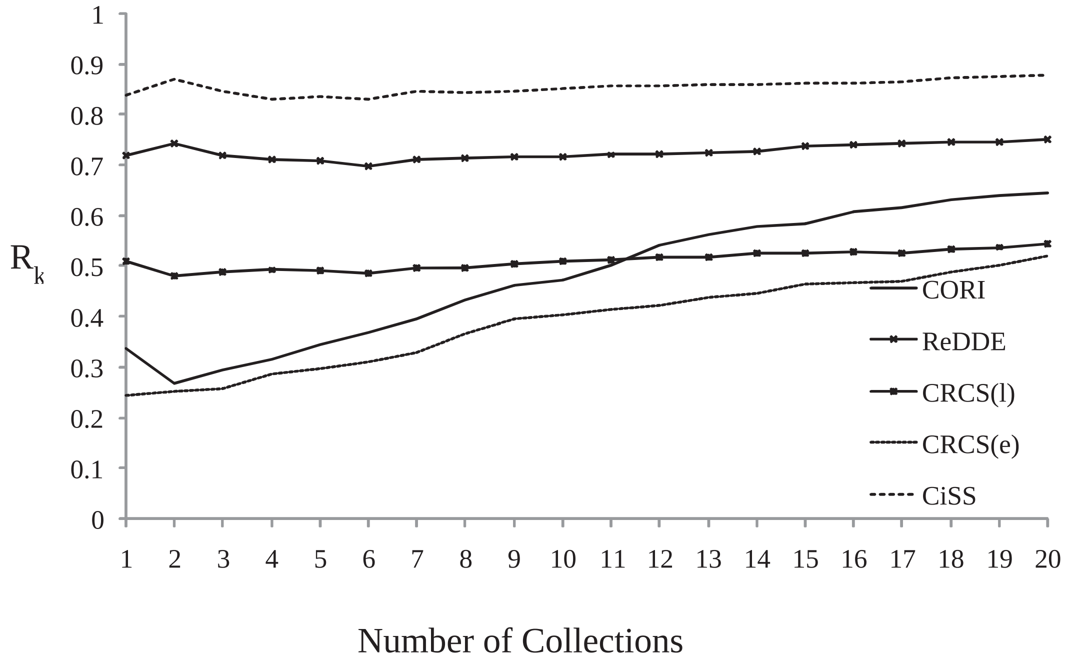


Figure 4.6 R_k values for the Relevant testbed.

collection sizes and concentration of relevant documents.

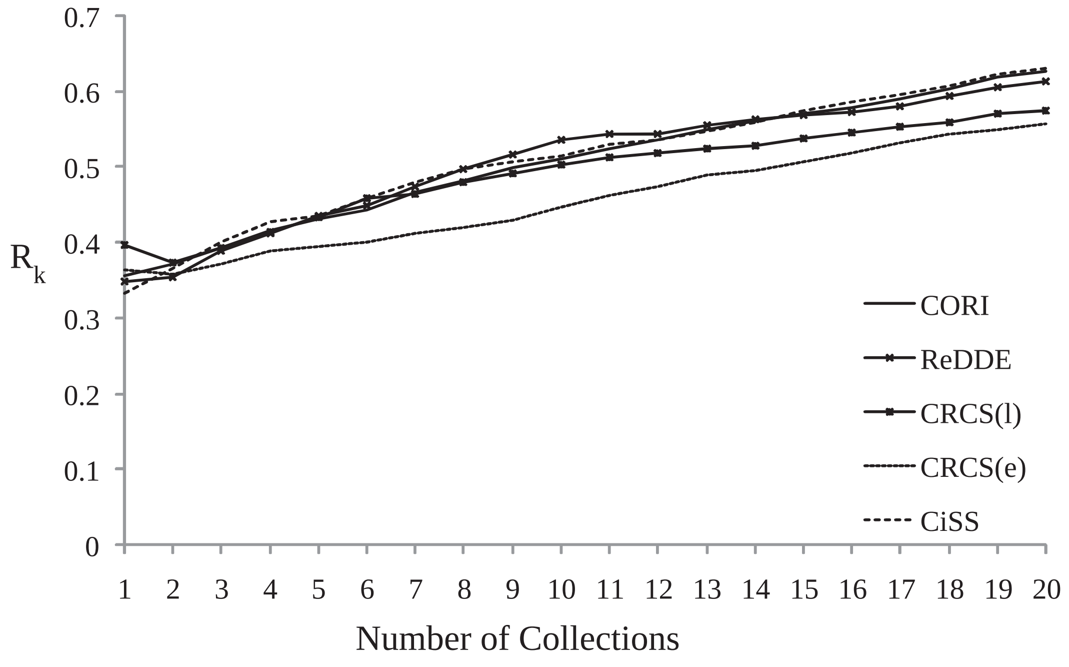


Figure 4.7 R_k values for the NonRelevant testbed.

In the K-means testbed (figure 4.8), all the algorithms perform similarly with some fluctuations although CORI and CRCS(l) do seem to be slightly more effective at settings of up to eight collections. Lastly, in the Web testbed (figure 4.9) CiSS-HiRe has a advantage for the first five collections, after which it is surpassed by CRCS(e), although the difference between the two algorithms remains small.

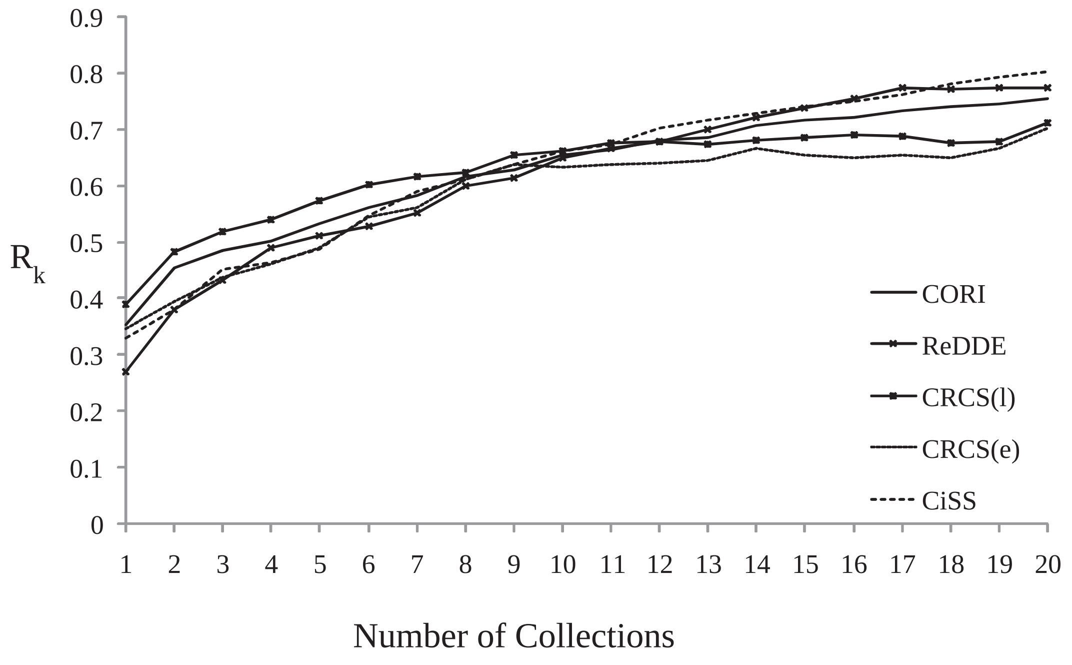


Figure 4.8 R_k values for the K-means testbed.

It can be observed that the wide diversity of the testbeds makes the goal of locating the collections with the greatest number of documents under every condition rather difficult. Nonetheless, the newly proposed algorithm has a more stable and effective performance across all testbeds, and is able to locate collections with the greatest number of documents equally or better than other state-of-the-art source selection algorithms. The above results provide an initial indication that the modeling of collections as integrals in the rank - relevance space does have some merit.

The results from precision-oriented environments are presented in Tables **4.1** to **4.6**. The ReDDE algorithm was used as baseline, because it has been reported to have a more stable performance across all testbeds (Si and Callan 2003a, Hawking and Thomas 2005, Shokouhi 2007). Measurements in bold and ‡ report statistically significant better performance

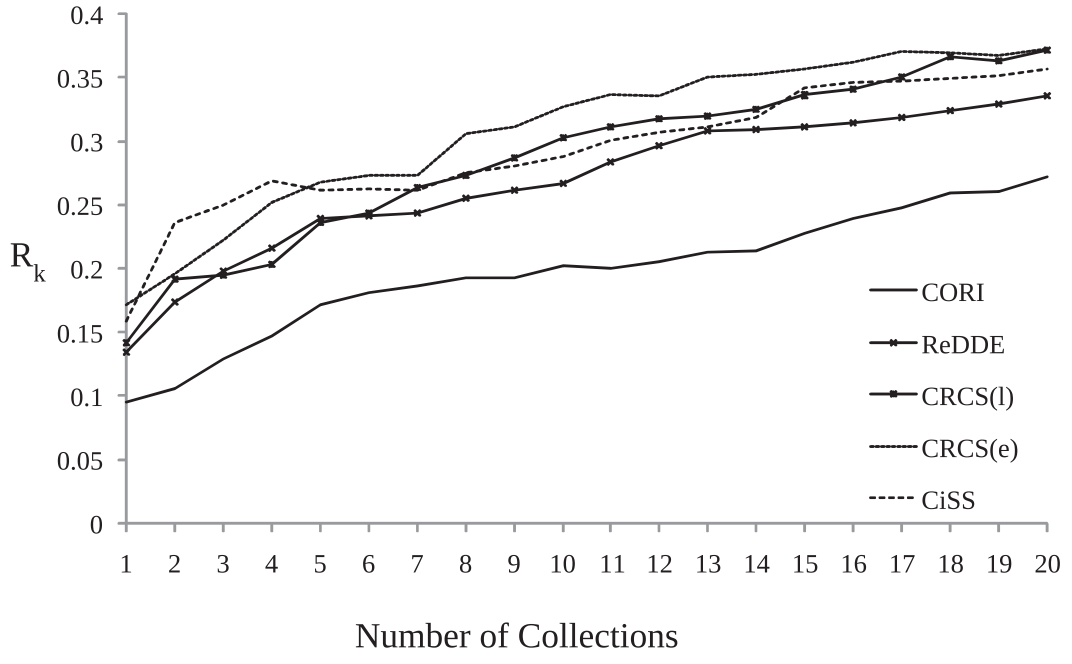


Figure 4.9 R_k values for the Web testbed.

(paired t-test $p < 0.1$ and $p < 0.05$ respectively) and measurements in *italic* statistically worse performance (paired t-test $p < 0.1$).

In the Uniform testbed (table **4.1**), both CiSS algorithms are more effective than other source selection algorithms. CiSS-HiPre also performs better than CiSS-HiRe and is able to maintain a high retrieval effectiveness even in lower ranks, despite the fact that it has 40% less documents available for merging. It is noteworthy that CiSS-HiPre is the only algorithm that attains a statistically significant better performance at this testbed than ReDDE, indicating the effectiveness of the segmentation of the collection integrals and the maximization the estimated gain of retrieving the top-ranked documents from remote collections, especially in comparison to CiSS-HiRe which considers the whole collection integral.

In the Representative and the Relevant testbeds (tables **4.2** and **4.3** respectively), both CiSS algorithms perform effectively indicating that they are both able to correctly identify the two collections containing most of the relevant documents. As previously, the results may have been due to a preference of the algorithms for bigger collections, but the results from the NonRelevant testbed (table **4.4**), where additionally CiSS-HiPre is able to produce

	P@5	**P@10**	**P@15**	**P@20**
CORI	0.4380	0.4000	0.3847	0.3675
ReDDE	0.4260	0.3860	0.3667	0.3555
CRCS(l)	0.4380	0.4000	0.3800	0.3625
CRCS(e)	0.4300	0.3940	0.3660	0.3530
CiSS-HiRe	0.4400	0.4010	0.3780	0.3670
CiSS-HiPre	0.4540	**0.4380‡**	**0.4093‡**	**0.3885‡**

Table **4.1** Precision at the Uniform testbed.

a statistically significant improvement over the baseline, show that this is not the case.

	P@5	**P@10**	**P@15**	**P@20**
CORI	*0.3900*	*0.3730*	*0.3493*	*0.3290*
ReDDE	0.4540	0.4350	0.4133	0.3955
CRCS(l)	0.4320	*0.3910*	*0.3813*	*0.3720*
CRCS(e)	0.4220	*0.3860*	*0.3627*	*0.3475*
CiSS-HiRe	0.4700	0.4510	0.4293	0.4115
CiSS-HiPre	0.4700	0.4450	0.4373	0.4175

Table **4.2** Precision at the Representative testbed.

CORI and the CRCS family of algorithms produce a statistically worse performance than ReDDE at the Representative and the Relevant testbeds, indicating the importance of an effective utilization of collection sizes (CORI makes no provisions for collection size) at testbeds with skewed collection sizes where most of the relevant documents are concentrated at a limited number of sources. At the NonRelevant testbed nonetheless, CORI is as effective as CiSS-HiPre.

CiSS-HiPre is also particularly effective in the K-means testbed (table **4.5**), especially in comparison to CiSS-HiRe. The CRCS family of algorithms are also very effective in this testbed, with CRCS(e) being particularly effective at P@5. It is noteworthy pointing out that CRCS(l) that attains better recall at this testbed (figure 4.8) doesn't perform as well in precision-oriented settings. The results show, as previously stated, that often a *high-recall* effectiveness doesn't necessarily translate into a *high-precision* performance. The results are in agreement with findings by Craswell (2000) and Si and Callan (2004).

	P@5	**P@10**	**P@15**	**P@20**
CORI	0.4160	*0.3690*	*0.3407*	*0.3180*
ReDDE	0.4720	0.4420	0.4253	0.3985
CRCS(l)	*0.4160*	*0.3840*	*0.3640*	*0.3415*
CRCS(e)	*0.3780*	*0.3500*	*0.3293*	*0.3065*
CiSS-HiRe	0.4760	0.4500	0.4360	0.4110
CiSS-HiPre	0.4720	0.4500	0.4353	0.4105

Table **4.3** Precision at the Relevant testbed.

	P@5	**P@10**	**P@15**	**P@20**
CORI	0.4320	0.4190	**0.4073‡**	**0.3925‡**
ReDDE	0.4220	0.3980	0.3813	0.3625
CRCS(l)	0.4400	0.4070	0.3933	0.3795
CRCS(e)	0.4320	0.4160	0.3973	0.3845
CiSS-HiRe	0.4420	0.3950	0.3793	0.3650
CiSS-HiPre	**0.4560**	**0.4270**	**0.4080**	**0.3900‡**

Table **4.4** Precision at the NonRelevant testbed.

The relatively low effectiveness of all the algorithms in the Web testbed (table **4.6**), is due to the fact that the distribution of relevant documents in this testbed is particularly skewed. The CRCS family of algorithms and CiSS-HiPre are slightly more effective in this testbed, although the differences observed are small and not statistically significant, due to the non-trivial number of queries that returned few or no relevant documents.

It can be concluded that the new algorithm and in particular CiSS-HiPre, provides a robust solution to the high precision problem in DIR environments being better or at least as effective as other state-of-the-art source selection algorithms, in the majority of settings.

4.6 Summary

In this chapter, a new source selection algorithm for uncooperative distributed information retrieval environments was presented, based on a novel modeling of the available remote collections as regions in a space created by the documents that they contain. In the

	P@5	**P@10**	**P@15**	**P@20**
CORI	0.4520	0.3840	0.3507	0.3160
ReDDE	0.4200	0.3620	0.3360	0.3090
CRCS(l)	0.4680	0.3980	0.3640	0.3380
CRCS(e)	**0.4960‡**	**0.4100**	0.3680	**0.3470**
CiSS-HiRe	0.4240	0.3800	0.3480	0.3230
CiSS-HiPre	0.4600	**0.4180‡**	**0.3733**	0.3360

Table **4.5** Precision at the K-means testbed.

	P@5	**P@10**	**P@15**	**P@20**
CORI	0.1663	0.1316	0.1074	0.0926
ReDDE	0.1789	0.1389	0.1123	0.0968
CRCS(l)	0.1958	0.1547	0.1228	0.1037
CRCS(e)	0.2021	0.1737	0.1481	0.1289
CiSS-HiRe	0.1916	0.1558	0.1242	0.1084
CiSS-HiPre	0.1958	0.1589	0.1319	0.1168

Table **4.6** Precision at the Web testbed.

experiments that were conducted it was demonstrated that the algorithm is equally or more effective than other state-of-the-art source selection algorithms in the majority of settings, both in recall and in precision oriented settings, overally constituting an effective solution to the source selection problem. Additionally, the algorithm requires no training phase, making it also very efficient in realistic environments.

One of the key novelties of the algorithm, in precision-oriented environments is that it only requires the number of documents in the final merged list be specified by the user, dynamically estimating the number of collections that are selected as well as the number of documents that are retrieved by each.

CHAPTER 5

Multiple Regression Rank Merging

5.1 Introduction

Having provided an effective and efficient solution to one of the most researched problems of distributed information retrieval, that of source selection, the investigation moves on to the last part of the process, *results merging*. It is notable that most successful algorithms, such as CORI (Callan, Lu, and Croft 1995) and SSL (Si and Callan 2003b) that were presented in chapter 2.4 make use of document relevance scores returned from the remote collections in order to be effective. However, in most modern information retrieval environments, information sources return only ranked lists of documents without scores, relying on the fact that the average user has no need for them, since they cannot be directly interpreted. In these environments, the utilization of algorithms that rely on relevance scores becomes problematic, forcing them to make assumptions that do not necessarily hold, such as assigning artificial relevance scores to the returned documents in a heuristic manner (Avrahami, Yau, Si, and Callan 2006). The new results merging algorithm presented in this chapter is designed explicitly to function effectively in such environments where the information provided by the remote collections is scarce.

In contrast to environments where relevance scores are available and the relevance of the returned documents in regard to the query can be extracted, much less information is actually conveyed in rankings. Even if a collection is marginally relevant to a query and the returned documents are only minimally relevant themselves, the rankings are the same as those provided by a very relevant collection returning very relevant documents. For example, a difference of one in ranking (i.e. 2 and 3) may be due to a minimal (i.e. 0.63 and 0.62) or more substantial (0.6 and 0.5) difference in score.

The algorithm presented, although influenced by the work of Si and Callan (2003b), which was presented in section 2.4.2, differs from their approach considerably in that it does not rely on document relevance scores returned from the remote collections, only ranked lists of documents. The algorithm assumes that the correlation between document ranks and relevance scores can be expressed through a logistic function and thus, using the sampled documents from the remote collections, maps collection-specific document rankings (the ranking that documents attain at the remote collection in respect to a particular query) to collection-specific document scores (the relevance score that documents would attain at the remote collection in respect to a particular query). Subsequently, using the centralized sample index and through linear regression, it assigns collection-independent relevance scores (the relevance scores that documents would attain if the query had been posed on a single global index) to the previously computed collection-specific document scores, thus producing a final merged document list. Because of the multiple regressions that it utilizes and the fact that it relies solely on ranks it was named *Multiple Regression Ranks-Merging* (*MRRM*).

5.2 Maximizing Usage of Local Resources

One of the motivations behind the MRRM algorithm is to function effectively in environments where the remote collections provide no cooperation at all. One could argue that the report of relevance scores is a form of cooperation on behalf of the remote collection, because they allow outside users to make deductions about the specifics of the retrieval algorithm employed. As already noted, in most modern information retrieval environments, relevance scores are not available, forcing existing results merging algorithms to assign artificial scores to the returned documents (Avrahami, Yau, Si, and Callan 2006), making assumptions that do not necessarily hold. Therefore, in order for any algorithm to be effective in such environments where no information is conveyed, it is important to maximize the usage of the already available local resources.

The sample documents that are acquired through query-based sampling are the main tool that is readily available since they are stored locally and are under the control of the DIR system. Their primarily use is for source selection and are usually discarded afterward. It was not until the work by Si and Callan (2003b) that they were also utilized in later stages of the DIR process. In that work, all the sampled documents from the individual collections were indexed into the *centralized sample index.*

The present work goes a step further and exploits this idea even more by regarding the locally-stored sampled documents of each remote as its representative. Under that notion, it is assumed that important statistics between the two document collections, sample and remote, share common patterns. For example, terms that have a high document frequency in the remote collection are expected to be similarly relatively frequent in the sample of the collection. In order to take advantage of the expected correlation, the documents that originated from each remote collection are indexed separately, creating a locally-stored *sample collection index* for each remote, which functions as its representative. The algorithm will utilize the local representatives of the remote collections in order to assign collection-dependent document scores to the ranked documents returned.

5.3 Expected Correlation between Ranking and Relevance Score

Previous work by Avrahami, Yau, Si, and Callan (2006) attempted to make up for the lack of relevance scores wherever that it was encountered by assuming that there is a linear mapping between ranks and relevance scores. Specifically, in cases where the remote collections did not return relevance scores, artificial scores were assigned to the returned documents, giving a score of 0.6 to the 1^{st} ranked document and decreasing at a steady rate until assigning a score of 0.4 to the last. Other mapping schemes are of course also possible but the work presented in that paper states that the above mapping produced the best results.

It has been shown nonetheless by Calvé and Savoy (2000) (see also section 2.4.2) that the decrease in relevance is not linearly connected to the ranking. Specifically, it was indicated that a logistic function provides a better mapping. According to that work, the probability that document D_i is relevant given its rank, is given by the formula:

$$Prob\,[D_i\ is\ Rel/x_i] = \frac{e^{a+bx_i}}{1+e^{a+bx_i}} \tag{5.1}$$

where x_i is the log of the rank of document D_i and a and b are the parameters of the mapping and actually depend on a variety of factors, such as the contents and the size of the collection etc. Figure 5.1 demonstrates the general expected correlation between probability of relevance and ranking using a logistic function with different parameters. Two are the important properties of the logistic function that make it attractive to this model, as noted by Nottelmann and Fuhr (2003b). First of all, the results are always in the [0,1] area, as a probability of relevance should be and secondly, it can produce a large variety of curves, by simply varying a and b.

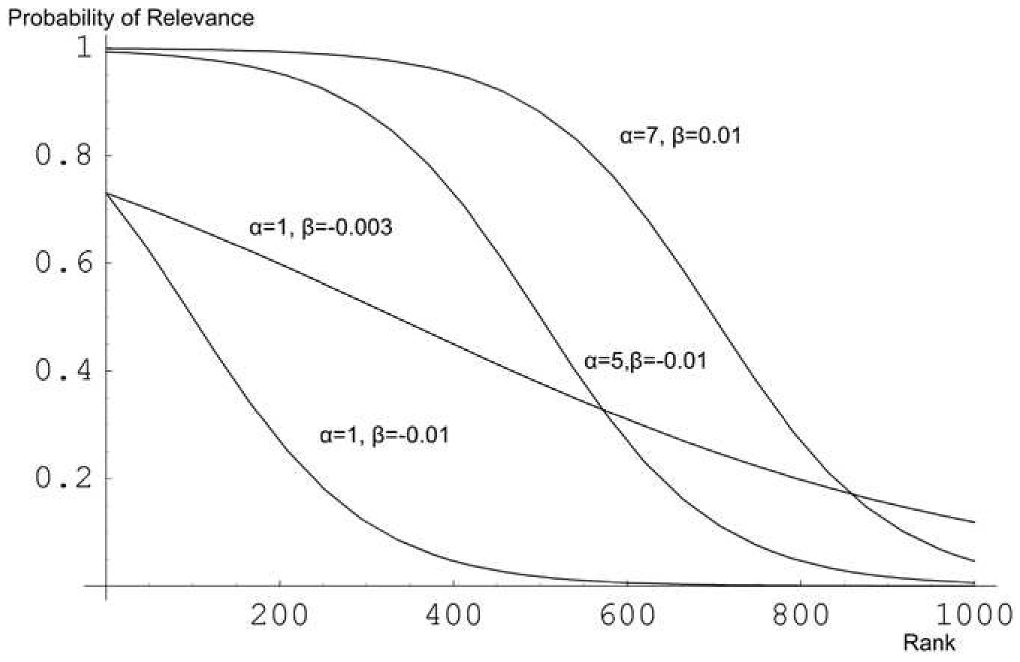


Figure 5.1: Plot demonstrating the expected correlation between rank (x) and probability of relevance (y) for various values of parameters a and b.

Based on the above, the present work moves away from heuristically assigning artificial scores linearly and estimates, during query time and without requiring any training, the actual graph for each collection on a per-query basis in order to produce accurate relevance scores.

5.4 Estimating Collection-dependent Relevance Scores from Rankings

After the source selection phase, where the most appropriate information sources have been selected, the query is delegated and executed at the selected remote collections and in parallel on the locally-stored sample collection indexes of the selected collections and the centralized sample index. We have used the INQUERY retrieval algorithm (Callan, Croft, and Harding 1992) to query both the local collections and the centralized index but any effective retrieval algorithm would do. Of course, in case CiSS has been used previously in the source selection phase, the originally document ranking produced by the centralized sample index may be retained and utilized again for the results merging stage increasing the efficiency of the retrieval process.

For each collection selected, two lists of documents are returned, one from the remote collection, containing only a ranked list of documents and one from the local sample, containing relevance scores. The result list from the centralized index is disregarded for the time being, as it is incorporated into the algorithm at a later stage. For each collection, the two document lists (sample - remote) are compared and all the *common* documents are stored along with the rank that they obtained at the remote collection and the relevance score at the local sample. These two elements will aid in the estimation of the relevance scores of the non-common documents.

5.4.1 Estimating the S-curve for each Collection

Given the common documents found between the remote and the sampled collections, the algorithm will attempt to estimate the S-curve for each collection, thus producing a model for assigning collection-dependent scores to the non-common documents returned from the remote collections. Influenced by the work of Calvé and Savoy (2000), we hypothesize that the correlation between the rank X of a document and the relevance score Y is given by a logistic function:

$$Y = \frac{e^{a+bX}}{1+e^{a+bX}} \tag{5.2}$$

Applying the following transformations, the above equation is modified into a liner one:

$$\frac{Y}{1-Y} = e^{a+bX} \tag{5.3}$$

$$ln\left(\frac{Y}{1-Y}\right) = a + bX \tag{5.4}$$

$$logit\,(Y) = a + bX \tag{5.5}$$

Parameters a and b of the above model must be estimated for each collection, in order to estimate the collection's corresponding S-curve. Since equation 6.5 is a linear one, that estimation can be accomplished through linear regression.

5.4.2 Linear Regression

Linear regression models are used when the relationship between two variables, dependent (y) and independent (x) or their transformation can be expressed with a linear function. The model can be formally stated as:

$$y = a + b * x + e \tag{5.6}$$

where a and b are the parameters of the model and e is the observed error (see below).

The aim of the model is to estimate the parameters a and b that minimize the error e which represents the difference between the observed values of y, denoted as y_i, and the ones estimated through the model, denoted as $\hat{y}_i$ where $i = 1, 2, \ldots, n$ and n is the number of observations, which in our case is the number of common documents between a remote collection and its local representative. The best way to accomplish this is through least-squares regression analysis. In particular, the algorithm aims at minimizing the sum of squared residuals S:

$$S = \sum_{i=1}^{n} [y_i - \hat{y}_i]^2 \tag{5.7}$$

The problem can be formalized using matrix terminology:

$$Y = X * B + e \tag{5.8}$$

where

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{bmatrix}, B = \begin{bmatrix} a \\ b \end{bmatrix}, e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

The optimal solution for parameters a and b is the one that minimizes S in equation (5.7) and is given by:

$$B = \left(X^{'}X\right)^{-1} X^{'}Y \tag{5.9}$$

The observations that are used for the above calculation are tuples (x_i, y_i) $i = 1, \ldots, n$ where x_i is the rank the i^{th} common document was assigned at the remote collection, y_i is the relevance score the i^{th} common document was assigned by the corresponding sample collection and n (as stated above) is the number of common documents found between the remote collection and its local representative.

Since we are particularly interested in the left peak of the S-curve, i.e. the top-ranking documents, we use only the first 10 common documents with a ranking lower than 300 and disregard the rest. Also, in order to better simulate the decline at the end of the graph (Fig. 5.1), we introduce an artificial common document at rank 3000 with score 0.001, effectively creating an extra tuple for regression at coordinates $(3000, 0.001)$. The choice of the exact coordinates of the artificial document were set empirically and initial experiments show that they don't affect the performance of the algorithm. Although these two adjustments are not necessary, they were found to increase the effectiveness of the algorithm as they result in a smoother decline at the right end of the graph.

The above process is repeated "on-the-fly" for each collection selected by the source selection algorithm, thus building a separate model for each. Note that, in contrast to Calvé and Savoy (2000), the estimated model is query-dependent, meaning that the same collection will produce different models for different queries, depending on the ranks and relevance scores of the common documents found. This property is desirable, since a collection may be very relevant for a specific query and less relevant for another and it allows the algorithm to assign high relevance scores to the returned documents in the former case and lower in the latter.

By applying equation 5.2 with the estimated parameters for each remote collection, the algorithm assigns collection-dependent scores to all the documents returned from the remote collections based on their ranking.

The estimation of relevance scores from rankings, using the above methodology is what differentiates the algorithm from previous approaches and makes it applicable in environments where only ranked lists of documents are returned. The previous approach of assigning scores linearly was based on the assumption that the relevance of documents declines linearly as the ranking increases, while the new approach is based on estimating the actual graph for each collection separately in regard to the particular query being posed.

One can expect that the best performance the above process can achieve is to assign the true relevance scores to the returned documents. The experiments that were conducted surprisingly showed that the above estimation method sometimes outperforms the true scores approach in a variety of settings, thus making the utilization of returned document scores sometimes unnecessary even when they are provided.

5.4.3 Estimating Collection-independent Relevance Scores

Having estimated collection-dependent scores for the documents returned from the remote collections, the algorithm moves on the second phase, which is to estimate collection-independent scores[1] for the returned documents and thus produce a final merged document list.

It is at this stage that the result list returned from the centralized sample index comes into use. The algorithm locates the common documents between the sample collections index and the centralized sample index and stores their sample and centralized scores.

An observation that should be made here is that a lot more common documents are expected to be found at this stage than in the first. The reason for the expected abundance of common documents is that the centralized sample index is comprised of the same subset of documents as the sample collections. Another factor is that since the particular collections were selected in the source selection phase, it is expected that they have at least some documents highly relevant to the particular query, which will appear high on the result list of the centralized index, with the condition of course that the source selection algorithm is at least somewhat effective. Experiments conducted, reported later in this thesis, support the initial expectations.

Based on the two scores, a separate linear regression model (equation 5.6) per collection is calculated "on-the-fly" that maps the sample scores of the common documents to the centralized scores. The model's purpose is to map collection-dependent scores to collection-independent scores, effectively assigning global scores to the documents returned from the

[1]Also referred to as "global scores", i.e. scores that the documents would obtain if the retrieval was done on a single centralized global index.

remote collections.

One might question whether linear regression is the best choice, or whether a polynomial non-linear regression would fit the data in a better way. Various reasons suggested the above decision. First of all, linear regression is consistent with the work by Si and Callan (2003b), where a linear model is used to fit the relevance scores attained from common documents at the remote collection and at the centralized index. Also, extensive early experiments showed that the benefit from going from a linear to a non-linear regression would be minimal, and that the data fits rather adequately with linear regression.

The tuples (x_i, y_i) $i = 1, \ldots, m$ that will be utilized in the estimation of the parameters a and b for each collection in this case are x_i, the score assigned to the i^{th} common document at the sample collection and y_i, the score assigned to the i^{th} common document at the centralized sample index where m is the number of common documents between the sample collection and the centralized sample index. Again, the preferred methodology for estimating the parameters is least-squares regression analysis (section 5.4.2).

Having estimated parameters a and b for each collection, the algorithm applies equation 5.6 to all the documents returned from the remote collections, using as independent variable the collection-dependent score x that was attributed to them during the first phase of the algorithm and producing a final collection-independent score y. The top 50 common documents were used for the second regression.

In summing up the algorithm, MRRM functions in three phases, one offline and two online. In the offline phase, the documents that were sampled from each collection are indexed separately, creating *local representatives* for the remote collections. Additionally, they are also indexed together, creating a *centralized sample index.*

In the first online phase of the algorithm, when a query is submitted to the DIR system and once the source selection stage is completed, the query is executed at the selected remote collections, their local representatives and the centralized sample index. The result lists from the remote collections and their local representatives are compared and all the common documents are stored, along with the rank that they attained from the remote collection and the relevance scores at the local representatives. Based on those documents, a logistic model is fitted into the data "on-the-fly" for each collection, that maps remote rankings to relevance scores. Applying the calculated mappings to the non-common documents, they are also assigned collection-dependent scores.

In the second online phase of the algorithm, the result lists from the local representatives of the selected collections and the centralized sample index are compared and all the common documents are stored, along with their respective scores. Based on those scores, a second, linear model is fitted for each collection "on-the-fly" that maps the sample scores, into global, collection-independent scores. By applying the estimated mappings to the non-common documents, they are assigned global scores. Finally, the documents are returned to the user in response to the submitted query in decreasing value of their final (collection-independent) score.

5.5 Experiment setup

The experimental results and the subsequent analysis are limited on the Uniform and the K-means testbeds. The skewness of the collection sizes are unrelated to the effectiveness of results merging algorithms, therefore no experiments were conducted on the Representative, Relevant and NonRelavant testbeds. Additionally, results merging algorithms have been traditionally tested in the above testbeds solely.

For queries, the title field from TREC topics $51-150$ were used for the Uniform collection and the description field from TREC topics $201 - 250$ for the K-means collection. More information is provided in Table **3.3**. For relevance judgments, the routing qrels for queries $51-100$ were used and the ad-hoc relevance judgments for queries $101-150$ for the Uniform testbed and the ad-hoc relevance judgements from TREC4 for the K-means testbed. Note that these are different from the ones used by Si and Callan (2003b), where they combined the routing and the adhoc relevance judgments for all queries. This difference, although it will produce somewhat different results, does not alter the conclusions drawn by the experiments, since the same conditions are applied to all the algorithms tested.

Two sets of experiments were conducted to examine the performance of the algorithm. In the first, the remote collections returned only ranked lists of documents, without relevance scores. Since both CORI and SSL need relevance scores in order to function, artificial scores had to be manufactured. Avrahami, Yau, Si, and Callan (2006) suggest that a good range is $0.6 - 0.4$. Indeed, after testing various ranges, it was found that although the differences weren't significant the $0.6 - 0.4$ range produced slightly better results. In the second set, the remote collections returned relevance scores and both CORI and SSL were able to use them normally. MRRM operated with no relevance scores in both settings.

Although CiSS could be utilized in the source selection phase, CORI was instead chosen in order to focus the study to the results merging phase. Additionally, CORI is considered as a standard algorithm when evaluating new approaches for source representation and results merging (Azzopardi, Baillie, and Crestani 2006, Si and Callan 2002), therefore its choice makes the experimental results presented here more compatible with other research.

5.6 Results

5.6.1 Number of Common Documents

In order for MRRM to function properly a minimum number of common documents need to be found between: a) the remote and their responding sample collections at the first phase of the algorithm and b) the sample collections and the centralized sample index at the second phase. Both regressions used in the model are effectively linear, which means that the need for common documents is not excessive (2 documents suffice), but the more common documents found, the better the estimation.

The number of common documents found in both phases in two settings are reported. In the first setting, remote collections are requested to return 1,000 documents and in the second 300 documents. Taking into consideration that the average collection size for the two testbeds is approximately 10,000 and 5,000 documents for Uniform and K-means respectively (table **3.2**) and that each collection sample has only 300 documents, which means that on average only 3% and 6% of the remote collections has been sampled for the two testbeds respectively, it becomes evident that the chances of the discovery of a common document are seemingly very small. Nonetheless, the non-uniformity of the sampling as observed by Shokouhi, Zobel, Scholer, and Tahaghoghi (2006), works in favor of MRRM, and SSL for that matter, because it predicts that the more verbose and authoritative documents are the ones that will be sampled through the query-based sampling process, thus providing enough documents for regression.

A similar problem has been examined before by Si and Callan (2003b), but the context of the comparison at that work was different, taking place only between the remote collections and the centralized sample index. On the contrary, in the work that is presented here, two different comparisons need to take place and both provide sufficient common documents for the final computed modeling to be effective. The experiments indicate that at the majority of cases at both settings, there are enough documents found to perform both regressions effectively.

Remote and Sample Collections

Figure 5.2 shows the number of common documents that are found between the sample and the remote collections when 10 collections are selected to return 1000 documents each. The instances (y axis) refer to the number of times that the number of common documents found between a sample and a remote collection is as the corresponding interval at the x axis. The black columns refer to the Uniform collection (queries $51 - 150$) and the white ones to the K-means (queries $201 - 250$). For example, in the Uniform collection there are approximately 100 instances out of 1000 (100 queries*10 collections) where $0 - 10$ common documents are found (first black column from the left on Figure 5.2). Respectively, there are 13 occasions out of 500 (50 queries*10 collections) that $0 - 10$ common documents are found in the K-means collection.

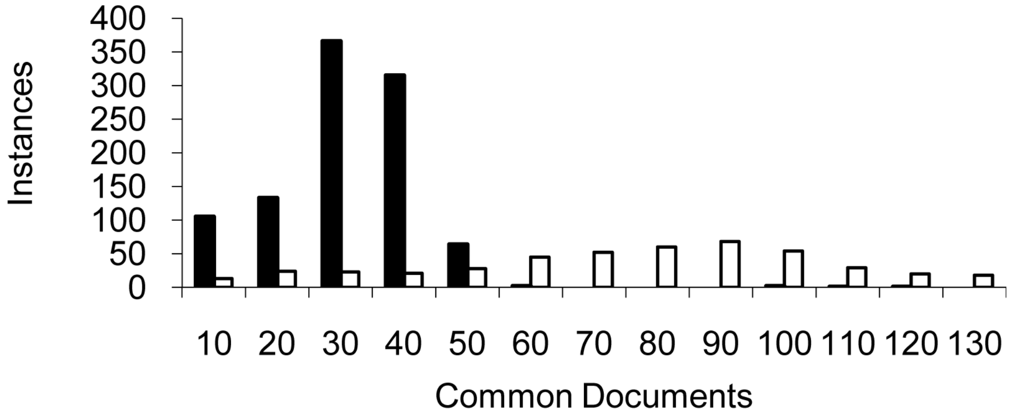


Figure 5.2: Common documents found between the sample and the remote collections, when 1000 documents are returned. The black histogram refers to the Uniform testbed and the white to the K-means.

For the majority of queries, the number of common documents found is between $20 - 30$ for Uniform and $80 - 90$ for K-means, indicating that most of the times there are enough documents discovered to perform the first phase regression effectively. Taking into consideration that the algorithm uses only the top 10 documents and disregards the rest (section 5.4.2), it can be concluded that the estimated model is 89.5% [2] for the Uniform and 97.4%[3] for the K-means of all cases "optimal". The results may be surprising but, as explained above, the non-uniformity of the sampling process is mostly responsible for the observed outcome. Paradoxically, a uniform sampling, which is still an open research issue under some environments (Thomas and Hawking 2007) would most likely result in a decrease of the number of common documents and potentially into a decrease of effectivess, both for MRRM and SSL. Of course, the benefits from a uniform sampling are most obvious in the estimation of the sizes of the remote collections rather than in the context that the sampling is used in this work.

There are a few cases where less than the minimum number of needed documents is found. Specifically, there are 33 occasions in which less than 3 documents are located for queries $51-150$ and only 1 occasion for queries $201-250$. For these instances, a "fall back" strategy is adopted and a default logistic function with values of $a = -0.1$ and $b = -0.05$ is used.

When 300 documents are requested from the remote collections (Figure 5.3) the number of common documents is, as expected, decreased. The histogram shows that in this setting, $0 - 10$ common documents are found in the majority of cases (541 instances out of 1000) for queries $51 - 150$ and $20 - 30$ common documents are found at 165 occasions for queries $201 - 250$. The number of cases where more than 10 common documents are found and the modeling is 'optimal" falls at 46.2%[4] for the Uniform testbed and 89.6%[5] for the K-means testbed. The number of occasions where less than 3 documents are located and a "fall-back" strategy has to be adopted increased to 52 for the Uniform collection, which although higher than in the previous setting, is still only a small fraction (5.2%). The fall back strategy had to be adopted only 13 times for the K-means collection.

The above results show that in the majority of cases (appr. 95% in both settings) the algorithm is applicable without any hindrances. The number of common documents found, although affected by the number of requested documents from the remote collections, is

[2]895 occasions out of 1000
[3]487 out of 500 occasions
[4]462 out of 1000 occassions
[5]448 out of 500 occassions

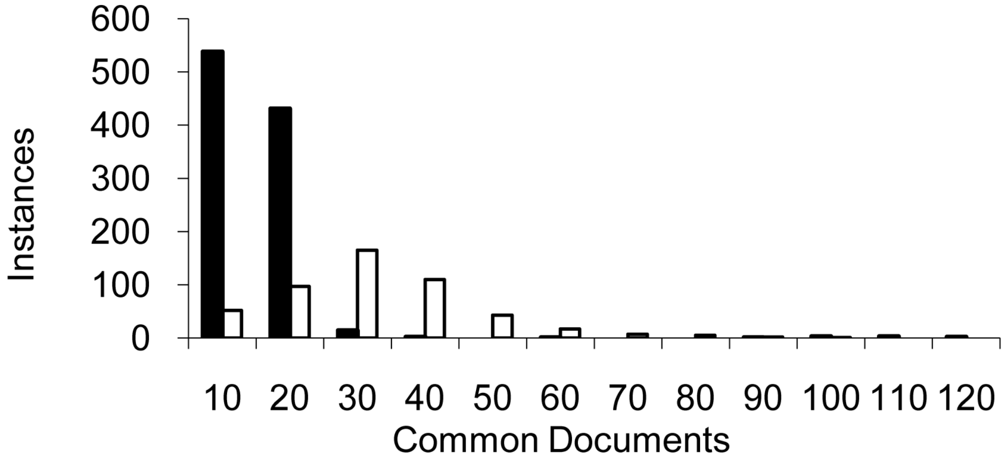


Figure 5.3: Common documents found between the sample and the remote collections, when 300 documents are returned. The black histogram refers to the Uniform testbed and the white to the K-means.

almost always sufficient for the operation of the algorithm and in many cases the modeling produced is optimal in reference to the limitation of the top 10 documents that are utilized.

In a realistic web environment, the algorithm could download on-the-fly a small number of documents, index them and calculate relevance scores for them using the statistics from the appropriate sample collection, in order to avoid adopting the fall back strategy. The *Hybrid* results merging algorithm that is presented in chapter 6 is exactly based on that requirement, although the approach adopted there differs significantly from the one that has been applied for MRRM. Alternatively, a more extensive sampling would also result in a increase of common documents between the sample and the remote collections.

Sample Collections and Centralized Sample Index

A second question raised is whether enough common documents are found between the sample collections and the centralized sample index in the second phase of the algorithm. As explained above, a significant number of common documents is expected to be discovered in this case, because of the fact that the centralized sample index is made of the same documents that comprise the sample collections. Since both the sample collections and the centralized index are kept locally and are under the control of the DIR system, the maximum number of documents is always requested from both sources. Results are shown on figure 5.4.

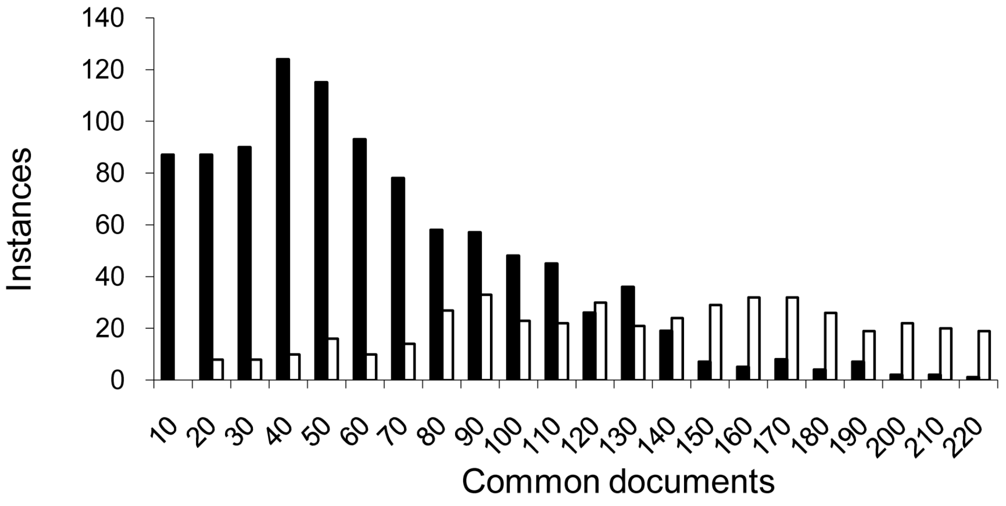


Figure 5.4: Common documents found between the samples and the centralized index. The black histogram refers to the Uniform testbed and the white to the K-means.

40 − 50 common documents are found in the Uniform testbed and 160 − 180 in the K-means testbed in the majority of cases, which more than suffice for MRRM to perform the second regression. As stated at section 5.4.3 only the first 50 common documents are utilized in the second phase regression, meaning that the estimated parameters for the model are "optimal" in 49.6% in the Uniform and 91.6% in the K-means testbeds of all cases. Although there is still a small number of instances (35 out of 1000 for the Uniform collection but none for the K-means collection) where there are less than 3 common documents found, more than enough documents are usually found to perform the second stage regression without any difficulties.

Overall, it can be observed that the number of common documents is most of the times more than adequate for the algorithm to function effectively and only rarely is a fall back strategy adopted. The above illustrate the feasibility of the algorithm in both clustered and uniform environments. Particularly, in clustered environments, the algorithm is most of the times able to locate sufficient documents to optimally estimate the parameters for the produced models. It remains to be seen, in the next section, if that abundance of common documents in the K-means testbed translates into an increased effectiveness.

5.6.2 Precision

In distributed information retrieval environments it is usually inefficient to retrieve all the relevant documents scattered in the remote collections. Especially, when the focus of the retrieval is on the results merging part of the process, the focus is on precision. Tables **5.1** to **5.16** report the results on precision at rankings 5 to 30 in a variety of settings.

In addition to the settings described above (i.e. with and without relevance scores), the algorithm was tested with different parameters for each setting. In the first, a significant number of collections, namely ten, was selected to contribute documents, while in the second that number was limited to three. Previous research has shown that selecting more than a limited amount of collections (i.e. $3 - 5$) results in an increase of "noise" being introduced in the final merged list, in the form of non-relevant documents, due to the increased number of collections contributing documents, some of which may return a small number of relevant documents but a large number of irrelevant documents. It would be interesting to examine how MRRM and the other results merging algorithm are able to cope with the increase of noise at this setting. On the contrary, the second setting offers a much more focused environment, with the utilization of only a small number of collections, namely three. These two diverse environments will allow for a more in depth overview of the proposed algorithm.

At each of these settings, two lines of experiments were conducted. In the first, the remote collections were requested to return a maximum of 1000 documents. At this setting, MRRM, as well as SSL that uses regression, are expected to perform optimally, because of the adequate number of returned documents that allow them to estimate more accurate models, as already noted in section 5.6.1. In the second setting, the number of returned documents was limited to 300. Although, as examined in section 5.6.1, MRRM is usually able to locate the minimum number of documents needed for regression even at this setting, the experiments conducted aim to offer a view of the algorithms performance in environments where the estimated model isn't the optimal.

The diversity of settings and the extensibility of results presented in the following section will provide both a deeper understanding into the performance of MRRM in realistic environments as well as a greater insight on a number of issues concerning other state-of-the-art results merging algorithms in distributed information retrieval environments.

Precision when remote collections return only ranked lists of documents

Results at settings where remote collections do not report document relevance scores are provided in tables **5.1** to **5.8**. The percentages in parentheses report the difference between MRRM and CORI (left parenthesis) and SSL (right parenthesis) respectively.

Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.1460	0.1560	0.1780 (+21.9%) (+14.1%)
At 10 documents:	0.1370	0.1340	0.1520 (+10.9%) (+13.4%)
At 15 documents:	0.1260	0.1267	0.1353 (+7.4%) (+6.8%)
At 20 documents:	0.1180	0.1205	0.1295 (+9.7%) (+7.5%)
At 30 documents:	0.1113	0.1103	0.1160 (+4.2%) (+5.1%)

Table **5.1**: Precision with 10 collections selected at the Uniform testbed, each one returning 1000 documents with No Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

At the first setting in the Uniform testbed (table **5.1**), where 10 collections are selected, each one returning 1000 documents, MRRM is able to outperform both CORI and SSL, often by a significant margin. The differences in precision are more substantial at precision at 5 and 10 documents and slowly decrease in later precision measurements, but still remain nontrivial. The results demonstrate that the modeling that MRRM provides are quite effective, allowing it to gain a clear advantage of the approach of attributing scores to ranks heuristically in a linear manner. The observed decrease in differences in precision from 15 to 30 documents may be attributed to the already discussed introduction of non-relevant documents in the merged lists. The results show that even though MRRM is also affected by the phenomenon, it manages to remain highly effective.

The results are almost similar in the K-means testbed (table **5.2**), although the difference observed between MRRM and CORI is particularly substantial, varying around 50%. The difference with SSL is less, although still above 10% in most settings, proving again that the linear modeling of scores isn't very effective. The above may also be an indication that algorithms that make use of some sort of regression may have an advantage in clustered environments.

Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.2640	0.3520	0.4000 (+51.5%) (+13.6%)
At 10 documents:	0.2220	0.3180	0.3520 (+58.6%) (+10.7%)
At 15 documents:	0.2133	0.2867	0.3187 (+49.4%) (+11.2%)
At 20 documents:	0.2010	0.2740	0.2930 (+45.8%) (+6.9%)
At 30 documents:	0.1793	0.2580	0.2560 (+43.2%) (-0.5%)

Table **5.2**: Precision with 10 collections selected at the K-means testbed, each one returning 1000 documents with No Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.1340	0.1600	0.1820 (+35.8%) (+13.8%)
At 10 documents:	0.1340	0.1460	0.1560 (+16.4%) (+6.8%)
At 15 documents:	0.1333	0.1373	0.1393 (+4.5%) (+1.4%)
At 20 documents:	0.1300	0.1290	0.1325 (+1.9%) (+2.7%)
At 30 documents:	0.1260	0.1177	0.1190 (-5.6%) (+1.1%)

Table **5.3**: Precision with 10 collections selected at the Uniform testbed, each one returning 300 documents with No Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

Tables **5.3** and **5.4** report the precision at settings where 10 collections are selected to return 300 documents each. At the Uniform testbed the behavior of the algorithms is similar to that observed in the previous setting (table **5.1**), with MRRM outperforming CORI by a substantial margin and SSL more moderately. What is surprising is that both algorithms that use regression perform similarly or slightly better in comparison to environments where collections return 1000 documents (table **5.1**), indicating that the reduction of documents does not significantly effect the accuracy of the estimated models and instead the decrease of the introduced noise is affecting the final precision in a more substantial manner. Finding the "golden section" between those two opposite factors may provide significant advances in the future.

Precision	CORI	SSL	MRRM
At 5 documents:	0.2800	0.3400	0.3960 (+41.4%) (+16.5%)
At 10 documents:	0.2340	0.3140	0.3520 (+50.4%) (+12.1%)
At 15 documents:	0.2160	0.2840	0.3173 (+46.9%) (+11.7%)
At 20 documents:	0.2050	0.2740	0.2900 (+41.5%) (+5.8%)
At 30 documents:	0.1833	0.2540	0.2540 (+38.5%) (±0%)

Table **5.4**: Precision with 10 collections selected at the K-means testbed, each one returning 300 documents with No Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

The results are also similar for the K-means testbed (table **5.4**). MRRM significantly outperforms CORI (reaching differences of up to 50%) and performs again significantly but more moderately better than SSL, in most precision measurements. Interestingly, the algorithms do not exhibit the same behavior as the Uniform testbed, where they attained better performance in comparison to the environment where collections return 1000 documents (table **5.2**), potentially because of the peak of performance reached at that setting and because of the fact that the collections at K-means are clustered, i.e. much more focused on specific topics, thus the introduced noise is minimal.

Overall, in environments where a significant number of collections are requested to return documents, the modeling that MRRM provides is able to steadily outperform the simple linear assignment of scores, resulting in an increased precision in both clustered and non-clustered testbeds, regardless of the number of returned documents. Particularly, in clustered environments the combination of logistic modeling and regression seems to be very effective.

Results for experiments where 3 collections are requested to return documents are reported at tables **5.5** to **5.8**. Not so surprisingly, the drop of effectiveness in comparison to settings where 10 collections are selected to contributed documents, isn't as profound as one would expect, verifying the initial statement that selecting an increased number of collections does not necessarily translate into an increase in precision.

At the first setting (table **5.5**) collections return 1000 documents each. The proposed algorithm is again able to outperform both CORI and SSL but the interrelations amongst

Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.1460	0.1480	0.1760 (+20.5%) (+18.9%)
At 15 documents:	0.1410	0.1300	0.1450 (+2.8%) (+11.5%)
At 20 documents:	0.1287	0.1193	0.1333 (+3.6%) (+11.7%)
At 25 documents:	0.1185	0.1070	0.1215 (+2.5%) (+13.6%)
At 30 documents:	0.1120	0.0933	0.1077 (-3.8%) (+15.4%)

Table **5.5**: Precision with 3 collections selected at the Uniform testbed, each one returning 1000 documents with No Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

the algorithms are different. MRRM steadily outperforms SSL at every precision measurement, with differences always above 10%. CORI proves to be very effective at settings like this when only a limited number of collections are contributed documents, recording a very steady performance, almost at par with MRRM, with the exception of precision at 5 documents, where it is surpassed by a significant margin. MRRM's performance at 5 documents is particularly good, inticating that the calculated modeling is very well fitted to the top documents.

Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.2640	0.3120	0.3480 (+31.8%) (+11.5%)
At 10 documents:	0.2220	0.2760	0.3100 (+39.6%) (+12.3%)
At 15 documents:	0.2120	0.2493	0.2827 (+33.3%) (+13.4%)
At 20 documents:	0.1990	0.2320	0.2600 (+30.7%) (+12.1%)
At 30 documents:	0.1753	0.2160	0.2340 (+33.5%) (+8.3%)

Table **5.6**: Precision with 3 collections selected at the K-means testbed, each one returning 1000 documents with No Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

At the K-means testbed (table **5.6**), the drop of precision in comparison to environments where 10 collections contribute documents (**5.2**) is more substantial, but still not as much as someone would expect taking into consideration that there is a reduction of 70% of contributing collections (3 versus 10). Both MRRM and SSL report drops in effectivess, while CORI actually performs better at this setting, affirming the hypothesis that it reaches its best performance in focused environments with a limited number of collections. MRRM, is again able to outperform both CORI and SSL, with significant differences (ranging around 30% in comparison to the former and 10% with the latter).

Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.1360	0.1580	0.1760 (+29.4%) (+11.4%)
At 10 documents:	0.1360	0.1300	0.1450 (+6.6%) (+11.5%)
At 15 documents:	0.1293	0.1187	0.1333 (+3.1%) (+12.3%)
At 20 documents:	0.1205	0.1075	0.1215 (+0.8%) (+13.0%)
At 30 documents:	0.1153	0.0947	0.1077 (-6.6%) (+13.7%)

Table **5.7**: Precision with 3 collections selected at the Uniform testbed, each one returning 300 documents with No Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

Precision when 3 collections return 300 documents are reported at tables **5.7** and **5.8**. At the Uniform testbed, both MRRM and SSL perform very well at precision at 5 documents, noting substantial differences from CORI but drop significantly in later measurements. In fact, SSL is outperformed by CORI at measurements from 10 documents and on, while MRRM is able to outperform it until 30 documents. The results may be indicative that regression algorithms are particularly optimized for early precision, producing models that fit very well to the first 5-10 documents but are not so precise for the ensuing documents. The above results may also be attributed to the fact that the actual document scores do not follow any sort of specific curve[6], thus estimating one through regression necessarily introduces approximations, which, as already stated, are deliberately best fitted to the top ranking documents.

Results at the K-means testbed (table **5.8**) are similar to those when 3 collections return 1000 documents (table **5.6**), although all algorithms note a minor drop in precision, but not in a significant manner. MRRM is again significantly better than CORI and more

[6]As already noted, a variety of factors are responsible for the actual score distribution, such as the difficulty of the query, the size of the collections, the algorithms employed etc.

Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.2800	0.3120	0.3520 (+25.7%) (+12.8%)
At 10 documents:	0.2360	0.2800	0.3120 (+32.3%) (+11.4%)
At 15 documents:	0.2133	0.2587	0.2813 (+31.9%) (+8.7%)
At 20 documents:	0.2030	0.2430	0.2570 (+26.6%) (+5.8%)
At 30 documents:	0.1807	0.2233	0.2300 (+27.3%) (+3.0%)

Table **5.8**: Precision with 3 collections selected at the K-means testbed, each one returning 300 documents with No Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

moderately than SSL.

Overall, the results clearly demonstrate the effectiveness of MRRM in environments where remote collections return only ranked lists of documents. We examined the performance of the algorithm in environments where a significant number of collections contribute documents to the final merged list and again when the number of collections is limited. Additionally, for each of the above settings, we tested its performance when the remote collections return an increased number of documents and again when they return fewer documents. In all of the above settings, both in uniform and in clustered testbeds, the performance of the proposed algorithm was found to be better or at least as good as that of previous state-of-the-art approaches, constituting a very effective and robust approach in score-lacking environments.

Precision when remote collections report relevance scores

In settings where relevance scores are available (Tables **5.9** to **5.16**), the comparison produces mixed results. It should be noted that the above setting clearly creates an "unfair" environment for MRRM, since the motivation behind the algorithm is explicitly to function in settings where relevance scores are not available.

The differences between the algorithms are not as significant as one might expect, and in some cases MRRM performs better. Specifically, in the Uniform testbed when 10 collections return 1000 documents each (table **5.9**), MRRM is outperformed by CORI only at precision at 5 documents and performs similarly to it at later measurements. SSL, being more effective

Uniform			
Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.1960	0.2060	0.1780 (-9.2%) (-13.6%)
At 15 documents:	0.1540	0.1740	0.1520 (-1.3%) (-12.6%)
At 20 documents:	0.1360	0.1593	0.1353 (-0.5%) (-15.1%)
At 25 documents:	0.1305	0.1575	0.1295 (-0.8%) (-17.8%)
At 30 documents:	0.1170	0.1490	0.1160 (-0.9%) (-22.1%)

Table **5.9**: Precision with 10 collections selected at the Trec123 testbed, each one returning 1000 documents with Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

than CORI, is able to outperform both approaches by a significant margin.

Uniform			
Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.1960	0.1960	0.1820 (-7.1%) (-7.1%)
At 10 documents:	0.1540	0.1760	0.1560 (-1.3%) (-11.4%)
At 15 documents:	0.1360	0.1527	0.1393 (-2.4%) (-8.8%)
At 20 documents:	0.1305	0.1530	0.1325 (-1.5%) (-13.4%)
At 30 documents:	0.1170	0.1443	0.1190 (-1.7%) (-17.5%)

Table **5.10**: Precision with 10 collections selected at the Trec123 testbed, each one returning 300 documents with Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

The results are mostly similar when 10 collections return 300 documents (table **5.10**) at the Uniform testbed, with the performance of CORI and MRRM being very close, although CORI is usually slightly better and SSL again outperforming both with a substantial margin.

The same doesn't apply at the last setting at the Uniform testbed where 3 collections are selected (tables **5.11** and **5.12**), where surprisingly, MRRM is able to outperform CORI by a substantial margin, ranging above 16% in most cases and SSL at precision at 5 documents (14.3% and 20.5% respectively when collections return 1000 and 300 documents). The

Uniform			
Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.1480	0.1540	0.1760 (+18.9%) (+14.3%)
At 10 documents:	0.1240	0.1440	0.1450 (+16.9%) (+0.7%)
At 15 documents:	0.1147	0.1480	0.1333 (+16.2%) (-9.9%)
At 20 documents:	0.1025	0.1365	0.1215 (+18.5%) (-11.0%)
At 30 documents:	0.0863	0.1193	0.1077 (+24.8%) (-9.7%)

Table **5.11**: Precision with 3 collections selected at the Trec123 testbed, each one returning 1000 documents with Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

results indicate that in environments where a limited number of collections is selected to contribute documents, the methodology described for estimating collection-dependent scores sometimes performs better than algorithms that make use of documents relevance scores as they are reported from the remote collections.

One noticeable fact is that CORI in this setting, where collections report relevance scores, performs better when 10 collections contribute documents rather than 3. The observation comes in contradiction to the previous setting, where only ranked lists of documents are reported, where CORI noted an increase in performance at the 3 collections settings. The result may be due to the significance of information that is actually conveyed in document scores. In settings where collections return only rankings, the CORI merging algorithm receives as input ten similar document lists having scores from 0.6 to 0.4 and only relies on the collection score in order to estimate the final merged list (equation 2.44). Thus in settings where 10 collections are selected it is much more easy for noise to enter the final list, in comparison to settings where the number of contributing collections is limited to only 3. At environments where relevance scores are reported, the algorithm fully utilizes the score differences between the documents returned from the same collection (equation 2.42) thus it is able to better distinguish between more and less relevant documents, achieving better precision.

The results are very interesting in the K-means testbed (tables **5.13** to **5.16**). Both CORI and SSL mark an increase in precision because of the availability of relevance scores

Uniform			
Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.1480	0.1460	0.1760 (+18.9%) (+20.5%)
At 10 documents:	0.1240	0.1340	0.1450 (+16.9%) (+8.2%)
At 15 documents:	0.1147	0.1367	0.1333 (+16.2%) (-2.5%)
At 20 documents:	0.1025	0.1280	0.1215 (+18.5%) (-5.1%)
At 30 documents:	0.0863	0.1133	0.1077 (+24.8%) (-4.9%)

Table **5.12**: Precision with 3 collections selected at the Trec123 testbed, each one returning 300 documents with Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

from the remote collections, but are still not able in most cases to reach the performance of MRRM, which doesn't make use of reported relevance scores.

K-means			
Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.3120	0.3720	0.4000 (+28.2%) (+7.5%)
At 10 documents:	0.2600	0.3280	0.3520 (+35.4%) (+7.3%)
At 15 documents:	0.2320	0.3160	0.3187 (+37.4%) (+0.9%)
At 20 documents:	0.2230	0.2920	0.2930 (+31.4%) (+0.3%)
At 30 documents:	0.1947	0.2660	0.2560 (+31.8%) (-3.5%)

Table **5.13**: Precision with 10 collections selected at the Trec4 testbed, each one returning 1000 documents with Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

In the first experiment, where 10 collections return 1000 documents each (table **5.13**, the performance gap between CORI and MRRM is around 30%, which although smaller than the one reported in settings where relevance scores aren't available (table **5.2**) is still substantial. Likewise, SSL performs, as expected, better at this environment but still lags in precision at 5 and 10 documents and matches the performance of MRRM only at later measurements.

K-means			
Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.3167	0.3520	0.3960 (+25.0%) (+12.5%)
At 10 documents:	0.2646	0.3280	0.3520 (+33.0%) (+7.3%)
At 15 documents:	0.2347	0.3067	0.3173 (+35.2%) (+3.5%)
At 20 documents:	0.2260	0.2850	0.2900 (+28.3%) (+1.8%)
At 30 documents:	0.1965	0.2607	0.2540 (+29.3%) (-2.6%)

Table **5.14**: Precision with 10 collections selected at the Trec4 testbed, each one returning 300 documents with Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

The results are identical when 10 collections are requested to return 300 documents each (table **5.14**), indicating again that the decrease of available documents for regression doesn't influence the effectiveness of the final computed model and the algorithm manages to outperform both CORI and SSL which make use of relevance scores.

At the last setting, where 3 collections are requested to return 1000 and 300 documents at the K-means testbed (tables **5.15** and **5.16** respectively) the produced results are somewhat mixed. MRRM is again able to outperform CORI, as in previous settings, but the performance of SSL is at par with MRRM and usually better. The results are somewhat surprising, given the robustness so far of the new algorithm in clustered environments and settings with a limited number of contributing collections. Taking into consideration that algorithms the use regression have proved very effective in clustered environments, the results are most likely due to the increased effectiveness of SSL at this setting rather than a problem with MRRM, something that is also evident by the fact that SSL performs better at this setting with 3 collections than in the previous with 10 collections (table **5.14**). The above may indicate that the usage of relevance scores in clustered environments where a limited number of collections is selected to contribute documents by algorithms that make use of regression may have some merit, but again more research is needed to reach a definite answer to that minor question.

Summarizing the findings when collections report relevance scores, MRRM outperforms CORI in all settings in the K-means testbed, although with a smaller difference than previously. The two algorithms perform similarly the Uniform testbed with CORI performing marginally better than MRRM in most settings with the exception of the last where it is outperformed by a significant margin.

The comparison with SSL is particularly interesting. In the Uniform collection, SSL outperforms MRRM in most settings, with the exception of the last setting again where MRRM performs surprisingly well, recording even a 20% difference at precision at 5 documents. In the K-means testbed the two algorithms perform closely in most settings, with MRRM outperforming SSL in the first two settings and SSL performing better in the last.

Overall, it can be concluded that although MRRM was originally designed to function in score-lacking environments, its performance closely approximates, and even sometimes surpasses that of state-of-the-art algorithms in settings where document scores are available.

K-means			
Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.2840	0.3600	0.3480 (+22.5%) (-3.3%)
At 10 documents:	0.2520	0.3160	0.3100 (+23.0%) (-1.9%)
At 15 documents:	0.2227	0.2933	0.2827 (+26.9%) (-3.6%)
At 20 documents:	0.2060	0.2670	0.2600 (+26.2%) (-2.6%)
At 30 documents:	0.1753	0.2380	0.2340 (+33.5%) (-1.7%)

Table **5.15**: Precision with 3 collections selected at the Trec4 testbed, each one returning 1000 documents with Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

5.7 Summary

A new results merging algorithm was presented in this chapter. It was designed explicitly to function effectively in environments where the remote collections return only ranked lists of documents, without relevance scores. In order to compensate for the lack of information from the remote collections, the algorithm takes full advantage of the available, locally-stored documents, by considering them as representatives of the remote collections they originated from. In addition, it uses a centralized sample index as a representative of a

K-means			
Precision	**CORI**	**SSL**	**MRRM**
At 5 documents:	0.2840	0.3640	0.3520 (+23.9%) (-3.3%)
At 10 documents:	0.2520	0.3240	0.3120 (+23.8%) (-3.7%)
At 15 documents:	0.2227	0.2907	0.2813 (+26.3%) (-3.2%)
At 20 documents:	0.2060	0.2660	0.2570 (+24.8%) (-3.4%)
At 30 documents:	0.1753	0.2353	0.2300 (+31.2%) (-2.3%)

Table **5.16**: Precision with 3 collections selected at the Trec4 testbed, each one returning 300 documents with Relevance Scores. Percentages in parenthesis report the differences between MRRM and CORI (left) and SSL (right).

single global index.

The algorithm functions by first assigning collection-dependent documents scores to the returned ranked documents, by assuming that the correlation between ranking and score can be approximated through a logistic function. Using the common documents discovered between the remote collections and their sampled representatives, the algorithm estimates the parameters for the S-curve for each collection "on the fly", thus assigning collection-dependent scores to all the returned documents.

Secondly, it assigns collection-independent documents scores to the returned documents, by assuming that the correlation between the already estimated collection-dependent scores and collection-independent scores is linear. By locating the common documents found between the sample representatives of the selected collections and the centralized sample index, it estimates the parameters for the second regression for each collection "on the fly" and returns the produced final merged list to the user.

The feasibility of the algorithm, which is based on the discovery of enough common documents in both stages to perform the required regressions, was tested in a variety of environments and settings. In clustered environments, its performance was found to be "optimal" in the majority of cases, regardless of the number of returned documents from the remote collections. In other environments, although the number of common documents located was affected by the number of requested documents from the remote collections, the algorithm was found to function without hindrances in the majority of cases and only rarely was it necessary to utilize the fall back strategy.

The performance of the algorithm was measured in various settings against two state-of-the-art algorithms and was found to be stably better when no relevance scores were available, indicating that the heuristic assignment of scores in a linear manner is ineffective. Regardless of the number of returned documents and the number of collections contributing documents to the final merged lists, the algorithm was found to be very robust and effective. In particular, in clustered environments the algorithm noted very significant differences, indicating that it is particularly effective in environments where the documents are divided into collections by their content.

In environments where the remote collections return relevance scores, the comparison produced mixed results, with the new algorithm performing better in multiple occasions. Nonetheless, it would be premature to conclude that the utilization of relevance scores in environments they are provided is unnecessary as there were settings in which algorithms that made use of them, performed significantly better.

The algorithm could also be particularly practicable in mixed environments, where some of the collections provide scores and some don't, since it is able to function very effectively in score-lacking environments and achieve performance which approximates that of state-of-the-art algorithms making use of relevance scores.

CHAPTER 6

Hybrid Results Merging

6.1 Introduction

Having provided a solution to the results merging problem for completely uncooperative environments, where remote information sources return only ranked lists of documents, a different approach to viewing the problem is proposed in this chapter and a novel algorithm is presented that implements this new view of the problem.

Overall, the problem of results merging in distributed information retrieval environments has been approached by two different directions in research. *Estimation* approaches attempt to calculate the relevance of the returned documents through ad-hoc methodologies, i.e. weighted score merging or regression, while *download* approaches, download all the documents locally, partially or completely, in order to estimate "first hand" their relevance. For example, CORI, SSL and MRRM can be classified under the former category, while the *Distance Feature Algorithms* or algorithms that completely download the returned documents can be classified under the latter. Both approaches have their advantages and disadvantages.

Estimation approaches, with the exception of MRRM, usually rely on document relevance scores being returned by the remote collections in order to achieve maximum performance, Nonetheless, most search engines do not return relevance scores, but only ranked lists of documents. In addition to that, regression algorithms including MRRM, which have proved to be more effective than weighted scores merging approaches, rely on a significant number of overlap documents in order to function effectively. Although the frequency of encountering overlap documents in the result list of a search engine depends on a lot of factors, such as the level of sampling from the remote collection, the effectiveness of the remote retrieval algorithm and the difficulty of the query, usually a list of 300-1,000 documents is required from each collection, in order to locate the necessary overlap documents to calculate an accurate regression model. Nonetheless, in most modern information retrieval environment, remote collections return a maximum of 100 documents per result page, which means that retrieving more documents requires multiple interactions with the remote collections, adding a significant overhead to the retrieval process. The above facts make the utilization of regression methodologies at least problematic in realistic web environments.

On the other hand, *download* approaches download all the returned documents, completely or partially, in order to estimate "first hand" their relevance. They have the advantage that they are able to function with a limited number of returned documents from the remote collections (10-100), since most of the relevant documents in the final merged list are expected to appear high on the result list of the remote collections. In the context that they were tested by Craswell, Hawking, and Thistlewaite (1999), they proved to be more effective than estimation approaches but, there hasn't been a comparison between them and state-of-the-art estimation approaches. Their main disadvantages is the increased time and bandwidth overhead that they pose on the retrieval process in order to download, index and rank the returned documents and thus they are very expensive in terms of time and bandwidth.

The new algorithm that is introduced reconciles the above two approaches, combining their strengths, while minimizing their weaknesses. It is based on downloading a limited, selected number of documents from the remote collections and estimating the relevance of the rest through regression methodologies. In addition to that, the algorithm doesn't require document relevance scores from remote collections, making it practicable in realistic environments where remote collections return only ranked lists of documents. Thus it combines the best of both worlds; the increased effectiveness of the download approaches and the limited time and bandwidth overhead of the estimation approaches. Also, in the context of the experiments conducted in this chapter, a comparison of effectiveness is presented for the first time between download and state-of-the-art estimation approaches.

6.2 A Hybrid Approach

The motivation behind the new results merging algorithm is to function effectively and efficiently in realistic information seeking web environments. It combines *regression* and *download* methodologies (thus the name Hybrid), taking advantage of the benefits of both, while trying to minimize their disadvantages. Instead of requesting an excessive number of documents from the remote collections or alternatively downloading every single document, it selectively downloads a limited number of documents that are used as training data for the regression model.

The goal of algorithm is to estimate on-the-fly a model for each collection, on a per-query basis, in order to map collection-specific rankings (the ranking that documents attain at the remote collection in respect to a particular query) to collection-independent relevance scores (the relevance scores that documents would attain if the query had been submitted on a single global index). As previous approaches, it functions by considering the centralized sample index, as a representative of the single global index that would be created if all the documents were available for crawling and indexing.

Additionally, the algorithm actively tries to maximize its effectiveness while at the same time minimizing any occurring efficiency issues. Considering that the most useful documents from each collection are the ones usually returned at the top 10-20 ranks, a hypothesis that is later proved by the conducted experiments, it requests a limited number of documents from the remote collections. Under this context, it is considered as granted that there won't be enough training data, in the form of matching or "overlap" documents between the remote collections and the centralized sample index, discovered at the returned list in order to efficiently train the regression model. Thus, it incorporates the decision of downloading documents into the algorithm itself, explicitly minimizing the produced overhead, while maximizing the gained accuracy.

One should also note that in the web environment, a list of 100 results is probably much more "expensive" to view in terms of bandwidth than one with 10 or 20 documents. Likewise, in most cases it can be considered equally, or at least not overwhelmingly more, expensive to download (completely or partially) two or three documents from the result list of the first 10 documents than viewing 300 or more results from a remote collection, that would require multiple interactions with the remote collection. Therefore, the downloading of a limited number of documents can be considered as an acceptable solution that doesn't pose an excessive overhead to the process in comparison to other existing approaches.

6.2.1 Mapping Collection-Specific Ranks to Collection Independent Relevance Scores

One of the main issues concerning the proposed algorithm is the fact that an appropriate model would have to be determined that provides an accurate mapping of rankings to relevance scores. In previous approaches such as the one presented by Avrahami, Yau, Si, and Callan (2006) it was assumed that this mapping is linear.

As already discussed in section 5.3, Calvé and Savoy (2000) proposed a logistic function in order to provide an accurate mapping between rank and relevance. According to that work, the probability that document D_i is relevant given the log of its rank x_i, is given by the equation:

$$Prob\,[D_i \ is \ Rel|x_i] = \frac{e^{a+bx_i}}{1+e^{a+bx_i}} \tag{6.1}$$

The general expected correlation between probability of relevance and ranking using a logistic function with various parameters is demonstrated in figure 6.1 (reproduced from section 5.3).

In section 5.3 the important properties of the logistic function were pointed out and they remain valid in the context that it is used here: first of all, the results are always in the [0,1] area, and secondly, it can produce a large variety of curves, by simply modifying a and b.

6.2.2 Curve Fitting

Mapping rankings to relevance scores

Influenced by the work by Calvé and Savoy (2000) and extending the work in MRRM of the previous chapter, it is hypothesized that the correlation between the *collection-dependent* rank X of a document and the *collection-independent* relevance score Y is given by a logistic function:

$$Y = \frac{e^{a+bX}}{1+e^{a+bX}} \tag{6.2}$$

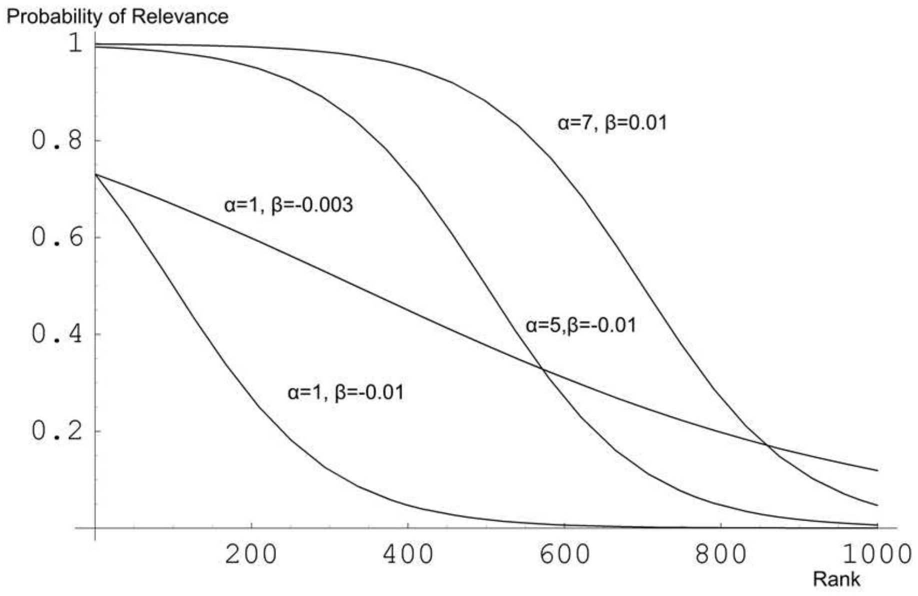


Figure 6.1: Plot demonstrating the expected correlation between rank (x) and probability of relevance (y) for various values of parameters a and b.

Although the function is the same as presented for MRRM in section 5.3, there is a subtle difference between the variable X and Y in that work and here. For MRRM, X expressed the $collection-dependent$ ranking of a document and Y was the the $collection-dependent$ relevance score. A simpler linear model was later applied in order to map the $collection-dependent$ relevance score into a final $collection-independent$ score.

In the context that the formula is used in the Hybrid algorithm, the two regressions are combined into one and Y is straightforward defined as the $collection-independent$ score. The new modeling proves to be more efficient and equally effective as the one presented in the previous chapter. That way, the storage requirements of the algorithm are also decreased, since there is no need to index the sampled documents of every information source separately, as with MRRM, in order to create *sample collection indexes*.

Applying the following transformations, already discussed in section 5.4.1, the above equation is modified into a liner one:

$$\frac{Y}{1-Y} = e^{a+bX} \tag{6.3}$$

$$ln\left(\frac{Y}{1-Y}\right) = a + bX \tag{6.4}$$

$$logit\,(Y) = a + bX \tag{6.5}$$

Estimations of the parameters a, b of the above model are needed, in order to calculate the S-curve for each collection, that will map *collection-dependent rank* X to *collection-independent relevance score* Y. Since equation (6.5) is a linear one, the estimation can be accomplished through linear regression.

A brief introduction to linear regression and the technique utilized in this thesis in order to estimate the parameters is provided in section 5.4.2, therefore it is not re-analyzed here.

A difference between the use of linear regression for the MRRM algorithm, as it was presented in chapter 5 and the Hybrid algorithm as it is presented here lies in the definition of tuples (x_i, y_i). For the Hybrid algorithm, the observations that are used for the estimation of the model are tuples (x_i, y_i), $i = 1, \ldots, n$ where x_i is the ranking at the remote collection of the i^{th} downloaded document (see below on which documents are utilized for regression), y_i is the relevance score of the document using the centralized sample index as a reference statistics database where n is the number of downloaded documents. The above fact derives from the already discussed difference in the definitions of X and Y in equation 6.2 between MRRM and Hybrid.

6.2.3 Selective Document Download

One of the most important aspects of the proposed algorithm is the integration of the download process as an indispensable part of the estimation of an accurate model. The aim of the process is to selectively download a limited number of documents from the remote collections, maximizing the effectiveness of the model. Excessive download would negate the efficiency advantages of the algorithm, while a very limited one would create an inaccurate model.

Two were the main issues concerning the details of the download decision. First of all, the rankings of the documents that would be downloaded and therefore utilized for regression had to be determined. Since most of the relevant documents are expected to appear at the top ranks it was evident that the download process would have to start from the top ranks. A decision had to be made so that appropriate sample data would be produced. Two approaches were possible. Documents could be downloaded either at a steady rate r (at ranks $n * r$, where $n \in N^*$) or at unequal intervals (i.e. at ranks 2^{2n-1}, $n = 1, 2, \ldots$). The former approach produced a more uniform sampling space, so it was adopted as the default choice. Further experiments about the effects of downloading documents at unequal intervals were also conducted and are also reported.

In order to further minimize the download overhead, any matching documents between the returned lists from the remote collections and the centralized sample index were utilized. Specifically, should any matching document be found at ranks:

$$\left[n * r - \frac{r}{2}, n * r + \frac{r}{2}\right), n = 1, 2, ...$$

the document at rank $n * r$ wasn't downloaded from the remote collection, but instead the matching document (and its corresponding rank) was used for regression. In effect, the rank space was divided into segments and at least one document was required to represent each segment. This step helped in reducing the download overhead into a minimum. Evidently, the actual existence of matching documents depend on a number of factors, such as the size of the remote collection, the effectiveness of the sampling phase, the difficulty of the query etc, but the process was designed to take advantage of them, aiming at minimizing the download overhead, regardless. In the experiments that were conducted r was set to 3, although early experiments showed that the algorithm's performance remains stable with other values as well. In order to provide a complete coverage of the algorithm, those experiments are also presented. Additionally, experiments were also conducted focusing on the number of downloaded documents, in order to demonstrate the efficiency gains of using already sampled documents.

Secondly, a stopping rule had to be determined that would end the download process, when the model was deemed sufficiently accurate. An adaptive download process was designed that estimated how accurate was the produced model after each document was downloaded and decided whether to continue or cease the download. Since, the focus is on estimating the "goodness of fit" of the regression, the coefficient of determination R^2 provided the appropriate solution:

$$R^2 = 1 - \frac{SS_E}{SS_T} \tag{6.6}$$

where

$$SS_E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, SS_T = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

SS_E is the *sum of squared errors* (i.e. the squared sum of the differences between the observed values y_i and the predicted values $\hat{y}_i$) and SS_T is the *total sum of squares* (i.e. the squared sum of the differences between the observed values y_i and their mean $\bar{y}$). Equation (6.6) calculates the coefficient of determination of the fit. The closer the value is to 1, the better the model fits the data.

In order to avoid overfiting the curve to the top 1 or 2 documents, the algorithm by default utilizes at least three top ranked documents (downloaded or pre-sampled) at the ranks determined by the above process and inserts an artificial document at rank $4*|N|$ with relevance score 0.001, where $|N|$ is the number of requested documents, effectively creating an extra tuple for regression at coordinates $(4*|N|, 0.001)$. The artificial document aims at better simulating the decline at the end of the graph (figure 6.1). The choice of the exact coordinates of the artificial document were set empirically and early experiments showed that they didn't affect the performance of the algorithm. It was nonetheless important that the artificial document was defined outside the actual rank space set by the actual documents so that it wouldn't interfere with the data provided by them (thus the choice of $4*|N|$ for the x-coordinate) and that it was close to zero (thus the choice of 0.001 for the y-coordinate) so that the final estimated curve would approach the x axis in an asymptotical manner.

The documents are added to the centralized sample index and their relevance scores are calculated. Again, the INQUERY retrieval algorithm was used to query the index, but any effective algorithm would do. Next, the algorithm estimates the parameters of model (equation 6.2) through regression using the obtained data and calculates the R^2 value. Should that value exceed a threshold, the model is deemed good enough and no further download occurs. If not, the algorithm proceeds at downloading the next document and re-estimates the R^2 value. The process continues until either the set threshold is exceeded or a maximum of 5 documents is downloaded. The threshold was heuristically set to 0.95, which produced a "good enough" model, without being too demanding on the process.

Note that the downloaded documents can afterward either be kept at the centralized sample index, practically updating it, or be deleted. In the line of experiments that were conducted, the second approach was adopted so that previous queries would not effect the results of subsequent queries.

The above process is repeated for each collection chosen by the source selection algorithm. Applying equation (6.2) with the estimated parameters for each remote collection, the algorithm assigns relevance scores to all the documents returned based on their ranking.

6.3 Experiment Setup

Although the environment that this thesis is particularly focused on, is a completely uncooperative one in which remote collections do not return documents relevance scores, in order to present the performance of CORI and SSL under the best possible light, they were allowed to make use of scores. As it was already demonstrated in section 5.6.2 and as it is generally expected, the performance of those algorithms in score-lacking environments, where estimates of documents relevance scores need to be made based on their ranking, is lower than the performance reported here.

The Download approach downloads every returned document (except for those that were already sampled, in which case the sampled document was used) and scores them locally, using the INQUERY retrieval algorithm. The Centralized Sample Index was again used as a "reference statistics database" in order to estimate global statistics. Neither the Download or the Hybrid approaches made any use of document relevance scores from the remote collections.

In order to limit the focus of the experimental results to the results merging part of the DIR process, CORI was used for source selection. For relevance judgments, both the routing and ad-hoc relevant judgments for the Uniform testbed and the K-means testbed were used, which is more in accordance with previous research.

6.4 Results

Tables **6.1** to **6.3** report the results of the initial experiments. Each table refers to one testbed. The left column indicates the number of documents that are requested from each remote collection (10, 100 or 1000 documents per collection). The Download and the new Hybrid approaches were only tested at the first two settings. This decision was based on the assumption that for the Download approach it is highly inefficient to download 1000 documents from each collection while the new algorithm was explicitly designed to function effectively with a limited number of returned documents. The last setting of 1000 documents is provided mainly for comparison reasons with the *estimation* approaches, especially SSL that cumulatively builds a more accurate model when there are more documents available.

Documents/ Collection	**Results Merging**	**P@5**	**P@10**	**P@15**	**P@20**
10 Docs	CORI	0.3820	0.3340	0.2987	0.2870
	SSL	0.3520	0.3480	0.3287	0.3225
	Hybrid	0.4400	0.4050	0.3813	0.3575
	Download	0.4660	0.4370	0.4080	0.3905
100 Docs	CORI	0.3820	0.3340	0.3027	0.2915
	SSL	0.3400	0.3540	0.3353	0.3205
	Hybrid	0.4340	0.4050	0.3793	0.3550
	Download	0.4640	0.4290	0.4027	0.3865
1000 Docs	CORI	0.3820	0.3340	0.3027	0.2915
	SSL	0.4100	0.3780	0.3500	0.3390

Table **6.1** Precision when 10 collections are selected at the Uniform testbed.

One of the first things that can be noted, applicable to all the test collections, is that the performance of most algorithms remains ineffected by the gradual increase of returned documents from the remote collections. The findings support our initial hypothesis that most of the relevant documents are returned on the top 10 results and very few relevant documents are added to the final merged list thereafter. In addition to that, the excess documents may introduce noise to the final merged lists, effectively deteriorating performance. In a realistic web environment it would potentially be much more beneficial to the retrieval process if more collections are added, potentially increasing the diversity and completeness of the final merged list, instead of requesting an increasing number of documents from a same limited number of collections. The only exception is, as expected, SSL whose performance generally increases as more documents are returned.

A second conclusion that is also common in all the test collections is that both of the approaches that incorporate some sort of downloading, Download or Hybrid, achieve performance that is persistently better than that of any *estimation* approach at most settings, with few exceptions. The above observation may imply that collection-dependent document relevance scores provide insufficient evidence for collection-independent document scores. Therefore, approaches that do not rely on such scores but take a more direct approach, such as downloading all or some, completely or partially (as it will be shown below) of the returned documents, thus not being susceptible to such estimation errors, are able to perform more adequately regardless of the performance of the underlying remote collections.

The intent behind the new Hybrid algorithm is to approximate the effectiveness of the Download approach, with as little overhead as possible. Clearly, the algorithm cannot avoid downloading some documents, which means that some overhead, even a limited, will be introduced to the retrieval process. What the Hybrid algorithm suggests is a limited compromise of efficiency, in exchange for a significant benefit in performance. The amount of introduced overhead will be studied subsequently and a comparison with the Download approach will be presented.

In the first setting of the Uniform testbed (Table **6.1**), where each collection returns 10 documents, the difference between the CORI - SSL and the Download - Hybrid groups is rather distinct and statistically significant, while the intra group differences aren't (i.e. the difference between Download and Hybrid isn't statistically significant). In the absence of sufficient training data SSL performs worse than CORI, although not in a statistically significant manner. The performance of CORI, Hybrid and Download remains mostly unchanged at the other settings, noting only small fluctuations, thus retaining their statistical significance. SSL's performance is surprisingly unstable, noting a decline in the second setting despite the additional documents and reaching a peak at the last. It is only at that setting, when the collections return 1000 documents each, that its performance comes near that of the Hybrid - Download group at the previous settings, but still doesn's manage to surpass it despite the excessive amount of documents and training data.

The K-means testbed (Table **6.2**) has been characterized as more difficult than the Uniform testbed, because of the heterogeneity of the individual collections and the skewness of the word distributions. It is possibly because of this fact that the Download approach attains such a notable performance gain, especially at P@5, while it retreats closer to the performance of the Hybrid in the following ranks. The performance of the Hybrid at the first setting is again above the performance of the CORI - SSL group, in a statistically significant manner. Again, CORI performs better than SSL, a behavior that doesn't persist in the second setting, where SSL's performance increases (in comparison to the Uniform testbed). Again, the Hybrid algorithm outperforms the best *estimation* approach, although

Documents/ Collection	**Results Merging**	**P@5**	**P@10**	**P@15**	**P@20**
10 Docs	CORI	0.3120	0.2600	0.2293	0.2040
	SSL	0.2560	0.1960	0.1960	0.1860
	Hybrid	0.3920	0.3280	0.2813	0.2520
	Download	0.5000	0.4080	0.3467	0.3110
100 Docs	CORI	0.3120	0.2600	0.2320	0.2230
	SSL	0.3160	0.3000	0.2653	0.2500
	Hybrid	0.3920	0.3280	0.2840	0.2520
	Download	0.4880	0.4140	0.3787	0.3510
1000 Docs	CORI	0.3120	0.2600	0.2320	0.2230
	SSL	0.3720	0.3280	0.3160	0.2920

Table **6.2** Precision when 10 collections are selected in the K-means testbed.

in a statistically significant manner only at P@5. At the last setting, where each collection returns 1000 documents the performance of SSL approximates very closely that of the Hybrid - Download approaches.

Documents/ Collection	**Results Merging**	**P@5**	**P@10**	**P@15**	**P@20**
10 Docs	CORI	0.1326	0.0926	0.0772	0.0705
	SSL	0.1095	0.1000	0.0856	0.0795
	Hybrid	0.2463	0.1884	0.1628	0.1484
	Download	0.2611	0.2211	0.1902	0.1721
100 Docs	CORI	0.1326	0.0926	0.0772	0.0705
	SSL	0.1516	0.1453	0.1277	0.1163
	Hybrid	0.2421	0.1884	0.1635	0.1479
	Download	0.2774	0.2242	0.1951	0.1742
1000 Docs	CORI	0.1099	0.0704	0.0573	0.0507
	SSL	0.1516	0.1493	0.1347	0.1216

Table **6.3** Precision when 100 collections are selected in the Web testbed.

The Web testbed (Table **6.3**) presents very interesting challenges because of the level of distribution that it offers and the diversity of the sizes and topicality of the underlying collections. In order to have meaningful measurements, it was decided to select 100 collections per query, which even though may seem excessive is at the same percentile scale as the previous testbeds (10% of the available collections). Early experiments showed that only few relevant documents were returned at the first 10 collections, making the comparison between the algorithms trivial. In this testbed, the performance differences between the CORI - SSL and the Hybrid - Download group are more profound in comparison to the previous testbeds, giving a possible hindsight about the actual performance of the approaches in realistic web environments. In the first setting, where each collection returns 10 documents, the difference between the Hybrid algorithm and the best performing *estimation* algorithm varies around +85% at any precision measurement, a statistically significant difference. In comparison, the difference between the Hybrid and the Download approach is not statistically significant. At the second setting, the difference between the two groups is reduced, but still remains at significant levels. SSL's performance at the last setting surprisingly remains at the same level as the second, possibly because the collections do not return as much as 1000 documents, thus not approaching the performance of the Hybrid algorithm at any setting.

Overall, the performance of the algorithms that incorporate some sort of downloading of documents, Hybrid or Download, almost always exceeds that of *estimation* approaches and in most cases with a statistically significant difference. It is only in the, unrealistic, case that each collection returns 1000 documents per query that SSL approximates that performance, but only rarely does it surpass it.

6.4.1 Studying the Efficiency Issues

In this section, a study the overhead that the Hybrid and the Download approaches introduce to the retrieval process will be presented. Thus, the number of documents that each approach downloads at the above settings was counted. As noted above, the Hybrid algorithm incorporates a progressive download criterion, while the Download approach downloads every returned document, unless it has already been sampled in the query-sampling phase. Results are presented on table **6.4**.

As expected, when remote collections are requested to return 100 documents, the Down-

Documents/	**Results**	**Test Collection**		
Collection	**Merging**	**Uniform**	**K-means**	**Web**
10	Hybrid	2.5 (-73%)	2.2 (-75%)	1 (-81%)
Docs	Download	9.3	8.8	5.4
100	Hybrid	2.5 (-97%)	2.2 (-98%)	1 (-98%)
Docs	Download	95.9	89.9	40.6

Table **6.4**: Number of downloaded documents per collection per query. The left column indicates the number of requested documents from each collection.

load approach downloads an excessive number of documents, while the Hybrid algorithm utilizing the progressive download methodology described above, limits the number of downloads to a minimum. Taking into consideration that the effectiveness gains in this setting are trivial when compared to the first setting where collections return 10 documents, the analysis will be focused to that setting, which is also more fair for comparison to the Download algorithm. The results are presented here only for completeness reasons.

When the remote collections return 10 documents each, the Download algorithm downloads on average 9.3 documents in the Uniform testbed and 8.8 documents in the K-means testbed. In comparison, the Hybrid algorithm downloads a maximum of 2.5 documents in either setting, noting a efficiency gain of more than 70% in both cases. A question may arise here since as it was mentioned above, the Hybrid algorithm by default utlizes 3 documents from each collection to avoid overfitting the curve to the top 1-2 documents. The answer lies in the usage of the already sampled documents, which provide some of the necessary representatives of the divided rank space thus demonstrating that using already sampled documents indeed help the efficiency of the algorithm. In addition to that, in most cases the initially produced model was above the set R^2 threshold and therefore the algorithm didn't resort to additional downloading.

The new Web testbed presented some surprising results. The query-based sampling process in this testbed proved to be unexpectedly efficient, since the Download algorithm had to download only 5.4 documents on average per collection. Again, the Hybrid algorithm proves to be much more efficient, resorting to downloading only 1 document per collection. The skewness of the size distribution of the collections may provide the explanation for the above results, potentially making the most the useful documents from the collections already available at the query-sampling phase, thus minimizing the needed downloading on query time.

The above results demonstrate the efficiency gains of the algorithm in comparison to the Download approach. Let it be noted, that in realistic web environments both the algorithms could download the cached version of the returned document which most search engines provide, instead of the actual one, thus reducing any delay in producing the final results. Still even in this case, the Hybrid approach remains a much more efficient solution, downloading on average only 2-3 documents per collection.

6.4.2 Attempting a Partial Download approach

For the next line of experiments, the performance of the Hybrid algorithm with partial download approaches was compared. Under this context, instead of downloading documents completely, only a part is downloaded, based on a size threshold. The above approach aims at laxing the efficiency constraints of the original Download approach, in which every returned document is completely downloaded. In parallel, the Hybrid algorithm was also tested under the partial download approach, in which the selected documents are also downloaded partially, in order to provide a measurement of its robustness. It should be noted that the partially downloaded documents are only utilized to produce the final merged document list and not be served to the user, should he/she choose to view the document. Should that happen, the user would be directed to the actual remote document. Under this context, a partial document approach is always applicable when there are significant constrains on the download overhead that can introduced into the process.

Since the initial experiments showed that requesting more than 10 documents from the remote collections did not attribute to a significant increase in precision, the algorithms were tested only in this setting. Taking into consideration that the average size of a single trec document is roughly 6kbytes, two size thresholds were tested: 2kbytes and 3kbytes, which approximately translate into downloading one third and half of the returned documents respectively. Results for all the test collections are reported at Tables **6.5** to **6.7**.

Generally, the performance of both algorithms using partial downloading is decreased, but not in a significant manner. The performance of the Hybrid algorithm is always at par with the Download approach, as in the previous settings where documents are completely downloaded. The analysis will focus on the comparison of the results of this setting to the

Uniform					
Size Threshold	**Results Merging**	**P@5**	**P@10**	**P@15**	**P@20**
2 kbytes	Hybrid	0.3980	0.3620	0.3373	0.3285
	Download	0.4420	0.4040	0.3873	0.3680
3 kbytes	Hybrid	0.4200	0.3870	0.3573	0.3410
	Download	0.4600	0.4160	0.3880	0.3755

Table **6.5**: Precision when 10 collections are selected to return 10 documents each at the Uniform testbed. The left column indicates the size threshold for the downloaded documents.

ones produced by the *estimation* approaches (Tables **6.1** to **6.3**) to examine whether the partial downloading has a significant effect on the performance of the Hybrid algorithm or whether the conclusions drawn at the initial experiments are still valid in this setting.

In the Uniform testbed, when only 2kbytes from the selected documents are downloaded, the performance of the Hybrid remains above the CORI - SSL group (Table **6.1**). It is only at the setting where each collection returns 1000 documents that the SSL manages to overpass the Hybrid approach, a result that doesn't persist when the threshold is increased to 3kbytes.

It is worth making a point here concerning the bandwidth requirements of the algorithms. In this setting for example, the Hybrid algorithm roughly downloads 10 results lists of 10 documents each plus 75kbytes of documents (3kbytes per document $*$ 2.5 documents (on average, see Table **6.4**) $*$ 10 collections). In comparison, the SSL algorithm needs to view 1000 results, which translate into 10 interactions with each remote collection, each interaction returning 100 results. Although the actual sizes of the results lists of search engines may vary from one to another, one can assume that the latter method would be at least problematic in realistic web environments.

In the K-means testbed, the conclusions are generally similar as the ones observed in the Uniform collection. Hybrid manages to outperform the CORI - SSL group in all the settings (Table **6.2**), with the exception of the last where collections return 1000 documents in which SSL performs better, but not in a statistically significant manner.

Finally, in the Web testbed, the performance of the Hybrid algorithm, even at the

K-means					
Size Threshold	**Results Merging**	**P@5**	**P@10**	**P@15**	**P@20**
2 kbytes	Hybrid	0.3640	0.3080	0.2667	0.2360
	Download	0.4480	0.3800	0.3133	0.2780
3 kbytes	Hybrid	0.3680	0.3200	0.2773	0.2450
	Download	0.4440	0.3780	0.3307	0.2910

Table **6.6**: Precision when 10 collections are selected to return 10 documents each at the K-means testbed. The left column indicates the size threshold for the downloaded documents.

Web					
Size Threshold	**Results Merging**	**P@5**	**P@10**	**P@15**	**P@20**
2 kbytes	Hybrid	0.2168	0.1547	0.1312	0.1163
	Download	0.2463	0.2053	0.1754	0.1574
3 kbytes	Hybrid	0.2274	0.1632	0.1404	0.1258
	Download	0.2526	0.2137	0.1796	0.1600

Table **6.7**: Precision when 100 collections are selected to return 10 documents each at the Web testbed. The left column indicates the size threshold for the downloaded documents.

2kbyte setting steadily remains well above that of any *estimation* algorithm. The differences increase when the size threshold is set to 3kbytes, but only marginally.

In general, the results demonstrate the robustness of the algorithms that incorporate a downloading process, even if documents are downloaded only partially. Additionally, the performance of the Hybrid algorithm steadily remains above that of *estimation* approaches in most settings, making it a very efficient and robust approach.

6.4.3 Downloading at different intervals

In the experiments that were presented this far, the Hybrid algorithm was set to download documents at equal intervals. Additionally, the rate was heuristically set to 3, based on preliminary experiments. It would be interesting to see whether adopting a different rate would effect the performance of the algorithm. One would expect that it remains robust to such modifications, not recording great differences in precision.

In this section, the effect of downloading documents at different equal intervals are presented and discussed. The experiments were run on the same testbeds described above requesting 10 documents from each remote collection, varying only the interval. The original algorithm used a rate of 3, so the algorithm was also tested with rates of 2, 4 and 5, as these provided more than one tuple in the first 10 results.

The results are presented on Table **6.8**. The original adopted rate is also presented for comparison reasons. The percentages in parenthesis report the differences in precision in reference to the original rate. The best performance is marked with bold in each precision setting.

Testbed	R	P@5	P@10	P@15	P@20
Uniform	2	**0.4420** (±0%)	**0.4170** (+3%)	0.3793 (-1%)	**0.3620** (+1%)
	3	0.4400	0.4050	**0.3813**	0.3575
	4	**0.4420** (±0%)	0.4020 (-1%)	0.3760 (-1%)	0.3535 (-1%)
	5	0.4320 (-2%)	0.3920 (-2%)	0.3747 (-2%)	0.3540 (-1%)
K-means	2	**0.4000** (+2%)	**0.3380** (+3%)	0.2907 (+3%)	0.2540 (+1%)
	3	0.3920	0.3280	0.2813	0.2520
	4	0.3680 (-6%)	0.3300 (+1%)	0.2853 (+1%)	0.2550 (+1%)
	5	0.3800 (-3%)	0.3200 (-2%)	**0.2920** (+4%)	**0.2570** (+2%)
Web	2	0.2316 (-6%)	0.1874 (-1%)	**0.1656** (+2%)	0.1463 (-1%)
	3	**0.2463**	**0.1884**	0.1628	**0.1484**
	4	0.2189 (-11%)	0.1874 (-1%)	0.1572 (-3%)	0.1389 (-6%)
	5	0.2189 (-11%)	0.1716 (-9%)	0.1544 (-5%)	0.1342 (-10%)

Table **6.8** Precision with varying equal intervals of downloading documents.

In general, the performance of the algorithm remains mostly stable on all testbeds. Varying the interval of downloading documents does produce some fluctuations in precision as expected, but at no point do those differences alter the conclusions drawn earlier about the effectiveness of the algorithm in respect to other approaches.

Specifically, in the Uniform and the K-means testbed, with the exception of P@5 in the latter, the differences range between -3% to $+4\%$, which are rather insignificant. One minor exception is the P@5 at the K-means testbed, where the difference between the original algorithm and an interval of 4 reaches a -6% drop, but is later equalized at P@10 and the following precision measurements.

In the Web, the differences are somewhat more substantial, noting significant drops, particularly at P@5, while are mostly leveled in latter precision measurements. The heterogeneity of the particular testbed and the skewed distribution of relevant documents may justify for the observed results, but more research may be needed in order to fully understand the noted fluctuations.

The above results demonstrate the robustness of the algorithm in reference to the adopted rate for downloading documents. Although some differences in performance are noted when the interval is changed, these are mostly insignificant. The optimal rate does seem to be 3, which has the most stable performance across all testbeds, justifying the original choice. This is probably because it successfully combines focusing in the top ranked documents with a somewhat dense and uniform sampling space across the ensuing ranks.

6.4.4 Downloading at unequal intervals

In order to further examine the robustness of the algorithm, its performance was also tested when downloading documents at unequal intervals. A variety of functions was implemented to provide a diversity of downloading schemes. Specifically, documents were downloaded at ranks 2^n, where n=0,1,2,... (i.e. ranks 1,2,4,8), 2^{2n-1} (i.e. ranks 1,2,8) and 3^n (i.e. ranks 1,3,9).

Each one of those functions was chosen because of the focus they assign to different ranks. The first one focuses particularly heavily on the top ranked documents, the second one on the first two top-ranked documents and also utilizes a last one toward the end of the result list and the third one provides a rather sparse sampling space. Results are reported on Table **6.9**. Again, percentages in parenthesis report the differences in precision in reference to the original reported algorithm.

The performance of the algorithm again remains mostly stable. The observed differences between the various downloading schemes and the original approach lie between -4% and $+6\%$ and aren't statistically significant. The 2^n approach seems to be performing somewhat better at all the testbeds in comparison to the original approach, while the rest of the approaches are mostly at par with the original version of the algorithm, performing sometimes

Testbed	R	P@5	P@10	P@15	P@20
Uniform	2^n	**0.4380** (±0%)	0.4110 (+1%)	0.3807 (±0%)	**0.3605** (+1%)
	2^{2n-1}	0.4360 (-1%)	**0.4130** (+2%)	0.3780 (-1%)	0.3595 (+1%)
	3^n	0.4300 (-2%)	0.3960 (-2%)	**0.3820** (±0%)	0.3555 (-1%)
K-means	2^n	**0.4040** (+3%)	0.3360 (+2%)	**0.2920** (+4%)	0.2530 (±0%)
	2^{2n-1}	0.3880 (-1%)	**0.3380** (+3%)	0.2853 (+1%)	**0.2540** (+1%)
	3^n	0.3920 (±0%)	**0.3380** (+3%)	0.2827 (+1%)	0.2520 (±0%)
Web	2^n	**0.2442** (-1%)	0.1884 (±0%)	0.1719 (+6%)	**0.1500** (+1%)
	2^{2n-1}	0.2358 (-4%)	**0.1958** (+4%)	**0.1733** (+6%)	**0.1500** (+1%)
	3^n	0.2358 (-4%)	0.1947 (+3%)	0.1649 (+1%)	0.1442 (-3%)

n=0,1,2 in all of the above settings. 2^{-1} was set to 1.

Table **6.9** Precision with varying unequal intervals of downloading documents.

slightly better and sometimes slightly worse, always within an acceptable margin.

In summary, the algorithm has been tested with a variety of different schemes for downloading documents, both at equal and unequal intervals and it was found that its performance remains mostly stable, noting minor fluctuations which rarely become statistically significant. The above results demonstrate the robustness of the proposed approach regardless of the parameters originally chosen.

6.4.5 Using a different reference statistics database

In the experiments that were conducted so far, the documents that where downloaded locally during the query-based sampling phase as a “reference statistics database” were used, as this seemed like a natural choice given the fact that they were readily available. The added advantage of this choice was a reduced download overhead, since some of the documents that were returned during the experiments were already sampled.

The primary reason for the utilization of a reference statistics database is to provide a representative for a single global database, thus aiding in the estimation of global-wide statistics, such as inverse document frequency. One may view the prerequisite of the creation of such a database during the sampling phase as a serious drawback of the proposed results merging algorithm, making it dependable on the effectiveness (or not) of previous phases of the distributed information retrieval process. Additionally, there is an interest in examining the effects of using a different, potentially more general, document corpus as a reference statistics database. Should the performance of the algorithm remain the same, that would effectively mean that the algorithm is independent of the query-based sampling phase, and could thus be utilized by making use of a random document corpus as a reference database.

For this reason, the following line of experiments was conducted. The centralized sample indexes from the Uniform and Web testbeds were interchanged and the performance of the algorithm using them as reference databases were measured. In effect, the centralized sample index created for the Web during the query-sampling phase was used in experiments with the Uniform / K-means testbeds as a reference statistics database, and vice versa. Note that the difference of the two corpora is rather substantial, based on the nature of the actual testbeds. The documents in the Web collection are web documents, covering various subjects, while the trec documents are newswire articles, written in a more authoritative manner. More on the differences of the two testbeds can be found on section 6.3.

Additionally, since previous research by Shokouhi, Zobel, Scholer, and Tahaghoghi (2006) has indicated that query-based sampling doesn't necessarily provide a true random sample of documents and is rather biased toward long documents, experiments with a true random selection of documents from the Uniform and Web collections were also conducted. Thus, two experiments were ran using different references statistics databases; one using a database that was produced with query-based sampling on a different corpus (this setting was named 'wtQryBased" and "trQryBased" respectively) and a second, using a database comprised of randomly selected documents ("wtRandom" and "trRandom" respectively) again from a different corpus.

The goal of those experiments is to see whether the use of a completely different document corpus (both biased and unbiased) would still provide helpful estimations for global-wide statistics, or whether the produced statistics would be so inaccurate, resulting in a significant drop of effectiveness of the algorithm.

The algorithm was also tested in two additional settings, as controls. In the first, no centralized sample index was utilized. Only the documents that were downloaded at each query were used as a reference statistics database. It is clear, that in this case the database is heavily biased toward documents that are at least somewhat relevant to the query (potentially all the documents would contain the query terms, thus providing largely inaccurate idf estimations) and in no way does it provide a representative of a single global collection, but it would be very interesting in exploring the effect of using such a database on the algorithm, testing whether the algorithm can function in a heavily biased setting.

Lastly, slightly altering the above setting, the experiment started with an empty centralized sample index, but the documents that were downloaded at each query were added to it, effectively building a reference statistics database incrementally (this setting was named "Dynamic"). The last setting could be employed in a realistic environment where the algorithm would start functioning without any locally-stored documents and use the downloaded documents as reference for future queries.

Results are reported on Table **6.10**. Note that the original version of the algorithm was used for this line of experiments.

In the Uniform testbed, the lack of a reference statistics database ("None") has a significant impact on the algorithm. The estimates produced are obviously inaccurate, producing an ineffective final ranking. On the contrary, the use a different reference database has little impact on the algorithm indicating that the algorithm could indeed function with a completely different document corpus in producing global-wide statistics.

What is surprising is the performance at the "dynamic" setting, which is only slightly deteriorated compared to the original. Conducting a per query comparison, it was found out that the differences weren't so substantial at the initial queries as one may have expected, as at these queries the reference corpus is still at a very initial stage, but instead they were scattered amongst the queries, in a rather unpredictable manner.

Combining these results with the significantly deteriorated performance of the algorithm when no reference database is available, one may reach the conclusion that for some more difficult queries (e.i. where the document frequency of the query terms vary significantly) a good reference corpus is helpful, while for others it is unnecessary.

The results are quite different at the K-means testbed. There are no substantial differences between the approaches, including the case where no reference database is being used. These results are somewhat surprising and partly come in contrast with the results from the tre123 testbed, potentially indicating that in well content-based defined collections with a limited relevant document distribution amongst collections, the use of a reference database is indeed redundant. It is believed that this is because any relatively successful source selection strategy will be able to locate the most promising collections, thus providing

Testbed	**CSI**	**P@5**	**P@10**	**P@15**	**P@20**
Uniform	None	0.3880 (-12%)	0.3610 (-11%)	0.3393 (-11%)	0.3310 (-7%)
	Dynamic	0.4300 (-2%)	0.3960 (-2%)	0.3720 (-2%)	0.3495 (-2%)
	wtQryBased	**0.4440** (±0%)	0.4050 (±0%)	**0.3793** (-2%)	**0.3560** (±0%)
	wtRandom	0.4380 (±0%)	**0.4060** (±0%)	0.3753 (-2%)	0.3475 (-3%)
K-means	None	0.4000 (+2%)	0.3300 (+1%)	0.2747 (-2%)	0.2470 (-2%)
	Dynamic	**0.4200** (+7%)	**0.3480** (+6%)	**0.2880** (+2%)	**0.2550** (+1%)
	wtQryBased	0.3840 (-2%)	0.3200 (-2%)	0.2733 (-3%)	0.2510 (±0%)
	wtRandom	0.4120 (+5%)	0.3360 (+2%)	0.2827 (+1%)	0.2510 (±0%)
Web	None	0.2189 (-11%)	0.1884 (±0%)	0.1621 (±0%)	0.1500 (+1%)
	Dynamic	**0.2421** (-2%)	**0.1916** (+2%)	**0.1677** (+3%)	**0.1521** (+2%)
	trQryBased	0.2295 (-7%)	0.1811 (-4%)	0.1663 (+2%)	0.1479 (±0%)
	trRandom	0.2063 (-16%)	0.1632 (-13%)	0.1544 (-5%)	0.1363 (-8%)

Table **6.10** Changing the statistics reference database.

a significant number of relevant documents for the final merging. Still, even at this setting one should not underestimate the importance of the results merging phase, as demonstrated by the significant differences between the *download* and the *estimation* approaches on Table **6.2**.

Lastly, in the Web testbed, there is a general drop of effectiveness at P@5, which usually doesn't persist at later precision measurements, closely approximating the original recorded performance. The only exception to the above is when the algorithm utilizes a random sampling from the Uniform collection, where the performance remains notably low (but still much higher than any *estimation* approach). The reasons why this may be happening aren't very clear, but it is assumed that a random selection of documents from the Uniform collection, may produce a bad selection of documents.

In summary, the above experiments indicate that the use of some sort of reference database does aid the performance of the algorithm. The utilized corpus need not be specific to the testbed, but may as well be a general corpus. The absence of any reference database is not suggested as the performance of the algorithm becomes rather unstable at this setting.

6.5 Summary

In this chapter, a novel results merging algorithm that builds on the ideas that were first put forth in the previous chapter was presented. The algorithm offers a hybrid solution to the two directions from which the problem has been approached in research, *estimation* and *download*, by combining their strengths and minimizing their drawbacks. It is based on downloading a limited number of selected documents from the remote collections with the help of which it creates a model on-the-fly on a per query basis for the estimation of the relevance of the rest through regression methodologies. In addition to that, the algorithm doesn't make use of document relevance scores from remote collections, making it practicable in completely uncooperative environments where they aren't provided.

In the experiments that were conducted it was shown that the effectiveness of the proposed algorithm is almost always above the best of the *estimation* algorithms, while approximating that of *download* approaches. Even if the selected documents are partially downloaded, the impact on the performance of the algorithm isn't significant, making it a robust solution in realistic web environments. It was also demonstrated that the new approach is much more efficient in terms of bandwidth overhead that *download* approaches, downloading on average 78% less documents.

Further experiments demonstrated that the algorithm is robust to variations of the originally chosen parameters, noting mainly minor fluctuations in performance. Additionally, it was shown that the algorithm is not dependent on the query-based sampling phase and could function quite effectively by making use of a general, but carefully chosen, document corpus.

CHAPTER 7

Conclusions - Future Directions

7.1 Introduction

Digital information is nowadays created at unprecedented rates. A significant amount of this information is stored and accessed through the World Wide Web. General purpose search engines, such as Google or Yahoo!, provide an easy mechanism for users to discover and access it in order to satisfy their information needs.

Nonetheless, they face serious challenges that limit their operation. They cannot index the whole of the Web, because of its prohibitive size and rapid rate of growth, therefore information that is otherwise available cannot be discovered by their users. Additionally and more importantly, they cannot reach or aren't allowed to analyze the content of a significant number of websites, collectively known as *invisible Web*.

The information that is stored in invisible websites is estimated to be an order of magnitude greater that the data that is available through general purpose search engines and is generally thought to be of significant quality, providing authoritative information produced by government agencies and information professionals.

A prospective solution to the above issues is provided by Distributed Information Retrieval (DIR). Federated Search systems act as intermediaries between users and information sources, offering the former the capability of simultaneously submitting queries to multiple remote collections, whether they are general purpose search engines or invisible websites, through a single interface.

Additional applications of Distributed Information Retrieval approaches have also been introduced in research and industry. Enterprise search systems, i.e. information retrieval systems that are tailored to analyze data in an enterprise setting and advanced peer-to-peer applications, able to offer more than simple filename retrieval, are two of the most prominent examples.

This dissertation offers prospective solutions to the issues that are pertinent to the DIR process in completely uncooperative environments, where information sources are assumed to return only ranked lists of documents in response to queries, with no additional information, such as relevance scores, snippets, etc.

This chapter concludes the dissertation by providing a general overview of the approaches that have been presented and comments on the contributions of the thesis. Specifically, section 7.2 summarizes the new approaches to Distributed Information Retrieval that have been put forth in this thesis. Section 7.3 reviews and analyzes the contributions of the work and section 7.4 provides some potential future directions of research to extend the work that has been presented in this thesis.

7.2 Summary of research

A novel source selection algorithm, named Collection-integral Source Selection (CiSS), was presented in chapter 4. The algorithm is based on modeling the available information sources as integrals in the rank - relevance space produced by a sample of the documents that they contain. Based on the above modeling, the algorithm explicitly focuses on addressing the two goals of source selection; *high recall*, which is important for source recommendation applications and *high precision* which aims to produce a high precision final merged list.

In order to achieve the former goal, the algorithm selects the collections that cover the largest area in the space. For the latter goal, the algorithm divides the area covered by the remote collections into segments, each representing an estimation of the relevance of the returned lists of document of the collections in regard to the submitted query and thus estimates the benefit of including them in the merged list. Based on that estimation, the optimal distribution of documents is calculated in order to maximize the area covered by the final document list that will be returned to the user and therefore maximize the overall gain.

The algorithm was tested in a wide set of environments and settings and its performance was found to be equivalent or better than other state-of-the-art source selection algorithms. In particular, in recall-oriented settings, the experiments showed that the algorithm is very effective in locating collections that have a significant number of relevant documents, regardless of their size and isn't biased towards large collections. In uniform-sized testbeds, the algorithm was able to perform at least as effective as other approaches and more effective when considering only a small number of collections. Lastly, in clustered and web environments, the algorithm was able to retain a performance which was at par with the best of source selection algorithms.

In precision-oriented environments, the algorithm was able to outperform previous approaches in the majority of settings and environments. Considering the fact that the algorithm had less documents available for merging than other approaches, it is evident that it provides a very effective solution to the high-precision goal.

Multiple-Regression Rank Merging, a new results merging algorithm, was presented in chapter 5. It is explicitly designed to function effectively in environments where the remote collections return only ranked lists of documents, without relevance scores. The algorithm is based on defining the correlation between document ranks and relevance scores as a logistic function. It operates in two phases. Initially, using the sampled documents from the remote collections, it maps collection-specific document rankings to collection-specific document scores via a logistic curve whose parameters are dynamically estimated and subsequently, using the centralized sample index and through linear regression, assigns collection-independent relevance scores to the previously computed scores, thus producing a final merged document list.

The feasibility of the algorithm was tested in a variety of environments and settings and it was found to function without hindrances in the majority of cases and only rarely was it forced to utilize the fall back strategy. The performance of the algorithm was measured in various settings against other state-of-the-art algorithms and was stably better when no relevance scores were available. In environments where the remote collections provide relevance scores, the comparison produced mixed results, with the new algorithm nonetheless performing better in multiple occasions.

A second Hybrid results merging algorithm, that extends the initial modeling of the MRRM algorithm and unifies the two directions from which the problem of results merging has been approached in research, was presented in chapter 6. The algorithm downloads a limited, selected number of documents from the remote collections and estimates the relevance of the rest through a single regression. It combines the increased effectiveness of the approaches that download documents with the limited time and bandwidth overhead of the approaches that estimate their relevancy through ad-hoc methodologies.

In the experiments that were conducted it was shown that the effectiveness of the proposed algorithm is almost always above the best of the estimation algorithms and approximates that of download approaches. Even if the selected documents are partially downloaded, only a small impact on the performance of the algorithm was observed, making it a robust solution in realistic web environments. Additionally, the new approach was found to be much more efficient in terms of bandwidth overhead that download approaches.

7.3 Contributions

One of the original motivations of this thesis was the development of algorithms that are able to function effectively and efficiently in environments where remote collections provide no cooperation. Previous approaches to results merging assumed the report of document relevance scores from information sources or the existence of snippet information. The algorithms therefore were domain-dependent and could only by applied to environments where certain conditions were met. Taking into consideration the wide applicability of DIR methodologies, it becomes evident that algorithms that require any form of cooperation from information sources or additional information are severely limited in their operation and capabilities.

The two results merging algorithms that are presented in this thesis, *MRRM* and *Hybrid* results merging, make no such assumptions and can be characterized as domain-agnostic. As a direct consequence, they can be applied to a wide variety of environments and settings with little or no modification. The generality of the approaches is considered a very important element that significantly augments their adaptability and practicality in real-world applications.

The Hybrid results merging algorithm provides for the first time a methodology that unifies the two approaches that have been suggested in research on tackling the results merging problem, *download* and *estimation*. The algorithm is able to combine the advantages of both approaches by achieving a performance that is similar to the former while at the same time being efficient and economical as the latter.

Both of the approaches are based on a novel methodology that was proposed for mapping document ranks to relevance scores during query-time, without the need of a training phase. Although the above mapping was utilized under specific scenarios and purposes, there are significant merits in the general approach that was followed and its applicability can easily be extended to other domains. For example, certain metasearch techniques require that search engines report document relevance scores in order to provide optimal effectiveness. The above methodology could be potentially used in order to map ranks to scores and therefore enhance the effectiveness of such approaches in environments where scores aren't provided.

The source selection solution that was proposed in this thesis presents a novel modeling of information sources as regions in a space created by the documents that they contain. The provided modeling offers a complete framework for addressing the source selection problem and its two subproblems of attaining high-recall and high-precision. An important aspect of the suggested modeling is that it captures real-world observations and widely accepted notions in IR, i.e. such as the non-uniformity of the sampling, the existence of unretrievable documents and the importance of top ranking documents for precision into a theoretical framework, by applying simple modifications to the initial general concept.

One of the key novelties of the approach, in precision-oriented environments is that it only requires the number of documents in the final merged list be specified by the user, dynamically estimating the number of collections that are selected as well as the number of documents that are retrieved by each. Although no user-oriented studies were conducted, the above feature is very important in realistic environments, where information about the distribution of relevant documents is unknown, making the retrieval process much more automated and simple as it alleviates the need for the user to heuristically set parameters.

7.4 Future Directions

The presented modeling of information sources as it was presented in this thesis as regions in 2-dimensional space takes into consideration only the documents that are contained in the collections as indicators of appropriateness. The above criterion may be considered as the most important in a great majority of domains and environments. Nonetheless, in real-world applications, the contents of a document collection may be only one of many factors that may contribute to its appropriateness. Other relevant factors may include the effectiveness of the retrieval algorithm in the remote collections, the average response time of the source, its proximity (especially important in p2p applications), occurring costs for submitting queries or viewing documents from the collection etc. The general framework that is presented here of modeling information sources can be extended in order to incorporate such factors into the proposed modeling, thus producing an integral not in 2-dimensional space, but in n-dimensional space, where $n-1$ would be the number of factors under consideration.

The parameters that were used for CiSS were defined and experimentally specified during the initial stages of development and implementation of the algorithm. In the experiments that were presented these parameters were heuristically set to constant values that remained unchangeable regardless of the underlying retrieval environment and information sources. As part of the future work that can be directed into extending the source selection approach that was presented, a study of the decline of relevance of the retrieved documents from remote collections will provide significant information that will aid in alleviating the need for the heuristic setting of these parameters. The results of this study will provide a general framework for providing estimations of the probable decrease of relevance of the retrieved documents dynamically and during query-time for each available information source therefore further increasing the applicability and effectiveness of the proposed algorithm.

Although the logistic model that was used by MRRM, the Hybrid algorithm and indirectly by CiSS in order to provide a mapping of document ranks to relevance scores proved quite effective, it is possible that another model may prove more precise. The potential model would need to posses certain properties, such as the ability to produce a wide variety of curves by modifying its parameters and provide an easy way to estimate those parameters, preferably without requiring human relevance judgments. Such a model would potentially enhance the performance of the proposed algorithms and provide significant results that may be applicable in other areas of Information Retrieval, such as filtering and routing tasks, etc.

Although the algorithms that were presented in this thesis are up to a certain point domain and application agnostic and can easily adapted to a variety of fields and environments with little modification, certain domains entail particular attention. For example, p2p retrieval presents particular challenges, first of all, because the majority of data that is available in p2p networks is in non-textual form and secondly because efficiency is of particular importance, so that the network can be scalable and grow to millions of users. The implementation of the proposed algorithms in this setting may provide valuable conclusions that may further aid in providing effective and realistic solutions to the problems of developing efficient and scalable p2p networks.

The evaluation of the approaches that have been put forth in this thesis has been exclusively focused on isolated testbeds and adhoc evaluation metrics. Although the experimental methodology that has been developed in the IR field aims to simulate real users and real information needs as best as possible, certain factors and phenomena, emerge only in realistic settings and cannot be studied in isolated environments. An example of such a phenomenon is the behavior of users during the time lapse between the query submission and the return of the final merged list by the DIR system. Experiments with real users could be designed and carried out in order to examine the best approach of presenting results in order to fill this time lapse. Potentially, the DIR system could return partial lists of results, automatically updating them as slower information sources return their results, in order to maintain the attention of the user.

Overall, although the field of DIR has already more than a decade of serious study and experimentation, a significant number of areas and applications remain largely unexplored. As already mentioned in section 1.3 the areas of Source Discovery and Result Page Parsing remain only minimally researched. Other potential areas of research, in addition to the ones mentioned above, include Graphical User Intefaces (GUI) design issues pertinent to DIR systems and their characteristics, personalization issues with regard to user preferences, past experiences and interests (potentially penalizing or rewarding information sources over others) etc. It was one of the aims of this thesis and of this section in particular, to highlight some of the potential directions of research for DIR systems.

It is the opinion of the author that the unprecedented increase of digital information nowadays and the inherit need to organize and retrieve this information in an effective and efficient manner can mainly be countered through DIR systems, regardless of the particular application and form (dark web retrieval, p2p networks etc) under which they may be utilized, therefore their study always remains current and imperative.

CHAPTER 8

Publications

The following publications have been made by the author during the study and writing of this dissertation:

- Paltoglou, G., Salampasis, M., Satratzemi, M. Simple adaptations of data fusion algorithms for source selection, In Proc. 31st European Conference on Information Retrieval, ECIR '09, to appear.

- Paltoglou, G., Salampasis, M., Satratzemi: Collection-integral Source Selection for uncooperative distributed information retrieval environments, Information Sciences, accepted.

- Paltoglou, G., Salampasis, M., Satratzemi, Modeling information sources as integrals for effective and efficient source selection, Information Processing and Management Journal, accepted.

- Paltoglou, G., Salampasis, M., Satratzemi, M. Integral Based Source Selection for Uncooperative Distributed Information Retrieval Environments, In Proc. Large-Scale Distributed Systems for Information Retrieval (LSDS-IR'08), p. 67 - 74.

- Paltoglou, G., Salampasis, F., Lazarinis , M. Indexing and Retrieval of a Greek Corpus, In Proc. Improving Non English Web Searching (iNEWS 2008), p. 47- 54.

- Paltoglou, G., Salampasis, M., Satratzemi, M.: A comparison of Centralized and Distributed Information Retrieval approaches, In Proc. 12th Panhellenic Conference on Informatics (2008), p. 21-25.

- Paltoglou, G., Salampasis, M., Satratzemi, M.: A results merging algorithm for distributed information retrieval environments that combines regression methodologies with a selective download phase, Information Processing and Management Journal 44(4): 1580-1599 (2008).
- Paltoglou, G., Salampasis, M., Satratzemi, M.: Hybrid Results Merging, In Proc. 16th Conference on Information and Knowledge Management, CIKM 2007, p. 321-331.
- Paltoglou, G., Salampasis, M., Satratzemi, M.: Results Merging Algorithm Using Multiple Regression Models, In Proc. 29th European Conference on Information Retrieval, ECIR 2007, p. 173-184.
- Paltoglou G., Salampasis, M., Satratzemi, M., Evangelidis, G.: Using linkage information to approximate the distribution of relevant documents in DIR, In Proc. 11th Panhellenic Conference on Informatics (2007),p. 234-244.

BIBLIOGRAPHY

Adler, R. and J. Stipins (2008). Improved flash indexing. Official Google Blog, published online: `http://googlewebmastercentral.blogspot.com/2008/06/improved-flash-indexing.html` .

Al-Maskari, A., M. Sanderson, and P. Clough (2007). The relationship between ir effectiveness measures and user satisfaction. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 773–774. ACM.

Allan, J., M. Connell, W. B. Croft, F. Feng, D. Fisher, and X. Li (2000). Inquery and trec-9. In *Proceedings of TREC-9*, pp. 551–577.

Alpert, J. and N. Hajaj (2008). We knew the web was big... Official Google Blog, published online: `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html` .

Álvarez, M., J. Raposo, A. Pan, F. Cacheda, F. Bellas, and V. Carneiro (2007). Deepbot: a focused crawler for accessing hidden web content. In *DEECS '07: Proceedings of the 3rd international workshop on Data enginering issues in E-commerce and services*, New York, NY, USA, pp. 18–25. ACM.

Ashish, N. and C. A. Knoblock (1997). Semi-automatic wrapper generation for internet information sources. In *COOPIS '97: Proceedings of the Second IFCIS International Conference on Cooperative Information Systems*, Washington, DC, USA, pp. 160–169. IEEE Computer Society.

Aslam, J. A. and M. Montague (2001). Models for metasearch. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 276–284. ACM.

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

Azzopardi, L., M. Baillie, and F. Crestani (2006). Adaptive query-based sampling for distributed ir. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 605–606. ACM.

Azzopardi, L. and V. Vinay (2008). Accessibility in information retrieval. In *ECIR '08: Proceedings of the 30th European Conference on Information Retrieval*, pp. 482–489.

Baeza-Yates, R. A. and B. A. Ribeiro-Neto (1999). *Modern Information Retrieval.* ACM Press / Addison-Wesley.

Bailey, P., N. Craswell, and D. Hawking (2000). Dark matter on the web. In *WWW-9 Poster Proceedings.*

Bailey, P., N. Craswell, and D. Hawking (2003). Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing and Management 39*(6), 853–871.

Baillie, M., L. Azzopardi, and F. Crestani (2006). An evaluation of resource description quality measures. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, New York, NY, USA, pp. 1110–1111. ACM.

Bar-Yossef, Z. and M. Gurevich (2006). Random sampling from a search engine's index. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, New York, NY, USA, pp. 367–376. ACM.

Bender, M., S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer (2005). Minerva: collaborative p2p search. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pp. 1263–1266. VLDB Endowment.

Bergman, M. K. (2001, September). The deep web: Surfacing hidden value. BrightPlanet, published online: `http://www.brightplanet.com/pdf/deepwebwhitepaper.pdf`.

Broder, A. (2002). A taxonomy of web search. *SIGIR Forum 36*(2), 3–10.

Callan, J. (2000). *Modern Information Retrieval*, Chapter Distributed information retrieval. Kluwer Academic Publishers.

Callan, J., J. Allan, C. L. A. Clarke, S. Dumais, D. A. Evans, M. Sanderson, and C. Zhai (2007). Meeting of the minds: an information retrieval research agenda. *SIGIR Forum 41*(2), 25–34.

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

Callan, J. P., W. B. Croft, and S. M. Harding (1992). The inquery retrieval system. In *DEXA '99: Proceedings of the Third International Conference on Database and Expert Systems Applications*, pp. 78–83.

Callan, J. P., Z. Lu, and W. B. Croft (1995). Searching distributed collections with inference networks. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 21–28. ACM.

Calmet, J., S. Jekutsch, and J. Schü (1997). A generic query-translation framework for a mediator architecture. In *ICDE '97: Proceedings of the Thirteenth International Conference on Data Engineering*, Washington, DC, USA, pp. 434–443. IEEE Computer Society.

Calvé, A. L. and J. Savoy (2000). Database merging strategy based on logistic regression. *Information Processing and Management 36*(3), 341–359.

Caverlee, J., L. Liu, and J. Bae (2006). Distributed query sampling: a quality-conscious approach. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 340–347. ACM.

Cho, J., H. Garcia-Molina, and L. Page (1998). Efficient crawling through url ordering. *Comput. Netw. ISDN Syst. 30*(1-7), 161–172.

Cope, J., N. Craswell, and D. Hawking (2003). Automated discovery of search interfaces on the web. In *ADC '03: Proceedings of the 14th Australasian database conference*, Darlinghurst, Australia, Australia, pp. 181–189. Australian Computer Society, Inc.

Coyle, K. (2006, November). Mass digitization of books. *The Journal of Academic Librarianship 32*(6), 641–645.

Craswell, N. (2000). *Methods for Distributed Information Retrieval.* Ph. D. thesis, ANU.

Craswell, N., D. Hawking, and P. Thistlewaite (1999). Merging results from isolated search engines. In *Proceedings of the 10th Australasian Database Conference*, Auckland, NZ, pp. 189–200. Springer-Verlag. `http://research.microsoft.com/users/nickcr/pubs/craswell_adc99.ps`.

Cutrell, E., D. Robbins, S. Dumais, and R. Sarin (2006). Fast, flexible filtering with phlat. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, New York, NY, USA, pp. 261–270. ACM Press.

Dumais, S., E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins (2003). Stuff i've seen: a system for personal information retrieval and re-use. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, New York, NY, USA, pp. 72–79. ACM.

Dumais, S. T. (1994). Latent semantic indexing (lsi) and trec-2. In D. K. Harman (Ed.), *The Second Text REtrieval Conference (TREC-2)*, pp. 105–115. National Institute of Standards and Technology.

Feldman, S. and C. Sherman (2003). The high cost of not finding information.

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

French, J. C., A. L. Powell, C. L. Viles, T. Emmitt, and K. J. Prey (1998). Evaluating database selection techniques: A testbed and experiment. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 121–129. ACM.

Fuhr, N. (1999). A decision-theoretic approach to database selection in networked ir. *ACM Transactions on Information Systems 17*(3), 229–249.

Gravano, L., K. Chang, H. Garcia-Molina, and A. Paepcke (1997). Starts: Stanford protocol proposal for internet retrieval and search. Technical report, Stanford, CA, USA.

Gravano, L., H. Garcia-Molina, and A. Tomasic (1994). The effectiveness of gloss for the text database discovery problem. In *SIGMOD Conference*, pp. 126–137.

Gravano, L., H. Garcia-Molina, and A. Tomasic (1999). Gloss: Text-source discovery over the internet. *ACM Transactions on Database Systems 24*(2), 229–264.

Gulli, A. and A. Signorini (2005). The indexable web is more than 11.5 billion pages. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, New York, NY, USA, pp. 902–903. ACM.

Haas, L. M., D. Kossmann, E. L. Wimmers, and J. Yang (1997). Optimizing queries across diverse data sources. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, San Francisco, CA, USA, pp. 276–285. Morgan Kaufmann Publishers Inc.

Harman, D. (1993). Overview of the first trec conference. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 36–47. ACM.

Harman, D. K. (1995). Overview of the fourth text retrieval conference (trec-4). In D. K. Harman (Ed.), *The Third Text REtrieval Conference (TREC-4)*.

Hawking, D. (2004). Challenges in enterprise search. In *ADC '04: Proceedings of the 15th Australasian database conference*, Darlinghurst, Australia, Australia, pp. 15–24. Australian Computer Society, Inc.

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

Hawking, D. and P. Thomas (2005). Server selection methods in hybrid portal search. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 75–82. ACM.

Hayes, B. (2008). Cloud computing. *Commun. ACM 51*(7), 9–11.

He, B., M. Patel, Z. Zhang, and K. C.-C. Chang (2007). Accessing the deep web: A survey. *Communications of the ACM*.

Jansen, B. J., A. Spink, and T. Saracevic (2000). Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management 36*(2), 207–227.

Jelinek, F. and R. Mercer (1985). Probability distribution estimation from sparse data. Technical report.

Lee, J. H. (1997). Analyses of multiple evidence combination. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 267–276. ACM.

Lew, A. and H. Mauch (2006). *Dynamic Programming: A Computational Tool.* Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Liu, K.-L., W. Meng, J. Qiu, C. Yu, V. Raghavan, Z. Wu, Y. Lu, H. He, and H. Zhao (2007). Allinonenews: development and evaluation of a large-scale news metasearch engine. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, New York, NY, USA, pp. 1017–1028. ACM.

Liu, K.-L., A. Santoso, C. Yu, and W. Meng (2001). Discovering the representative of a search engine. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, New York, NY, USA, pp. 577–579. ACM.

Lu, J. and J. Callan (2003). Content-based retrieval in hybrid peer-to-peer networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, New York, NY, USA, pp. 199–206. ACM.

Lu, Y., W. Meng, L. Shu, C. Yu, and K. lup Liu (2005). Evaluation of result merging strategies for metasearch engines. In *In Proc. WISE 2005: Web Information Systems Engineering*, pp. 53–66.

Lyman, P. and H. R. Varian (2003). How much information? 2003. SIMS University of California at Berkeley, published online: `http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/internet.htm`.

Macdonald, C. and I. Ounis (2006). Voting for candidates: adapting data fusion techniques for an expert search task. In *CIKM '06: Proceedings of the 15th international conference on Information and knowledge management*, New York, NY, USA, pp. 387–396. ACM.

Macdonald, C., I. Ounis, and I. Soboroff (2007). Overview of the trec-2007 blog track. In *The Sixteenth Text REtrieval Conference (TREC 2007)*.

Manning, C. D., P. Raghavan, and H. Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Miller, J. (2007). Most fed data is un-googleable. Federal Computer Week, published online: `http://www.fcw.com/online/news/151098-1.html?CMP=OTC-RSS`.

Najork, M. and J. L. Wiener (2001). Breadth-first crawling yields high-quality pages. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, New York, NY, USA, pp. 114–118. ACM.

Nottelmann, H. and N. Fuhr (2003a). Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 290–297. ACM.

Nottelmann, H. and N. Fuhr (2003b). From uncertain inference to probability of relevance for advanced ir applications. In *ECIR '03: Proceedings of the 25th European Conference on Information Retrieval*, pp. 235–250.

Nottelmann, H. and N. Fuhr (2004). Combining cori and the decision-theoretic approach for advanced resource selection. In *ECIR '04: Proceedings of the 26th European Conference on Information Retrieval*, pp. 138–153.

Ntoulas, A., P. Zerfos, and J. Cho (2005). Downloading textual hidden web content through keyword queries. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, New York, NY, USA, pp. 100–109. ACM.

Ogilvie, P. and J. Callan (2003). Combining document representations for known-item search. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, New York, NY, USA, pp. 143–150. ACM.

Ogilvie, P. and J. P. Callan (2001). Experiments using the lemur toolkit. In *The Third Text REtrieval Conference (TREC-10)*, pp. 103–108.

Page, L., S. Brin, R. Motwani, and T. Winograd (1998). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.

Powell, A. L. and J. C. French (2003). Comparing the performance of collection selection algorithms. *ACM Transactions on Information Systems 21*(4), 412–456.

Powell, A. L., J. C. French, J. Callan, M. Connell, and C. L. Viles (2000). The impact of database selection on distributed searching. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 232–239. ACM.

Raghavan, S. and H. Garcia-Molina (2001). Crawling the hidden web. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, San Francisco, CA, USA, pp. 129–138. Morgan Kaufmann Publishers Inc.

Rasolofo, Y., F. Abbaci, and J. Savoy (2001). Approaches to collection selection and results merging for distributed information retrieval. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, New York, NY, USA, pp. 191–198. ACM.

Rasolofo, Y., D. Hawking, and J. Savoy (2003). Result merging strategies for a current news metasearcher. *Inf. Process. Manage. 39*(4), 581–609.

Rijsbergen, C. J. V. (1979). *Information Retrieval.* Newton, MA, USA: Butterworth-Heinemann.

Robertson, S., S. Walker, H.-B. M., and G. M. (1994). Okapi at trec-3. In D. K. Harman (Ed.), *The Third Text REtrieval Conference (TREC-3)*, pp. 500–225. National Institute of Standards and Technology.

Rose, D. E. and D. Levinson (2004). Understanding user goals in web search. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, New York, NY, USA, pp. 13–19. ACM.

Seo, J. and B. Croft (2009). Blog site search using resource selection. In *CIKM '08: Proceedings of ACM 17th Conference on Information and Knowledge Management*, New York, NY, USA. ACM.

Shokouhi, M. (2007). Central-rank-based collection selection in uncooperative distributed information retrieval. In *ECIR '07: Proceedings of the 29th European Conference on Information Retrieval*, pp. 160–172.

Shokouhi, M., M. Baillie, and L. Azzopardi (2007). Updating collection representations for federated search. In anonymous (Ed.), *SIRIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 511–518. ACM Press.

Shokouhi, M., F. Scholer, and J. Zobel (2006, January). Sample sizes for query probing in uncooperative distributed information retrieval. In X. Zhao, J. Li, H. Shen, M. Kitsuregawa, and Y. Zhang (Eds.), *Proceedings of the APWeb Asia Pacific Web Conference*, Harbin, China, pp. 63–75. LNCS 3841.

Shokouhi, M., J. Zobel, F. Scholer, and S. M. M. Tahaghoghi (2006). Capturing collection size for distributed non-cooperative retrieval. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 316–323. ACM.

Si, L. and J. Callan (2002). Using sampled data and regression to merge search engine results. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 19–26. ACM.

Si, L. and J. Callan (2003a). Relevant document distribution estimation method for resource selection. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, New York, NY, USA, pp. 298–305. ACM.

Si, L. and J. Callan (2003b). A semisupervised learning method to merge search engine results. *ACM Transactions on Information Systems 21*(4), 457–491.

Si, L. and J. Callan (2004). Unified utility maximization framework for resource selection. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, New York, NY, USA, pp. 32–41. ACM.

Si, L. and J. Callan (2005). Modeling search engine effectiveness for federated search. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 83–90. ACM.

Si, L., R. Jin, J. Callan, and P. Ogilvie (2002). A language modeling framework for resource selection and results merging. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, New York, NY, USA, pp. 391–397. ACM.

Sparck Jones, K. and C. J. Van Rijsbergen (1975). Report on the need for and provision of an 'ideal' information retrieval test collection.

Suel, T., C. Mathur, J. wen Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram (2003). Odissea: A peer-to-peer architecture for scalable web search and information retrieval. In *WebDB*, pp. 67–72.

Sutherland, W. J. (1996). Basic techniques. In *Ecological census techniques: a handbook*, pp. 11–110. Cambridge University Press.

Thomas, P. and D. Hawking (2007). Evaluating sampling methods for uncooperative collections. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 503–510. ACM.

Voorhees, E. M. (1998). Variations in relevance judgments and the measurement of retrieval effectiveness. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 315–323. ACM.

Voorhees, E. M., N. K. Gupta, and B. Johnson-Laird (1995). Learning collection fusion strategies. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 172–179. ACM.

Voorhees, E. M., N. K. Gupta, and B. J. Laird (1994). The collection fusion problem. In D. K. Harman (Ed.), *The Third Text REtrieval Conference (TREC-3)*, pp. 500–225. National Institute of Standards and Technology.

Wang, L., J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl (2008). Scientific cloud computing: Early definition and experience. In *HPCC '08: Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*, Washington, DC, USA, pp. 825–830. IEEE Computer Society.

Waterhouse, S., D. M. Doolin, G. Kan, and Y. Faybishenko (2002). Distributed search in p2p networks. *IEEE Internet Computing 6*(1), 68–72.

Wu, Z., W. Meng, C. Yu, and Z. Li (2001). Towards a highly-scalable and effective metasearch engine. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, New York, NY, USA, pp. 386–395. ACM.

Xu, J. and J. Callan (1998). Effective retrieval with distributed collections. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 112–120. ACM.

Xu, J. and W. B. Croft (1999). Cluster-based language models for distributed retrieval. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 254–261. ACM.

Yager, R. R. and A. Rybalov (1998). On the fusion of documents from multiple collection information retrieval systems. *Journal of the American Society for Information Science and Technology 49*(13), 1177–1184.

Yuwono, B. and D. L. Lee (1997). Server ranking for distributed text retrieval systems on the internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pp. 41–50. World Scientific Press.

Zhai, C. and J. Lafferty (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 334–342. ACM.

Zhao, H., W. Meng, Z. Wu, V. Raghavan, and C. Yu (2005). Fully automatic wrapper generation for search engines. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, New York, NY, USA, pp. 66–75. ACM.

Zhu, Y. and Y. Hu (2007). Efficient semantic search on dht overlays. *J. Parallel Distrib. Comput. 67*(5), 604–616.

Zobel, J. (1998). How reliable are the results of large-scale information retrieval experiments? In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 307–314. ACM.