Distributed Resource Discovery Systems

BY

Lei Iat Seng

A Thesis Presented to
The Hong Kong University of Science and Technology
In Partial Fulfillment
of the Requirements for
the Degree of Master of Philosophy
in Computer Science

Hong Kong, Jan 1998

Copyright © by Lei Iat Seng 1998
Authorization Page

I hereby declare that I am the sole author of the thesis.

I authorize The Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Le Jet Song

I further authorize The Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Le Jet Song
Distributed Resource Discovery Systems

BY

Lei Iat Seng

APPROVED:

[Signature]
DR. DIK L. LEE, SUPERVISOR

[Signature]
PROF. ROLAND T. CHIN, HEAD OF DEPARTMENT

Department of Computer Science

24 Jan 1998
I would like to express my gratitude to my supervisor Dr. Dik L. Lee for his guidance in supervising this research. I also want to thank Dr. Fangzhen Lin, Dr. Nevin L. Zhang and Dr. Kamalakar Karlapalem for being my defense committee members.

This thesis would not exist without the assistance and encouragements from my friends. Ricci Leong helps me in all aspects, from proof reading to supplying computing resources and snacks. Nelson Chu gives me encouragements and hints to happiness in my most depressed days. Hours of online chats with Fabian Ho not only provide valuable breaks in my working time, but also untie many knots in my mind.

And I will not forget the warm family feeling from my flatmates. Eric Lam, Kenneth Lau, Chow To and Joe Law have shared many TV time with me. Without their laughters and delicious meal, I would not recover soon enough to finish my thesis.

I also want to thank the God that I still don’t believe. Several coincidences that saved me from giving up seem to be His careful arrangement.

LEI IAT SENG

The Hong Kong University of Science and Technology
Jan 1998
Contents

Title Page i
Authorization Page ii
Signature Page iii
Acknowledgments iv
Table of Contents v
List of Figures viii
List of Tables x
Abstract xi

Chapter 1 Introduction 1
Chapter 2 Related work 5
Chapter 3 An abstract DRD model 10
  3.1 Agents .................................................. 11
  3.2 Indexing information .................................. 12
  3.3 Interaction ............................................. 14
Chapter 4 Microscopic view – Collection ranking 17
  4.1 An ideal goodness score .............................. 18
4.2 Evaluating collection ranking methods ......................................... 20
4.3 Experiment datasets ............................................................... 21
4.4 Comparing existing methods ..................................................... 24
4.5 Observation and Improvement .................................................. 29
4.6 Chapter summary ................................................................. 32

Chapter 5  Macroscopic view – Architectures and Interoperation  40

5.1 Hierarchical architecture .......................................................... 41
  5.1.1 MISA tree as search tree .................................................. 42
  5.1.2 MISA tree as a hierarchical partitioning of collections ............ 46
5.2 ARD merging ................................................................. 48
  5.2.1 Max-based ARD merging ............................................... 49
  5.2.2 Sum-based ARD merging ............................................... 55
5.3 Building a MISA tree ............................................................. 57
  5.3.1 Large branch-out ......................................................... 57
  5.3.2 The dilution problem ..................................................... 58
  5.3.3 Topic specific MISA ..................................................... 59
5.4 Flat architecture ................................................................. 60
5.5 Weaving a CISA mesh ............................................................ 64
  5.5.1 ARD coherency ............................................................ 64
  5.5.2 An ideal weaving scheme .............................................. 66
  5.5.3 Approximating the ideal weaving scheme ........................... 68
5.6 Hybrid architectures ............................................................ 70
  5.6.1 Comparing and reconciling the hierarchical and flat architectures ......................................................... 70
  5.6.2 Hierarchical architecture with overlaps .............................. 74
  5.6.3 Flat architecture with ARD propagation .............................. 78
5.7 Interoperation ................................................................. 81
5.8 Other issues ................................................................. 84
  5.8.1 Decentralizing ISA navigation ....................................... 84
Chapter 6  Conclusion  86

Bibliography  89
List of Figures

1.1 Centralized index server ........................................ 2
1.2 Centralized meta-indexer ......................................... 3

2.1 A hierarchy of Whois++ servers ............................... 6
2.2 Centroids in Whois++ ........................................... 7
2.3 Architecture of Ingrid ........................................... 8

3.1 Overview of agents in a DRD system .......................... 16

4.1 A typical result list from collection $C_i$ ..................... 30
4.2 Evaluation of existing collection ranking methods. (Dataset 2) . 34
4.3 Evaluation of new and improved collection methods (Dataset 3,
        $P_a = 0.8$) ..................................................... 35
4.4 Evaluation of $G_i(cvu), G_i(eidf), G_i(gidf), G_i(gloss')$ and $G_i(weidf)$
        (Dataset 1) ..................................................... 36
4.5 Evaluation of $G_i(cvu), G_i(eidf), G_i(gidf), G_i(gloss')$ and $G_i(weidf)$
        (Dataset 2) ..................................................... 37
4.6 Evaluation of $G_i(cvu), G_i(eidf), G_i(gidf), G_i(gloss')$ and $G_i(weidf)$
        (Dataset 3) ..................................................... 38
4.7 Evaluation of collection methods in the split case (Dataset 3) . 39
4.8 Evaluation of $G_i(cvu), G_i(eidf), G_i(gidf), G_i(gloss')$ and $G_i(weidf)$
        (Dataset 4) ..................................................... 39
5.1 The pure hierarchical architecture: MISA tree on top of primary ISAs ........................................ 43
5.2 Navigation algorithm for MISA tree: collection selection parameter ............................... 45
5.3 Navigation algorithm for MISA tree: general collection selection parameter ......................... 46
5.4 Virtual documents in $G_a(gloss, s_{\text{min}})$ ................................................................. 54
5.5 The pure flat architecture: CISA mesh and CISAs ......................................................... 62
5.6 Navigation algorithm for pure CISA mesh ................................................................. 63
5.7 Patches of high goodness score CISAs separated by low score ridges .................................. 64
5.8 A group of coherent CISAs with descending goodness scores ........................................ 67
5.9 Revised navigation algorithm for pure CISA mesh ....................................................... 69
5.10 Query processing at an abstract ISA ................................................................. 74
5.11 Navigation algorithm for abstract ISA mesh .............................................................. 75
5.12 The extended hierarchical architecture: acyclic mesh with overlaps ................................ 76
5.13 A cubic CISA mesh ................................................................................................. 81
5.14 MISA on CISA mesh on MISA trees ........................................................................ 82
5.15 General query processing algorithm for the IUA to interact with various kinds of abstract ISAs ................................................................. 83
List of Tables

4.1 The four base datasets used in the evaluation of collection ranking methods .............................................. 22
4.2 Some statistics of the Smart test collections ......................................................................................... 22
4.3 Some statistics of dataset 3 (disjoint collections from the Reuters-21578 test collection) .................... 23
4.4 Some statistics of dataset 4 (overlapping collections from the Reuters-21578 test collection) ............. 24
4.5 Summary of some existing collection ranking methods ................................................................. 28
4.6 Summary of the newly introduced collection ranking methods .................................................... 32

5.1 An example comparing ARD merging functions for $G_4(gloss)$ .................................................. 50
5.2 Comparison of pure hierarchical and flat architectures ............................................................... 72
Distributed Resource Discovery Systems

BY

Lei Iat Seng

A Thesis Presented to
The Hong Kong University of Science and Technology
In Partial Fulfillment
of the Requirements for
the Degree of Master of Philosophy
in Computer Science

Hong Kong, Jan 1998

Abstract

Large-scale network of information sources and digital libraries amplifies the information overload problem constantly. A popular resource discovery tool on the World Wide Web nowadays is centralized index server, which attempts to be a 'know-it-all' of online resources. It suffers from many shortcomings resulted from the centralization of metadata extraction, index building and query processing. A better approach is to use disparate index servers for individual resource repository. A user query is only routed to a site if the repository is likely to contain relevant resources. This task is often called collection selection, and the
server to perform this is usually called meta-indexer. Nevertheless, most research projects assume a centralized meta-indexer which will eventually be degraded by centralization.

In this work, we look into issues of distributing the collection selection subsystem itself. An abstract model for distributed resource discovery systems is introduced as a common model to describe different architectures. We then study some algorithms in collection ranking used in a centralized meta-indexer, and show how they can be scaled up for distributed architectures. Two basic architectures, the hierarchical and flat architectures, are first considered. The features of these architectures are examined and mixed to form two new hybrid architectures. Finally, we consider the interoperability of the various architectures, each of which is suitable for different scenarios.
Chapter 1

Introduction

In this information age, we are facing the overwhelming problem of information overload with the advent of large scale networked information systems and digital libraries. The rapid growth in the quantity, the vast heterogeneity, and the volatility of resources available online post great challenges to designers of resource discovery systems.

A popular resource discovery tool on the World Wide Web [6] nowadays is centralized index servers, commonly known as search engines. See Fig. 1.1. Web pages are retrieved in full source form to the index server, where metadata is extracted to build an index. Queries are submitted and processed there. Resource metadata extraction, index building, and query processing are all done in the central site. One may improve the system to consume less bandwidth and handle resource heterogeneity better by decoupling the metadata extraction to a site close to the information providers.

However, the centralized resource discovery system with a single index server has several important shortcomings. Centralization induces scalability problem and demands extravagant computing resources and bandwidth. Network traffic from both index building and query processing congregates at the central index server, which also becomes a single point of failure. Moreover, some information providers may prefer to keep their own indices. This gives rise to a number of
disparate index servers. What is desired is a collection selection subsystem that helps the user to decide which index servers have documents relevant to the user query. A user can then first post his query to the server selection subsystem, which recommends a list of promising index servers. The user only needs to route the query to these index servers. The introduction of a collection selection subsystem and the breakdown of a huge index into disparate indices is an important step in decentralizing resource discovery system.

A popular approach to the collection selection subsystem is a meta-indexer which extracts some metadata from the disparate index servers. See Fig. 1.2. In a sense, a meta-indexer indexes the ordinary index servers. It can estimate the relevancy of documents in the index servers and help the user to route a query to promising ones only. Many literatures on distributed resource discovery systems assume this single meta-indexer architecture. This architecture, however, reiterates the problem of centralization with a single meta-index. Although scalability problem becomes less harsh, the single meta-indexer still presents a bottleneck of query processing and enforces a central management.

In general, a distributed resource discovery (DRD) system should satisfy the following requirements:

- **scalability**: The DRD system has to scale up well as the quantity of online resources, information providers and information consumers increases constantly.
- *site autonomy*: Some information providers would like to maintain autonomy of their resource repositories and not allow external agents to index their resources directly. Only the index servers of the information providers themselves have full metadata about their resources. A query with relevant resources in such an index server has to be processed there.

- *topical clustering*: Not all resources are of same importance to an user. Both information sources and needs form topical and geographical clustering. Users from a community may be more interested in certain topics and topics with a lot of user call for specialized service and dedicated computing resources. Leveraging this clustering phenomenon helps to allocate more computing resources to popular information needs.

- *resilience*: The system should support replication of service and decentralization. The failure of a single site should not paralyze the whole system.

We propose a distributed collection selection subsystem in a DRD system. In this research, we look into the architectures and algorithms for vector-based model in such DRD system. After reviewing some related work in Chapter 2, we propose an abstract DRD model in Chapter 3. This model serves as a common ground for discussing different architectures. Then, an essential microscopic function of the system, collection ranking, is examined in Chapter 4. Finally, we adapt the
collection ranking methods to distributed collection selection subsystems using different architectures. This is discussed in detail in the macroscopic issues in Chapter 5. We give a conclusion and list some future work in Chapter 6.
Chapter 2

Related work

There are numerous distributed resource discovery systems proposed in the literature and in use nowadays. Many of them apply centralized collection selection. For boolean-based retrieval model, two main architectures can be found in the literature for distributed collection selection.

The centralized server for collection selection is usually called broker or meta-indexer. Each index server uploads to it some metadata describing its own collection of resources. The meta-indexer can then select promising index servers for a user query by collection ranking. Collection ranking method differs for different retrieval models. Examples for the vector-based retrieval model include CVV [35], CORI [12], and gGLOSS [19]. More information of them are given in Chapter 4. Works like CAFE [13], MeDoc [7], ALIWEB [24] and Indie [14] focus on other aspects like collection selection based on non-functional parameters (e.g. cost), heterogeneity of schema and interface of individual index servers, and consistency maintenance of the meta-index. Harvest [8] applies a hierarchical architecture for gathering resource metadata. However, it is still centralized in collection selection.

Whois++ [28, 10] proposes a hierarchical architecture for distributed collection selection for boolean-based retrieval model. The resource description record in Whois++ is a list of attribute-value pairs. A centroid of the same format is
used to represent a collection. The value of an attribute in the centroid is equal to the union of those of the resource records in the collection. These centroids are transferred between Whois++ servers in a hierarchy. A Whois++ server can either upload the centroids received intact, or merge them into a single centroid using similar union operation and upload the result only. Fig. 2.2 is an example of the resource description record and centroid in a simple hierarchy (called Whois++ mesh) in Fig. 2.1. The query in Whois++ is formulated as attribute-value pairs. It matches a record if all the values in an attribute appear in the corresponding attribute of the record. For instance, the query "hobby: music" matches both records in server A. The user agent navigates the Whois++ mesh [15] by matching the query with the centroids. In our example, the query matches the centroid of server C, and so the query is routed there. Then the user agent finds that the query only matches the centroid of server A. The query is therefore routed to server A, but not server B. Whois++ can be considered as a special case of our model which supports vector-based model.

![Diagram](image)

**Figure 2.1: A hierarchy of Whois++ servers**

An alternative to hierarchical architecture and metadata merging is the flat architecture. Servers holding similar collections are linked together. When the user finds that his query is relevant to a collection, he is likely to find other relevant collections nearby. This concept is used for boolean-based retrieval model in Ingrid [17]. Each node in Ingrid is characterized by a set of index terms. Nodes featuring the same terms are linked together to form a term cluster. Fig. 2.3 shows three term clusters. Nodes that appear in more than one clusters serve
Two resource description records at A
name: philip
hobby: reading programming music

name: tom
hobby: music swimming

Centroid at A
name: philip tom
hobby: reading programming music swimming

Centroid at B
name: peter mary
hobby: reading football

Merged centroid at C
name: philip tom peter mary
hobby: reading programming music swimming football

Figure 2.2: Centroids in Whois++

as bridges between them. Consider the query “database scheduling distributed-
system”. Query processing starts at a node in a certain cluster suggested by a
special server (not shown). The query is then routed to nodes that appear in
more and more clusters until it reaches the one that belongs to all three clusters.

NetAgent [29] presents a different interpretation of a network of nodes. Each
node, called an agent, is labeled with some topics or interest of a community
of users. Connection between agents (called association) is suggested by users.
Users search information by browsing the network like hypertext. At the same
time, the network is trained by changing the weight of an association according
to users' feedback. This connectionist approach is not explored in our current
work.

Navigating a mesh of inter-related nodes is also analogous to client-side re-
source discovery systems in the World Wide Web. InfoSpider [26] and FishSearch
[9] search for more relevant documents along the hyperlinks from a starting doc-
Figure 2.3: Architecture of Ingrid

ument. On the other hand, some document ranking algorithms for hypertext like [34] propagate metadata along hyperlinks.

The probabilistic model in [4] is a theoretical approach to probabilistic distributed collection ranking. It is currently being implemented in Dsmily [3], a distributed system for multimedia information retrieval.

Distributed directory service is sometimes also viewed as a resource discovery system. The query in this case is a name in the namespace of the directory. Familiar examples are DNS and X.500. In addition to name lookup, some directory services, like LDAP [32] for X.500, support a search function limited to the records within one server. Incidentally, we believe that it is desirable to integrate distributed directory service and hierarchical architecture in distributed collection selection.

Finally, there are work in progress in standardization of data structure and protocol at the implementation level. RDM [21] (Resource Description Message) is an elaboration of the SOIF format used in Harvest. It defines a text format for a list of attribute-value pairs typed by a schema. RDF (Resource Description Framework) exemplifies current work in metadata format and model for WWW. More information is available at http://www.w3.org/Metadata/.
URN (Universal Resource Name), a successor of URL [5], is another work in progress in IETF (http://www.ietf.org/html.charters/urn-charter.html) on the design of resource locator. Common Indexing Protocol, CIP, [1] extracts the centroid transfer protocol from Whois++. More information is available at the 'find' Charter of IETF at http://www.ietf.org/html.charters/find-charter.html. STARTS [18] is intended to be a standard client/server protocol between the user agent and the index server in query processing. It defines standard format of query and response. KQML [16] is a more general language for inter-agent communication. In addition to the simple client-server protocol in STARTS, it also supports the notion of brokerage directly and models the interaction with meta-indexer better.

There are much previous work in search tree. The abstraction of generalized search tree in GiST [22, 2] gives us valuable insights in our approach of structuring a hierarchy of agents as a search tree.
Chapter 3

An abstract DRD model

One may view a distributed resource discovery system (DRD) system as a matchmaking blackbox between information consumers and information providers. An information consumer, also simply known as user, specifies his information needs in a query. An information provider offers metadata of the resources that it owns. The DRD system tries to determine the relevancy of a resource by comparing this metadata and the query. We will limit the interaction between the user and the DRD system to a stateless client server paradigm which is employed in most information retrieval systems. The user submits a query to the system and obtains a response containing a list of relevant resources. In particular, the alternative paradigm of information filtering, where information needs are specified as user profiles and new incoming documents are dispatched to potential interested users, is not considered.

In this chapter, we present an abstract model of DRD system. The model is designed to be a high level abstraction, with the aim that both traditional and our newly proposed architectures can be described and interoperate under a unified model. Because of the lack of standard terminology, we will also introduce our own and borrow some contemporary terms where appropriate. The processing components in the model are introduced in Section 3.1, followed by a description of the information exchanged in Section 3.2 and their interaction in Section 3.3.
3.1 Agents

Numerous interconnected components reveal themselves as we open up the black-box. We call the component through which an information consumer interacts with the system an *Index User Agent* (IUA). The main functionality of an IUA is to provide a single service access point for the user to submit information need. It interprets and translates the user query into internal query format of the DRD system, and communicates with the ‘guts’ of the DRD system properly to return a list of relevant resources. Hence, IUA shields the user from the complexity of the internal structure of the DRD system. We will focus on this functionality of IUA. Other capabilities like learning and personalization are beyond the scope of this thesis.

At the information provider end sits a set of *Index Extraction Agents* (IEAs). IEAs extract metadata termed *resource description* (RD) from the providers’ resource repositories. The RD of a resource has two main parts: *resource locator*, a handle in the resource’s native access mechanism through which the user can access the resource, and *resource summary*, metadata of the resource for determining relevancy. Metadata extraction can be complex by itself, as the necessary metadata may not be (explicitly) stored in the resource. It then requires a human expert or an intelligent agent to explore the semantics of the resource. The IEA encapsulates this task. Notice that the IEAs are the only contact points between the DRD system and the information providers. Resource retrieval is done by native resource access mechanism external to the DRD system. Incidentally, the access mechanism may be embedded into the IUA for user convenience.

Both the query in internal format and the resource description are fed into a fabric of processing components between the IUAs near the information consumers and the IEAs near the information providers. We call them *Index System Agents* (ISAs) generically and refer to a set of ISAs connected in a certain topology a mesh. The ISAs form the ‘guts’ of the DRD system and cooperatively perform the matching process. They work with the query in internal format,
resource descriptions and other metadata derived from them and are insulated from the format and access mechanism of the resources.

3.2 Indexing information

In this section, we look more closely at the ISAs in a DRD system. We peek into what information are transferred between them to understand their 'knowledge' and 'capability'.

We call two connected ISAs acquaintances. The acquaintance relation is a long-term agreement between the ISAs to cooperate for query processing. In preparation to process upcoming user queries, the ISAs transfer indexing information between them. These information are usually resource descriptions or their derivatives and are crucial for the effective cooperation of the ISAs. In terms of indexing information transfer, the acquaintance relation is unidirectional. We will say that a downstream ISA uploads indexing information to an upstream ISA. Denote the set of downstream acquaintance of an ISA $A$ as $Acq(A)$.

We now consider the kinds of indexing information that should be transferred between $A$ and $B \in Acq(A)$. An analogy in the research community sheds some light on these. Suppose a researcher wants to find literature on the subject of distributed resource discovery system. He himself may be very knowledgeable in the subject and has bibliographic references of relevant works at his disposal. These knowledge are acquired from others beforehand through various means like books, online search, and research meetings. On the other hand, he may be aware of his ignorance in the subject, but knows the names of colleagues from whom he may ask for recommendation on literature or other researchers who might know the answers.

In a DRD system, the first kind corresponds to resource description RD. Since ISA does not extract RD himself, he must either obtain them from IEAs, which perform RD extraction, or from other ISAs, which relay to him the RDs in his 'knowledge'. The RD uploading interface of IEA and ISA are assumed to be the
same for convenience.

The second kind of indexing information is a lossy compression of a set of resource descriptions owned by some other agents. We call this *aggregate resource description*, or ARD for short. ARD has similar structure as RD, consisting of *collection metadata* and *collection locator*. Collection metadata summarizes a set of resources in a collection. It gives some hints of how relevant the resources in the collection are to a query. The collection locator states which ISA $\mathcal{A}$ ‘speaks for this collection’ and how an interested agent should interact with him. (The notion of putting together ‘who’ and ‘how to interact’ can be found in URL.) Since the aim to consult $\mathcal{A}$ is to process a query, we refer to this action as *query routing*. The collection summarized by an ARD is often decentralized and owned by several ISAs. Therefore, $\mathcal{A}$ may in turn suggest the agent to route the query to other ISAs. Conceptually, an IUA needs the appropriate protocol handler to process a collection locator.

We call the set of RDs or ARDs received and maintained by an ISA his *RD base*. For simplicity, we restrict one’s RD base to either all RDs or ARDs. (In the architectures considered in this work, two separate pure RD bases serve the purpose of a mixed RD base.) Hence there are two kinds of ISAs. *Primary ISA* (PISA for short) receives resource description from his acquaintance. He can answer a query without query routing. The role of query processing through a prepared index of RD is often known as index server in the literature. Primary ISAs in this work are assumed to have this capability if not stated otherwise. The role of an ISA with RD base of ARDs varies in different architectures. We will simply refer to them as *abstract ISAs*. An abstract ISA basically cannot ‘speak for’ the individual resource in the collections summarized by the ARDs in his RD base.

A primary ISA keeps RD in his RD base. He can upload the RDs intact. Alternatively, he can summarize his RD base by one or more ARDs and upload the ARDs instead. This is known as RD aggregation. We will mainly use a single merged ARD. There are several reasons that ARD is preferred over intact RD
upload. ARD is smaller in size. The transfer of the entire set of RD may incur too much traffic due to the huge size and frequent updates. \( B \in \text{Acq}(A) \) may view his RD base proprietary and is not willing to give it to \( A \). Lastly, \( A \) may not be willing to accept the whole RD base because of, say, computing resource limitation.

An abstract ISA keeps ARDs in his RD base and can only upload ARD. Nonetheless, he has the options of whether and how to merge the multiple ARDs into one or more ARDs. We will consider only merging into a single ARD. Merging of multiple ARDs \( ard_i \)'s into a single ARD \( ard_m \) is often defined such that \( ard_m \) somehow summarizes the individual \( ard_i \)'s. The detail, however, depends on the architecture of the abstract ISA mesh.

In the case where a primary ISA uploads RDs or an abstract ISA uploads ARDs without merging, their upstream acquaintance does not need their help in query processing. This role of ISA is known as mediator. A mediator collects RDs or ARDs, performs some value-added operations like filtering or clustering, and uploads the processed RDs or ARDs. Issues on mediators are not considered in detail in this thesis. In most cases, we will assume a primary ISA or abstract ISA upload a single ARD.

We describe the centralized search engine and meta-indexer in the introduction here again as an example of the application of our terminology. The centralized search engine functions as a conglomerate of a single primary ISA and IEs which can be decoupled. The centralized meta-indexer is a single abstract ISA \( M \) on top of a set of primary ISAs \( P_j \)'s. Each \( P_j \in \text{Acq}(M) \) uploads an aggregate resource description of his RD base to \( M \).

### 3.3 Interaction

We now give an overview of the agents, indexing information transferred and their interaction in the abstract DRD model. See Fig. 3.1.

Index extraction agents \( E_k \)'s extract resource descriptions (RDs) from resource
repositories and submit them to primary ISAs $\mathcal{P}_j$'s. These $\mathcal{P}_j$'s make an index from the RDs received and are capable of estimating the relevancy of these resources to a query. They also upload an aggregation of their RD base to some abstract ISAs $\mathcal{A}_i$ as aggregate resource descriptions (ARDs). In preparation of query processing, $\mathcal{A}_i$'s exchange ARDs among themselves. Update versions of the RDs and ARDs are periodically transferred. Primary ISAs $\mathcal{P}_j$'s are the only kind of agents that can return a list of locators to relevant resources. Abstract ISAs only have ARDs and must rely on primary ISAs for the final resource ranking. When a user query is submitted to an $\mathcal{A}_i$, query routing must be performed.

Query processing starts when an information consumer submits a query to the index user agent $\mathcal{U}$. $\mathcal{U}$ translates the query to the internal query format and in turn submits it to the ISA mesh. Conceptually, we restrict all the interactions to take place between $\mathcal{U}$ and the ISAs ($\mathcal{A}_i$'s and $\mathcal{P}_j$'s) to a request-response protocol. $\mathcal{U}$ is responsible to keeping the intermediate states and results of query processing. An ISA does not route a query to another ISA himself, but only suggests the routing to $\mathcal{U}$. Because of this simplification, query processing proceeds in the form of consecutive consultation of various ISAs from $\mathcal{U}$. This consists of three main steps. First, $\mathcal{U}$ interacts with the abstract ISA mesh using suitable protocol. He cooperates with $\mathcal{A}_i$'s to select a set of $\mathcal{P}_j$'s for further query processing. Then, $\mathcal{U}$ submits the query to each of the selected $\mathcal{P}_j$'s, which returns a ranked list of locators to relevant resources. Finally, $\mathcal{U}$ merges or combines the separate results into a single list and presents it to the information consumer. $\mathcal{U}$ may need more information from $\mathcal{P}_j$'s to merge the results correctly.

The first step, selecting primary ISAs $\mathcal{P}_j$'s, is called collection selection or server selection in some literature. A microscopic part of this is rating several collections for their appropriateness for query routing. This is called collection ranking and is the subject of Chapter 4. The topology of the abstract ISA mesh, the indexing information transferred among them, and their interaction with the IUA vary according to the architectures used. These will be discussed in detail in Chapter 5.
Figure 3.1: Overview of agents in a DRD system
Chapter 4

Microscopic view – Collection ranking

In the abstract DRD model, aggregate resource description (ARD) is used to represent a collection, i.e. a set of resources, in a primary ISA $\mathcal{P}_i$. Given the ARD and the user query, an agent should be able to make an educated guess of the relevancy of the resources in the collection, and decide whether to route the query to $\mathcal{P}_i$ or not. We will follow the usual approach of using a numerical score, called goodness score $G_i$, as the metric for the 'relevancy of collection'. The process of computing goodness score is called collection ranking. Goodness score is the basic information used in collection selection. The overall effectiveness of the DRD system depends heavily on the performance of collection ranking done at each agent involved.

In this chapter, we examine collection ranking for vector-based retrieval model. Since the collection ranking methods we will examine utilize term frequency statistics in a document, we will use the more specific term 'document' for a resource in this chapter. The concept of goodness score is first clarified in Section 4.1 and issues in their evaluation in Section 4.2. Information on the datasets we use in evaluating collection ranking methods is given in Section 4.3. We then examine several existing collection ranking methods in Section 4.4 and propose some new
methods and improvements in Section 4.5. In the last section, the best collection ranking methods in this chapter are compared.

4.1 An ideal goodness score

Given a user query \( q \) and an ARD \( ard_i \) representing a collection of documents \( C_i \) in \( P_i \), collection ranking method computes a scalar goodness score \( G_i(q) \) to estimate how relevant the documents in the collections are. Depending on our desired emphasis, \( G_i(q) \) will also be written as \( G(ard_i, q) \) or \( G(P_i, q) \). (We will encounter ARD not generated from RD aggregation of a collection in primary ISA in later chapter.) The query \( q \) is often omitted in the notation. We will list some requirements of collection ranking and devise an ideal goodness score for evaluation.

Let us first study the case of boolean-based retrieval model, in which each document is ranked as either relevant or non-relevant to a query. One may route the query to all collections with at least one relevant documents. This approach is used in Whois++ [15] and search in distributed directory service like X.500. However, when there are a large number of collections, it may incur a very high cost to query every such collections. Alternatively, the estimated number of relevant documents can be taken as a measure of goodness as in GIOSS [20]. One may then select those collections with relatively large goodness score for query routing. Goodness score should reflect the number of relevant documents in a collection.

Boolean-based retrieval model usually results in an unordered set of results too large for the user. In an ever-growing distributed information environment, it would be much more useful to have the results listed in descending order of estimated relevancy. Therefore, we assume a vector-based retrieval model. We use the well-known vector-based retrieval model TFxIDF [30] for document ranking in our collection ranking work. TFxIDF rates the relevancy of a document based on the frequency of terms in the document and the collection. Given a set of terms
\{t_1, t_2, \ldots, t_j, \ldots\} and a set of documents \(C^* = \{d_1, d_2, \ldots, d_k, \ldots\}\), term frequency \(TF_{k,j}\) is the times the term \(t_j\) appears in the document \(d_k\), and the document frequency \(DF_j\) is the number of documents in \(C^*\) that contains the term \(t_j\). A query \(q\) is represented as a set of terms present in the query. The following formula estimates the relevancy score of document \(d_k\):

\[
sim_k = \sum_{t_j \in q} (0.5 + 0.5 \frac{TF_{k,j}}{TF_{k,max}}) \cdot \log \left( \frac{N}{DF_j} \right)
\]

where \(N = \) the total number of documents in \(C^*\) and \(TF_{k,max} = \) the maximum term frequency in the document \(d_k\). For notational convenience, we will refer to the two terms in the product above as the (adjusted) term frequency \(tf_{k,j}\) and the inverse document frequency \(idf_j\), and use \(j\) to denote the term \(t_j\) when it is not ambiguous.

\[
sim_k = \sum_{j \in q} tf_{k,j} \cdot idf_j
\]

A collection \(C_i\) of documents is a subset of \(C^*\) and is represented by the ARD \(ard_i\). In collection ranking, we try to compare the goodness of a family of collections \(C = \{C_1, C_2, \ldots, C_i, \ldots\}\) with respect to a certain query \(q\). It is expected that the documents with higher relevancy score are more likely to be useful to the user. Obviously, the relevancy score should also be considered in computing goodness score. In addition, distribution of high relevancy score documents differs among \(C_i\)’s. For instance, collection \(C_1\) may have a few high score documents, while collection \(C_2\) has a lot of lower score documents. A collection ranking method should return suitable recommendation according to the user’s preference for precision or recall. This option is captured in the collection ranking parameters, \(s_{min}\) and \(r_{min}\), of a collection ranking method. The user may specify that only document with \(sim_k \geq s_{min}\) or \(sim_k \geq r_{min} \times \text{maxscore}\) should be included in goodness score, where \(0 \leq r_{min} < 1\) and \(\text{maxscore}\) is the score of the highest score documents among \(C_i\)’s. With these in mind, we define an ideal goodness score \(G_i^*(s_{min})\) as follows.

19
\[ G_i^*(s_{\text{min}}) = \sum_{d_k \in C_i, \text{sim}_k \geq s_{\text{min}}} \text{sim}_k \]

### 4.2 Evaluating collection ranking methods

If both collection ranking and collection selection are fully specified, one can evaluate the DRD system by inspecting the relevancy of documents returned for the sample query. We prefer to evaluate collection ranking separately by comparing it with an ideal goodness score. This helps to localize performance problem and avoids the complexity of the interaction between collection ranking and collection selection.

Two schemes to compare collection ranking are found in the literature. In [19], the authors evaluated the performance of gGLOSS using metrics similar to precision and recall. The ‘precision measure’ is the number of collections among the top \( n \) rank collections in the evaluated goodness score that actually belong to the top \( n \) rank collections in the ideal goodness score. The ‘recall measure’ is the ratio between the sum of the goodness scores of the top \( n \) rank collections in the evaluated goodness score and that in the ideal goodness score. [25] also proposes a similar scheme for boolean-based model.

View the result of collection ranking as a vector of \(|C|\) elements \( V = \langle G_1, G_2, ..., G_{|C|} \rangle \), where \( G_i \) is the goodness score of the collection \( C_i \). We adopt the more convenient evaluation method of computing some kind of similarity measure between the vector \( V \) and the vector \( U \) for an ideal goodness score. The cosine angle is taken as the similarity measure in [35]. It is the dot product of normalized \( U \) and \( V \), and ranges from 0 to 1, with 1 being exact match.

\[
\text{accuracy}(U, V) = \frac{U \cdot V}{|U| \cdot |V|} = \frac{\sum_i^{\mid C \mid} U_i V_i}{\sqrt{\sum_i^{\mid C \mid} U_i^2 \cdot \sum_i^{\mid C \mid} V_i^2}}
\]

Noting that if we are taking \( U \) and \( V \) as mere relative distribution of relevant documents, we can scale them linearly without violating the semantics. In particular, we normalize a vector \( U \) such that \( \sum_i U_i = 1 \). This views the vector
of goodness score as a point in a region on a hyperplane in the \(|C|\) dimensional space \(\langle x_1, x_2, ..., x_{|C|} \rangle\), where \(\sum_i x_i = 1\) and \(x_i \geq 0\), \(\forall i\). For example, this region is an equilateral triangle with the three vertices at \((1, 0, 0)\), \((0, 1, 0)\) and \((0, 0, 1)\) in the three dimensional space for the case \(|C| = 3\). It is bounded by the three lines \(x_1 + x_2 = 1, x_2 + x_3 = 1\) and \(x_3 + x_1 = 1\). Generally, the region is bounded by \(|C|\) hyperplanes of \(|C| - 1\) dimension \(\sum_{i \in S} x_i = 1\), where \(S \subseteq \{1...|C|\}\) is a subset of size \(|C| - 1\). This view suggests the Euclidean distance between the normalized \(U\) and \(V\) as a measure for difference between the ranking results. Since the maximum distance between two points in the hyperplane is \(\sqrt{2}\), we define the \text{diff}(U, V)\) measure as follows. It ranges from 0 to 1, with 0 being exact match.

\[
\text{diff}(U, V) = \frac{1}{\sqrt{2}} \cdot \sqrt{\sum_i \left( \frac{U_i}{|U|} - \frac{V_i}{|V|} \right)^2}
\]

The \text{diff}(U, V)\) measure is a stricter measure than the cosine angle measure near the boundary of the region on the hyperplane. It gives a slightly higher penalty to an inaccurate ranking towards the boundary than one towards the center.

### 4.3 Experiment datasets

From preliminary experiments we found that the relative performance of two collection ranking methods may change for different document sets and test queries. We speculate that it is due to the difference in the length of the queries and the distribution of relevancy scores in the collections. The performance is studied systematically by deriving four datasets from the Smart and Reuters test collections. See Table 4.1.

Datasets 1 and 2 are derived from four collections (CACM, CISI, CRAN and MED) from the Smart System developed at Cornell University (ftp://ftp.cs.cornell.edu/pub/smart/). Table 4.2 shows some statistics about the test collections. The four collections are from different domains. Each of them also
<table>
<thead>
<tr>
<th>dataset</th>
<th>documents</th>
<th>queries</th>
<th>no. of source</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7079</td>
<td>64</td>
<td>4</td>
<td>short query, Smart</td>
</tr>
<tr>
<td>2</td>
<td>7079</td>
<td>571</td>
<td>4</td>
<td>long query, Smart</td>
</tr>
<tr>
<td>3</td>
<td>9215</td>
<td>204</td>
<td>6</td>
<td>very long query, Reuters</td>
</tr>
<tr>
<td>4</td>
<td>10318</td>
<td>215</td>
<td>19</td>
<td>very long query, Reuters, overlap category</td>
</tr>
</tbody>
</table>

Table 4.1: The four base datasets used in the evaluation of collection ranking methods

comes with test queries. Both our datasets 1 and 2 take the four collections as source collections. Dataset 2 also uses the queries provided. In dataset 1, we try to test the case of short queries. The original CACM sample queries are in natural language. We manually shorten them by taking only important keywords to obtain the test queries of dataset 1.

<table>
<thead>
<tr>
<th></th>
<th>number of documents</th>
<th>number of queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACM</td>
<td>3204</td>
<td>64</td>
</tr>
<tr>
<td>CISI</td>
<td>1460</td>
<td>112</td>
</tr>
<tr>
<td>CRAN</td>
<td>1400</td>
<td>365</td>
</tr>
<tr>
<td>MED</td>
<td>1033</td>
<td>30</td>
</tr>
<tr>
<td>total</td>
<td>7079</td>
<td>571</td>
</tr>
</tbody>
</table>

Table 4.2: Some statistics of the Smart test collections

Datasets 3 and 4 are derived from the Reuters-21578 text categorization test collection (http://www.research.att.com/~lewis). Most of the documents in the test collection are assigned to one or more topics from a total of 135 categories in the category set TOPICS. We avoid overlapping between collections in dataset 3. See Table 4.3. Six ‘topic groups’ are chosen deliberately so that they have negligible overlaps. (The topic group “interest, money-fx” refers to the set of documents assigned to either “interest”, “money-fx”, or both categories.) The small number of documents that belong to more than one ‘topic groups’ are assigned to one arbitrarily. Dataset 4 utilizes the overlaps between categories.
The top 19 categories with the most documents are taken as source collections. Many documents appear in more than one categories. See Table 4.4. The fourth column is the number of documents in this category that also belong to other categories. The Reuters-21758 test collection comes with no sample queries. We randomly select 3% of the articles from a collection and use the first paragraph of a selected article as the query for the collection. This decision is justified by the fact that the documents in the test collection are news articles, in which the first paragraph usually summarizes the whole article.

<table>
<thead>
<tr>
<th>topic group</th>
<th>number of documents</th>
<th>number of queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq</td>
<td>634</td>
<td>16</td>
</tr>
<tr>
<td>crude</td>
<td>626</td>
<td>17</td>
</tr>
<tr>
<td>earn</td>
<td>525</td>
<td>15</td>
</tr>
<tr>
<td>grain</td>
<td>2418</td>
<td>65</td>
</tr>
<tr>
<td>interest, money-fx</td>
<td>3954</td>
<td>63</td>
</tr>
<tr>
<td>trade</td>
<td>1058</td>
<td>28</td>
</tr>
<tr>
<td>total</td>
<td>9215</td>
<td>204</td>
</tr>
</tbody>
</table>

Table 4.3: Some statistics of dataset 3 (disjoint collections from the Reuters-21578 test collection)

Each of our datasets has a number of source collection $C_i$'s. A $C_i$ holds documents of the same topic and is taken as a coarse topical cluster. The accuracy of a method may change if the collections ranked are not topically clustered. This is studied by generating different distribution from a dataset under the notion of document affinity probability [31], $P_a$. Given source collections $C = \{C_i\}$'s. We redistribute the documents in $C_i$'s to a set of destination collections $D = \{D_j\}$ on which collection ranking evaluation is performed. Each $C_i$ is assigned to one or more home collections $\text{Home}(C_i) \subset D$. A document in $C_i$ will be assigned to a non-home collection $D_j \notin \text{Home}(C_i)$ with the probability $(1 - P_a)/|D|$, and to $D_k \in \text{Home}(C_i)$ with the probability $P_a/|\text{Home}(C_i)| + (1 - P_a)/|D|$. For most of our tests we set $|D| = |C|$ and each $C_i$ has a unique home. We also consider a split case with $|D| = 2 \cdot |C|$ and each $C_i$ is assigned to two distinct homes.
<table>
<thead>
<tr>
<th>category</th>
<th>category name</th>
<th>documents</th>
<th>overlaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>acq</td>
<td>2448</td>
<td>67</td>
</tr>
<tr>
<td>17</td>
<td>coffee</td>
<td>145</td>
<td>23</td>
</tr>
<tr>
<td>20</td>
<td>corn</td>
<td>254</td>
<td>253</td>
</tr>
<tr>
<td>29</td>
<td>crude</td>
<td>634</td>
<td>191</td>
</tr>
<tr>
<td>33</td>
<td>dlr</td>
<td>217</td>
<td>210</td>
</tr>
<tr>
<td>36</td>
<td>earn</td>
<td>3987</td>
<td>34</td>
</tr>
<tr>
<td>44</td>
<td>gnp</td>
<td>163</td>
<td>43</td>
</tr>
<tr>
<td>45</td>
<td>gold</td>
<td>135</td>
<td>11</td>
</tr>
<tr>
<td>46</td>
<td>grain</td>
<td>628</td>
<td>527</td>
</tr>
<tr>
<td>56</td>
<td>interest</td>
<td>513</td>
<td>223</td>
</tr>
<tr>
<td>74</td>
<td>money-fx</td>
<td>801</td>
<td>440</td>
</tr>
<tr>
<td>75</td>
<td>money-supply</td>
<td>190</td>
<td>23</td>
</tr>
<tr>
<td>77</td>
<td>nat-gas</td>
<td>130</td>
<td>83</td>
</tr>
<tr>
<td>82</td>
<td>oilseed</td>
<td>192</td>
<td>121</td>
</tr>
<tr>
<td>109</td>
<td>ship</td>
<td>305</td>
<td>136</td>
</tr>
<tr>
<td>120</td>
<td>sugar</td>
<td>184</td>
<td>38</td>
</tr>
<tr>
<td>127</td>
<td>trade</td>
<td>552</td>
<td>111</td>
</tr>
<tr>
<td>130</td>
<td>veg-oil</td>
<td>137</td>
<td>53</td>
</tr>
<tr>
<td>131</td>
<td>wheat</td>
<td>306</td>
<td>305</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>10318</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Some statistics of dataset 4 (overlapping collections from the Reuters-21578 test collection)

The standard stemming and stop-word removal algorithms are applied. The performance of a collection ranking method is computed by the *diff* measure between the method and the ideal goodness score $G^*_i(r_{\text{min}})$ at several checkpoint values of $r_{\text{min}}$. The checkpoints are taken at $r_{\text{min}}$ instead of a fixed number of top $H$ documents as in [35] because the number of ‘highly relevant’ documents differs widely among the queries in the datasets.

### 4.4 Comparing existing methods

We compare the performance of some existing collection ranking methods in this section. A description of the calculation of the goodness scores is first given.
Afterwards, we will compare them using the datasets described in the previous section.

Given a user query $q$ and a family of collections $C = \{C_1, C_2, \ldots, C_i, \ldots\}$. $C_i \subset C^*$ is represented by the ARD $ard_i$. There are two kinds of goodness scores. $G_i$ may be a function of $ard_i$ and $q$ only, or it may depend on $ard_i$'s of all collections in $C$. We call them absolute and relative goodness scores respectively. Among the methods to be examined, $G_i(cvv)$, $G_i(cori)$ and $G_i(gloss, r_{min})$ are relative whereas $G_i(idf)$ and $G_i(gloss, s_{min})$ are absolute. Note that our ideal goodness score $G_i^*(s_{min})$ is also absolute.

All collection ranking methods considered in this work use statistical information like term and document frequency in the computation. The document frequency for collection $C_i$, $df_{i,j}$, is defined as the number of documents in $C_i$ which contain the term $j$. The (global) document frequency $DF_j$ is written as $df_j$ for coherence in notation. Also define weighted document frequency $wdf_{i,j} = \sum_{d_k \in C_i} tf_{k,j}$. $wdf_{i,j}$ can be viewed as $df_{i,j}$ weighted by $tf_{k,j}$, or put it the other way round, if all $tf_{k,j}$'s involved are 1, $wdf_{i,j} = df_{i,j}$. An ARD should include the above statistical information needed for the collection ranking methods used. Now we are ready to look at the definition of these goodness scores.

- **CVV**: CVV (Cue Validity Variance) was introduced in [35]. Intuitively, $cvv_j$ measures the collection discrimination power of term $j$. It is the population variance of the cue validity $cv_{i,j}$ of the term $j$ among the collections. In the original work, the goodness score of collection $C_i$ is computed as $\sum_{j \in q} df_{i,j} \cdot cvv_j$. It has the advantage that only document frequency is needed. However, our experiment shows that replacing $df_{i,j}$ with $wdf_{i,j}$ yields slightly, but constantly, better performance. We will use this modified version thereafter.

$$
G_i(cvv) = \sum_{j \in q} wdf_{i,j} \cdot cvv_j
$$

$$
cvv_{i,j} = \frac{df_{i,j}/|C_i|}{df_{i,j}/|C_i| + \frac{\sum_{k \in C_i} df_{k,j}}{\sum_{k \in C_i} |C_k|}}
$$
\* \textbf{C}ori: CORI (Collection Retrieval Inference Network) [12] is an extension of the INQUERY retrieval system [11] to the collection ranking problem. Inference network technology is applied in both document ranking and collection ranking. The collection ranking goodness scores are computed by replacing $tf_{k,j}$ with $df_{i,j}$ and $idf_j$ with $icf_j$. $icf_j$, the inverse collection frequency, is defined in analogy to $idf_j$ as the inverse of the collection frequency $cf_j$, which is the number of collections in which the term appears. For example, this transformation to the TFxIDF formula gives the following goodness score.

$$ G_i(icf) = \sum_{j \in q} df_{i,j} \cdot icf_j \quad \text{where} \quad icf_j = \log\left(\frac{|C|}{cf_j}\right) $$

The goodness scores in CORI is more sophisticated. It is computed as the combined belief $P(t_j|C_i)$ that $C_i$ contains relevant documents due to the presence of the $t_j \in q$.

$$ G_i(cori) = \prod_{j \in q} P(t_j|C_i) \quad \text{where} \quad P(t_j|C_i) = d_b + (1 - d_b) \cdot ndf_{i,j} \cdot icf_j $$

$$ ndf_{i,j} = d_t + (1 - d_t) \frac{\log(df_{i,j} + 0.5)}{\log(df_{i,\text{max}} + 1.0)} $$

$$ icf_j = \frac{\log(|C|/cf_j + 0.5)}{\log(|C| + 1.0)} $$

where $d_b$ and $d_t$ are the minimum belief and term frequency component when a term appears in a collection. They are set to 0.4 in our experiment. $df_{i,\text{max}}$ is the maximum of $df_{i,j}$ for all terms in $C_i$.

\* \textbf{gGLOSS:} We adopt the two assumptions in the High-Correlation Max($l$) case in gGLOSS [19] to get $G_i(gloss, s_{\text{min}})$. Name the terms in the query $q$ in non-decreasing order of $df_{i,j}$ as $a_1, a_2, ..., a_{|q|}$. Then take the following two assumptions about the distribution of the terms among the documents in the collection $C_i$.

First, a document that contains the term $a_k$ also contains the terms $\{a_{k'} : k' \geq k\}$. Second, a term appears evenly among the documents that contain it. Each term $j$ then contributes a term frequency factor $tf_{i,j}$ of $(wdf_{i,j}/df_{i,j})$ to each of these
'virtual documents'. We end up with \(|q|\) bands of virtual documents. Label this band as band \(a_0, a_1, \ldots, a_{|q|-1}\). The band \(a_k\) contains \(df_{i,a_{k+1}} - df_{i,a_k}\) documents each of which contains the terms \(\{a_{k'} : k' \geq k\}\). Set \(df_{i,a_0} = 0\). Denote the relevancy score of a virtual document in band \(a_k\) of collection \(C_i\) as \(sim^i_k\) and define the gGLOSS goodness score as the sum of the relevancy score of these virtual documents with \(sim^i_k \geq s_{\text{min}}\).

\[
sim^i_k = \sum_{j = a_{k'}, k \leq k' \leq n} (wdf_{i,j} / df_{i,j}) \times idf_j
\]

\[
G_i(\text{gloss}, s_{\text{min}}) = \sum_{0 \leq k < |q|, \sim^i_k \geq s_{\text{min}}} \sim^i_k \times (df_{i,a_{k+1}} - df_{i,a_k})
\]

The above summation in effect takes bands sequentially starting from band \(a_0\) up to \(a_{k^*}\), where band \(a_{k^*+1}\) is the first band with \(\sim^i_{k^*+1} < s_{\text{min}}\). Notice that the order of \(df_{i,j}\) may differ for each collection and so the above virtual document set has to be set up in the calculation of the goodness score for each collection. The virtual documents with the highest relevancy score in \(C_i\) are the ones in band \(a_0\), with relevancy score \(\sim^i_0\). Let \(\text{maxscore} = \max_i \sim^i_0\) and the gGLOSS goodness score with relative threshold \(r_{\text{min}}\) is defined as follows.

\[
G_i(\text{gloss, } r_{\text{min}}) = G_i(\text{gloss, } s_{\text{min}} = r_{\text{min}} \times \text{maxscore})
\]

- **IDF**: An easy definition of the goodness of a collection \(C_i\) is the sum of the relevancy scores of all documents in \(C_i\), i.e., \(G_i(idf) = \sum_{d_k \in C_i} \sim^i_k\). This measure has the nice property that it can be computed exactly using only the weighted document frequency \(wdf_{i,j}\) and the inverse document frequency \(idf_j\).

\[
G_i(idf) = \sum_{d_k \in C_i} \sim^i_k = \sum_{d_k \in C_i} \sum_{j \in q} t_{f_{k,j}} \cdot idf_j
\]

\[
= \sum_{j \in q} \left( \sum_{d_k \in C_i} t_{f_{k,j}} \right) \cdot idf_j
\]

\[
= \sum_{j \in q} wdf_{i,j} \cdot idf_j
\]
Note that $G_i(idf) = G_i^*(r_{\text{min}} = 0)$ from the definition of $G_i^*(r_{\min})$. Comparing it with the ideal goodness ranking $G_i^*(r_{\min})$ reveals how much the junk documents having low relevancy scores affect the ranking. Moreover, it is also noteworthy that $G_i(idf) = G_i(gloss, r_{\min} = s_{\min} = 0)$.

Table 4.5 is a summary of the above collection ranking methods. The inverse collection frequency $icf_j$ in CORI can be inferred from document frequency $df_{i,j}$. CVV does not use $wdf_{i,j}$ in the original work. The last column lists the collection ranking parameters taken.

<table>
<thead>
<tr>
<th>Method</th>
<th>absolute / relative</th>
<th>metadata in ARD</th>
<th>collrank-param</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVV</td>
<td>relative</td>
<td>$(wdf_{i,j})$</td>
<td>$df_{i,j}$</td>
</tr>
<tr>
<td>CORI</td>
<td>relative</td>
<td>$df_{i,j}$</td>
<td>$(icf_j)$</td>
</tr>
<tr>
<td>gGLOSS</td>
<td>absolute / relative</td>
<td>$wdf_{i,j}$</td>
<td>$df_{i,j}$</td>
</tr>
<tr>
<td>IDF</td>
<td>absolute</td>
<td>$wdf_{i,j}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5: Summary of some existing collection ranking methods

Experiments show that the relative performance of the above four methods remains similar for different values of $P_a$ for a certain dataset. Fig. 4.2 shows the evaluation of the methods for dataset 2 at $P_a = 0, 0.5, 0.8, 1$. (Preliminary experiments show that a slight change of $P_a$ for small $P_a$ yields less difference on the performance of the methods than that for larger $P_a$. So we use denser checkpoints for larger $P_a$. ) As $P_a$ increases, the difference in the accuracy among the methods tends to increase, but the order of the methods in terms of performance remains more or less the same. CORI does not perform well using our metric $\text{diff}(\cdot, \cdot)$. It is comprehensible because the score is meant to be the probability of the presence of relevant resources rather than the sum of relevancy scores. We will not include this in our further exploration. $G_i(gloss)$ performs much better than $G_i(idf)$ at large $r_{\min}$, and both of them converges to $\text{diff}(\cdot, \cdot) = 0$ at small $r_{\min}$ as expected. $G_i(cvu)$ behaves differently for different datasets. We will turn back to this later.
4.5 Observation and Improvement

We now concentrate on improving the absolute goodness scores in the previous section, namely, $G_i(idf)$ and $G_i(gloss)$. They are interesting representatives to show different approaches in using similar statistical information ($d_{f_i,j}$ and $wdf_{i,j}$) for collection ranking. The new or improved methods introduced in this section fall into two families. The first one is like $G_i(idf)$, with the general formula $\sum_{j \in q} wdf_{i,j} \cdot \delta_j$, where $\delta_j$ describes the importance of a term $j$. It is independent of the collection being ranked and is analogous to $idf_j$ in TFxIDF. The second family is like $G_i(gloss)$ and utilizes $df_{i,j}$ of each collection to obtain more accurate goodness score. We will refer to them as the IDF and gGLOSS family.

- **EIDF**: In the general formula of the IDF family, $G_i(idf)$ takes $\delta_j = idf_j$. $G_i(idf)$ measures the ideal goodness score at $r_{min} = 0$. In other words, it takes all documents with non-zero relevancy score into consideration. As shown in the experiment results in previous section, this ranking can be very different from $G_i^*(r_{min} > 0)$. Consider a typical result list. Fig. 4.1 depicts the list of documents in a collection in descending order of relevancy scores to a certain query. There are a few documents of high relevancy scores at the front, followed by a tail of low relevancy scores. $G_i(idf)$ is the area below the curve shown and the area of the shaded area denotes $G_i^*(s_{min})$, where $s_{min} = r_{min} \times$ (maximum relevancy score among the collections ranked). The effectiveness of $G_i(idf)$ depends on how long the ‘junk tail’ is. Since the junk tail consists mainly of documents that only contain terms of lower $idf_j$, we may reduce the effect of the junk tail to the goodness score by diminishing the term $\delta_j$ for lower $idf_j$. Our preliminary experiments showed that $\delta_j = (idf_j)^a$, $a = 4, 5$ or $6$ is a better value than $\delta_j = idf_j$. Later, we found that $\delta_j = idf_j \cdot e^{tidf_j}$ outperforms the fixed exponent adjustment.

One way to interpret the meaning of $G_i(eidf)$ is to consider it as an approximation of the score of highest rank document in the virtual document set of $G_i(gloss)$, which is $G_i = \sum_{j \in q} wdf_{i,j}/df_{i,j} \cdot idf_j$. Replace the adjustment term $df_{i,j}$ with the global document frequency $df_j$ to make it independent of the collection. (This is a
very coarse approximation.) We obtain \( G_i = \sum_{j \in q} wdf_{i,j} \cdot (idf_j/df_j) \). Incidentally, since the number of documents globally remains relatively constant, we may multiply the above formula by \( N \) to get the following 'equivalent', but more fancy, formula. This formula corresponds to \( \delta_j = (idf_j/df_j) \) or \( (idf_j \cdot e^{idf_j}) \).

\[
G_i(eidf) = \sum_{j \in q} wdf_{i,j} \cdot (idf_j \cdot e^{idf_j})
\]

- adjusted gGLOSS: Besides the junk tail, dataset 3 has revealed another problem of \( G_i(gloss) \) as in the previous section. When a query contains more than a few terms, with several terms with lower idf\(_j\), these terms often have low correlation. For example, consider the query "I want to find articles about agent communication language applied to distributed resource discovery system". The High-Correlation assumption will overestimate the goodness score of a collection by correlating lower idf\(_j\) terms like 'language' and 'system'. We propose to lessen this problem by deflating \( df_{i,j} \) for term \( j \) with lower idf\(_j\) before applying the original gGLOSS algorithm. This results in wider bands and therefore smaller average relevancy score towards the junk tail.

\[
df_{i,j} = df_{i,j} \cdot \alpha^{idf_{\text{max}}-idf_j}
\]

where \( idf_{\text{max}} = \max_{j \in q} idf_j \). From experiment, \( \alpha = 1.3 \) is an optimal value. We will write this score as \( G_i(gloss') \).

- GIDF, gGLOSS-like adjustment to IDF: Another way to improve \( G_i(idf) \) is to add an adjustment weight to \( \delta_j = idf_j \cdot w_j \) in a manner similar to \( G_i(gloss) \).
In order to make \( w_j \) independent of the collections, we use the global document frequency \( df_j \) instead of collection specific \( df_{i,j} \). Construct the virtual document set in gGLOSS on the hypothetical collection \( C_a \) with \( wdf_{a,j} = df_{a,j} = df_j \). The terms \( a_1, a_2, \ldots \) are in ascending order of \( df_j \), that is, in descending order of \( idf_j \). We have \( sim_k^a = \sum_{k' > k} idf_{a,k'} \). Now suppose band \( a_{k*} \) is the last band with \( sim_k^a \geq r_{\min} \cdot sim_0^a \), where \( sim_0^a = \sum_{j \in q} idf_j \). Write the term \( a_{k*} \) as \( j^* = a_{k*} \). Terms before \( j^* \) (in the order of \( a_1, a_2, \ldots \)) will contribute \( df_{a,j} \cdot idf_j \) to \( G_a(gloss) \), whereas terms after \( j^* \) will only contribute \( df_{a,j^*} \cdot idf_j \) as the rest \( df_{a,j} - df_{a,j^*} \) appearance in the rest bands is discarded. This is reflected in the following value of \( w_j \) for terms \( j \) after \( j^* \).

\[
w_j = \frac{df_{j^*}}{df_j} = \frac{N/df_j}{N/df_{j^*}} = e^{idf_j - idf_{j^*}}
\]

Experiment shows that the deflation adjustment in \( G_i(gloss') \) also improves the resultant goodness score. We incorporate this into the determination of the stopping band \( a_{k*} \). Band \( a_{k*} \) is the last band that still has \( sim_k^a \geq r_{\min} \cdot sim_0^a \cdot \alpha^{idf_{k*} - idf_{\max}} \). The deflation parameter \( \alpha \) is also taken to be 1.3 in our experiment.

Now we can define the goodness score \( G_i(gidf, r_{\min}) \).

\[
G_i(gidf, r_{\min}) = \sum_{j \in q} wdf_{i,j} \cdot idf_j \cdot w_j, \quad \text{where}
\]

\[
w_j = 1 \quad \text{if } k \leq k^*, \text{where } j = a_k
\]

\[
e^{idf_j - idf_{j^*}} \quad \text{otherwise}
\]

It is interesting to note that from the above definition \( G_i(gidf, r_{\min} = 0) = G_i(idf) \) and \( G_i(gidf, r_{\min} = 1) = G_i(eidf) \times e^{idf_{\max}} \), where \( idf_{\max} = \max_{j \in q} idf_j \). Since \( idf_{\max} \) is constant for a fixed query, the ranking given by \( G_i(gidf, r_{\min} = 1) \) and \( G_i(eidf) \) are scalar multiple of each other and they have \( \text{diff}(\cdot, \cdot) = 0 \).

- **WEIDF, weighted EIDF**: \( G_i(eidf) \) takes only the top rank document as a representation of a collection. We try to weight it by the (very coarse) estimated
number of documents with large enough relevancy score.

\[ G_i(weidf, r_{\text{min}}) = G_i(idf) \times W_i(r_{\text{min}}) \text{ where} \]

\[ W_i(r_{\text{min}}) = \sum_{j \in B} df_{i,j}, \quad B = \{ j \in q : idf_j \geq r_{\text{min}} \times \max_{j \in q} idf_j \} \]

<table>
<thead>
<tr>
<th></th>
<th>absolute/relative</th>
<th>metadata in ARD</th>
<th>collrank-param</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIDF</td>
<td>absolute</td>
<td>wdf_{i,j}</td>
<td></td>
</tr>
<tr>
<td>GIDF</td>
<td>absolute</td>
<td>wdf_{i,j}</td>
<td>r_{\text{min}}</td>
</tr>
<tr>
<td>GLOSS'</td>
<td>absolute/relative</td>
<td>wdf_{i,j}, df_{i,j}</td>
<td>s_{\text{min}} or r_{\text{min}}</td>
</tr>
<tr>
<td>WEIDF</td>
<td>absolute</td>
<td>wdf_{i,j}, df_{i,j}</td>
<td>r_{\text{min}}</td>
</tr>
</tbody>
</table>

Table 4.6: Summary of the newly introduced collection ranking methods

Table 4.6 compares the above methods. \( G_i(idf) \) and \( G_i(gidf, r_{\text{min}}) \) belong to the IDF family and \( G_i(gloss', r_{\text{min}}) \) and \( G_i(weidf, r_{\text{min}}) \) to the gGLOSS family. We have compared the above methods with datasets 1, 2 and 3 at \( P_a = 0, 0.5, 0.8, 1.0 \), and found that our new methods show improvement over the original ones. The improvement is larger for large \( P_a \), and is most obvious in dataset 3. Fig 4.3 shows the comparison for dataset 3 at \( P_a = 0.8 \). As expected, performance of \( G_i(gidf) \) converges to that of \( G_i(idf) \) for low \( r_{\text{min}} \), and coincides with that of \( G_i(idf) \) at high \( r_{\text{min}} \). \( G_i(gloss') \) performs better than \( G_i(gloss) \) for middle range \( r_{\text{min}} \), where correlation of lower idf_j terms is reduced. \( G_i(weidf)'s \) good performance is a little bit surprising, though, given the rough weight of \( \sum df_{i,j} \).

4.6 Chapter summary

We conclude this chapter by comparing the best methods we have encountered so far in this chapter: \( G_i(cuv), G_i(eidf), G_i(gidf, r_{\text{min}}), G_i(gloss', \cdot) \) and \( G_i(weidf, r_{\text{min}}) \).

It is important to clarify the collection ranking parameters accepted by the methods. \( G_i(cuv) \) and \( G_i(eidf) \) take no parameter. \( G_i(gidf, r_{\text{min}}) \) and \( G_i(weidf, r_{\text{min}}) \) take only \( r_{\text{min}} \). Nevertheless, this relative threshold \( r_{\text{min}} \) does not make them relative goodness score. \( G_i(gloss', s_{\text{min}}) \) is also absolute. However, \( G_i(gloss', r_{\text{min}}) \)
is a relative goodness score because the document with maximum relevancy score
can appear in any collections being compared. Because of the difficulty of using
relative goodness score in distributed collection selection in Chapter 5, we will
use the absolute version only. This raises the question of whether the evaluation
result of \( G_i(gloss', r_{\text{min}}) \) also applies to \( G_i(gloss', s_{\text{min}}). \) We think that the answer
is positive. The reason is that for a fixed query and a fixed set of collections to
be ranked, there is a mapping between \( s_{\text{min}} \) and \( r_{\text{min}}. \) The question now becomes
who and how to guess \( s_{\text{min}} \) if the user insists to specify his requirement as \( r_{\text{min}}. \)

The above five methods fall into three groups. \( G_i(cuv) \) is relative while the
rest are absolute goodness scores. \( G_i(eidf) \) and \( G_i(gidf, r_{\text{min}}) \) are of the IDF
family and have the general form \( \sum_{j \in q} wdf_{i,j} \cdot \delta_j, \) where \( \delta_j \) is independent of the
collection being ranked. \( G_i(gloss', s_{\text{min}}) \) and \( G_i(weidf, r_{\text{min}}) \) are of the gGIOSS
family and use collection specific statistics. \( G_i(gidf, r_{\text{min}}) \) and \( G_i(gloss', s_{\text{min}}) \)
converge to the ideal ranks at \( r_{\text{min}} = s_{\text{min}} = 0 \) by their definitions, while the
others do not.

We compare the five methods with dataset 1, 2 and 3. The results are shown
in Fig. 4.4, 4.5 and 4.6 respectively. Recall that dataset 1 uses short queries
and datasets 2 and 3 use longer queries. In all cases, the relative performance
of the methods remain similar for different values of \( P_a. \) Large \( P_a \) amplifies the
performance difference between the methods. The split case of these datasets are
also tested. We found that the plot of a split case is similar to one of the non-split
case with a smaller \( P_a. \) For example, in Fig. 4.7, the split case with \( P_a = 1.0 \)
looks like the non-split case with \( P_a = 0.8. \) It is also interesting to note that the
methods perform better for larger \( P_a. \)

The gGIOSS family yields more accurate ranks for large \( r_{\text{min}}, \) whereas for
small \( r_{\text{min}}, \) the IDF family has better performance. This break even point moves
towards \( r_{\text{min}} = 0 \) as the length of queries increases. \( G_i(eidf) \) and \( G_i(gidf, r_{\text{min}}) \) are
good candidates if the query is short and the user is more concerned about recall.
Overestimation of middle range relevancy score of \( G_i(gloss', s_{\text{min}}) \) becomes more
obvious for longer queries. For this case, \( G_i(weidf, r_{\text{min}}) \) works surprising well.

33
$G_i(cvv)$ has similar performance as $G_i(gidf, r_{\text{min}})$ for short and very long queries (dataset 1 and 3). However, in dataset 2, it performs better than $G_i(gidf, r_{\text{min}})$ (except for $r_{\text{min}} \approx 0$). It is also slightly better than the gGLOSS family for a small range of $r_{\text{min}}$ (e.g. 0.1–0.2 at $P_a = 0.8$, dataset 2). One may interpret this as that $G_i(cvv)$ is specialized to medium length query for the short range of $r_{\text{min}}$.

Finally, we run experiment on the dataset 4. Our aim is to see how well the collection ranking methods work for a set of topically clustered collections with overlaps. Fig. 4.8 shows the results. Note that the collection ranking methods do not care whether the collections overlap. The collections just look like a slightly mixed topically clustered case.

![Graphs showing performance comparison](image)

(a) $P_a = 0.0$  
(b) $P_a = 0.5$  
(c) $P_a = 0.8$  
(d) $P_a = 1.0$

Figure 4.2: Evaluation of existing collection ranking methods. (Dataset 2)
Figure 4.3: Evaluation of new and improved collection methods (Dataset 3, $P_a = 0.8$)
Figure 4.4: Evaluation of $G_i(cvv)$, $G_i(eidf)$, $G_i(gidf)$, $G_i(gloss')$ and $G_i(weidf)$ (Dataset 1)
Figure 4.5: Evaluation of $G_i(cvv)$, $G_i(eidf)$, $G_i(gidf)$, $G_i(gloss')$ and $G_i(weidf)$ (Dataset 2)
Figure 4.6: Evaluation of $G_i(cvv)$, $G_i(eidf)$, $G_i(gidf)$, $G_i(gloss')$ and $G_i(weidf)$ (Dataset 3)
Figure 4.7: Evaluation of collection methods in the split case (Dataset 3)

Figure 4.8: Evaluation of $G_i(cvv)$, $G_i(eidf)$, $G_i(gidf)$, $G_i(gloss')$ and $G_i(weidf)$ (Dataset 4)
Chapter 5

Macroscopic view – Architectures and Interoperation

Equipped with the essential microscopic tool of collection ranking developed in the previous chapter, we are now ready to proceed to the macroscopic issues of a DRD system. We will look into the internal working of the abstract ISA mesh which performs collection selection.

The abstract ISA mesh cooperates with the IUA to select a set of primary ISA $P_j$'s that are likely to yield relevant results. This process is called collection selection. These $P_j$'s upload ARDs describing their collections. Their appropriateness to answer a query is measured by the goodness score developed earlier in Chapter 4. Assume each $P_j$ uploads a single ARD $ard_j$ that summarizes his collection. We will write the goodness score of the collection as $G_j$, $G(ard_j)$, or $G(P_j)$.

Obviously, this is not desirable for the abstract ISA mesh just to return a list of all $P_j$ and $G(P_j)$ to the IUA and expect the user to inspect each of them. We will consider two ways to let a user specify criteria for 'good enough' collections that deserve query routing. The user can enforce a minimum goodness $g_{min}$. The IUA will then only route the query to those $P_j$'s with $G(P_j) \geq g_{min}$. Sometimes, however, the user does not understand the scale of the goodness score. He may
also worry about setting a threshold that yields too many or too little collections. As an alternative to an absolute threshold \( g_{\text{min}} \), the user may set a relative threshold and specify that the top \( n \) collections to be selected for query routing. We will refer to these as collection selection parameters. In addition, goodness score calculation takes collection ranking parameters \( s_{\text{min}} \) or \( r_{\text{min}} \) (which, incidentally, are 'good enough' criteria for documents). The IUA and the abstract IUA mesh should then take both collection ranking and selection parameters into consideration in their cooperation.

We now turn our attention to the core problem of this thesis — structuring a group of abstract ISAs to perform collection selection in a distributed manner. The nature of these abstract ISAs, which remains unexplained in the abstract model in Chapter 3, will be discussed in detail in the context of the architectures. An abstract ISA architecture consists of three aspects: the topology of the abstract ISA mesh, the ARDs transferred among the ISAs, and the navigation algorithm the IUA used to navigate the abstract ISA mesh. Two basic architectures are first considered. The first three Sections 5.1, 5.2 and 5.3 discuss the pure hierarchical architecture in detail. The next two Sections 5.4 and 5.5 examine the flat architectures. We then compare the two architectures and mix their features to produce two hybrid architectures in Section 5.6. The interoperability of the architectures considered is addressed in Section 5.7. Finally, we will consider other issues in the last section.

5.1 Hierarchical architecture

Hierarchical structure is widely used in computer science. In fact, in gGLOSS [19], a two-layer architecture of meta-indexer is mentioned to perform distributed collection selection. It is conventionally referred to as a hyper-indexer on meta-indexers. However, that remains to be a brief note lacking details. There lacks a rigorous model for hierarchical architecture for DRD system.

We define a hierarchical architecture for decentralizing collection ranking and
selection as a tree of abstract ISAs known as *meta ISA* (MISA for short). See Fig. 5.1. The meta ISAs $M_i$'s and their acquaintance relations form a directed tree called meta ISA tree. There is a single root MISA $M_r$, and there are no loops or cross-linkings. Therefore, $M_i$'s form a layered structure in which an $M_i$ receives indexing information from the MISAs on the layer below. The leaf nodes of the MISA tree have primary ISAs $P_j$'s as acquaintance, i.e., $Acq(M_i) \subseteq \{P_j\}'s$. In addition, a $P_j$ only uploads to (at most) one $M_i$ and so $Acq(M_i)'s$ of leaf nodes do not overlap. Incidentally, the degenerate case of a single MISA corresponds to a centralized collection selection system — meta-indexer.

Each $P_j$ uploads an ARD of their RD base to the leaf nodes in the MISA tree. For any $M_i$ except the root node, the ARDs received from acquaintances on the lower layer ISAs are merged into a single ARD, which is then uploaded. There is an upward cumulative propagation of ARDs from $P_j$'s to $M_r$. The locator part of the ARD uploaded by $M_i$ (of the form misa://misa-i) points to $M_i$ himself and the metadata part summarizes the resources in the collections of $P_j$'s underneath the subtree rooted at $M_i$. The merging of metadata in the ARDs is specific to the collection ranking methods in use. The computation of the merged ARD for the goodness scores discussed in Chapter 4 will be given in Section 5.2.

IUA's navigation in the MISA tree is the most unclear point in hierarchical DRD architectures in the literature, unfortunately. We explore two views of the MISA tree: a distributed search tree and a hierarchical partitioning of collections at $P_j$'s. It turns out that ARD merging differs for these two views.

### 5.1.1 MISA tree as search tree

Our first approach is to treat the MISA tree as a search tree. Generally, a search tree is a tree with the objects to be searched at the leaf nodes. Key information about child nodes is stored in each internal node. The search proceeds as traversing the tree from the root down to the leaves. In our model, a MISA tree together with $P_j$'s forms a search tree. The key information in the RD base
Figure 5.1: The pure hierarchical architecture: MISA tree on top of primary ISAs of the MISA $M_i$ is the result of merging ARDs. We need to design ARD merging method and the traversal algorithm such that good enough $P_j$’s will be returned.

It is instructive to examine the case for boolean-based retrieval. In such system, the presence of certain keywords in a query is taken as relevancy. Satisfactory ARD merging is easy. For example, in Whois++ [10, 15], resource description is a list of attributes and their values. The query is to check whether certain keywords appear in certain attributes. A user asking for information about “Beethoven’s Pastoral symphony” may formulate his query as “category: classical-music; author: Beethoven; keywords: pastoral symphony”. Both centroid computation (resource description aggregation) and centroid merging (ARD merging) are defined as union of the values in each attribute. If a keyword appears in the centroid of certain node, it will also appear in the centroid of its parent node (upstream acquaintance). Conversely, if the user agent finds that a keyword is not in the centroid of a node, then there is no need to pursue the branch anymore.

Analogous fruitless branch pruning can be performed in a vector-based retrieval system considered in this work. The goodness of the collection of a primary ISA $P_j$ is measured as $G(P_j)$ using a certain collection ranking method. Assume each $M_i$ uploads a single merged ARD and write $G(M_i)$ for the goodness score calculated from that ARD using the same collection ranking method.
We will attempt to design ARD merging such that $G(\mathcal{M}_i) \geq G(\mathcal{P}_j)$ for all $\mathcal{P}_j$ at the leaf nodes of the subtree rooted at $\mathcal{M}_i$. This can be achieved by enforcing the following at each MISA: $G(\mathcal{M}_i) \geq G(\mathcal{A}), \forall \mathcal{A} \in \text{Acq}(\mathcal{M}_i)$. This goodness score summarizes the merging ARDs by estimating the best $G(\mathcal{P}_j)$ underneath the subtree. We therefore call this max-based ARD merging and write the goodness score as $G^\text{max}(\cdot)$. Note that $G^\text{max}(\mathcal{P}_j) = G(\mathcal{P}_j)$ as there is no merging. The above requirement is restated in the following optimistic principle, a.k.a., never underestimate your offspring principle. We use the operator $\bigoplus$ to denote general ARD merging.

**Optimistic principle in ARD merging:** Assume $\mathcal{A}_i \in \text{Acq}(\mathcal{M})$ uploads ARD $\text{ard}_i$ to $\mathcal{M}$. If $\mathcal{M}$ in turns uploads the merged ARD $\text{ard}_m = \bigoplus_i \text{ard}_i$, he should have $G^\text{max}(\text{ard}_m) \geq G^\text{max}(\text{ard}_i), \forall i$ for all queries.

With satisfactory max-based ARD merging at each MISA, we can now devise navigation algorithm that prunes fruitless branch using $G^\text{max}(\cdot)$. A tree traversal algorithm for collection selection parameter $g_{\text{min}}$ is shown in the pseudocode IUA::GetGoodEnoughPisa in Fig. 5.2. A $\mathcal{P}_j$ is good enough iff $G(\mathcal{P}_j) \geq g_{\text{min}}$. The IUA traverses the MISA tree using depth-first search down from the root $\mathcal{M}_r$. The traversal proceeds as a sequence of query routing down the MISA tree. The IUA submits the query, the collection selection parameter $g_{\text{min}}$ and collection ranking parameters to a MISA and gets back a list of child ISAs that have $G^\text{max}(\cdot) \geq g_{\text{min}}$. The list can either be all MISAs (with locator like misa://misa-name) or PISAs (with locator like pisa://pisa-name). If for a certain MISA $G(\mathcal{M}_i) < g_{\text{min}}$, it follows from the enforcement of the optimistic principle at each MISA that there will be no $\mathcal{P}_j$ underneath $\mathcal{M}_i$ with $G(\mathcal{P}_j) \geq g_{\text{min}}$. Therefore, this fruitless subtree can be pruned confidently. A MISA does not need to return such an unpromising acquaintance to the IUA. This pruning decision can also be done in the IUA (not shown). In this case, the MISA just returns all the ARDs uploaded by his acquaintance and the IUA does the ranking. The IUA may then discard the unpromising MISAs in query routing. At the end of the traversal, the IUA will
obtain a list of all $P_j$'s that have goodness score above $g_{\text{min}}$. The navigation in MISA tree is said to be complete, meaning that all good enough $P_j$'s are retrieved at the end of the navigation. Note that query routing to individual $M_i$ can be done in parallel.

```plaintext
function MISA::ProcessQuery (self, q, collsel-param, collrank-param)
begin
    // collsel-param stands for collection selection parameters.
    // collrank-param stands for collection ranking parameters.
    // collsel-param consists of $n$ and $g_{\text{min}}$.
    // perform collection ranking of my acquaintance
    // return a list of my acquaintances that are good enough, that is,
    // the top $n$ $M_i$ or $P_j$ in $Acq(self)$ with $G^\text{max}(\cdot) \geq g_{\text{min}}$
    return $L_{\text{good}}$child
end

function IUA::GetGoodEnoughPisa ($M_{\text{root}}$, q, $g_{\text{min}}$, collrank-param)
begin
    // $L_{\text{good}}$ is a list of Pisas that are good enough
    $L_{\text{good}} := \text{empty list}$
    $L_{\text{child}} := \text{MISA::ProcessQuery($M_{\text{root}}$, q, $g_{\text{min}}$, collrank-param)}$
    if $L_{\text{child}}$ contains only PISAs
        $L_{\text{good}} := L_{\text{child}}$
    else
        // $L_{\text{child}}$ contains only MISAs with goodness $\geq g_{\text{min}}$
        for each MISA $M_i \in L_{\text{child}}$ // can be parallelized
            $L' := \text{IUA::GetGoodEnoughPisa($M_j$, q, $g_{\text{min}}$, collrank-param)}$
        concatenate $L'$ to $L_{\text{good}}$
    end
    return $L_{\text{good}}$
end
```

Figure 5.2: Navigation algorithm for MISA tree: collection selection parameter $g_{\text{min}}$

**IUA::TraverseMisaTree** in Fig. 5.3 describes an algorithm to find the top $n$ $P_j$'s with largest goodness. This algorithm is also complete. The while loop of the algorithm will stop in two cases. The first is when all $M_i$'s ever appearing in $Q$ have been visited and $Q$ contains only PISAs now. The other case is when any unvisited $M_i$ in $Q$ is after the first $n$ entries (which are all PISAs). These
\( M_i \)'s and therefore any \( P_j \)'s under them have goodness scores not greater than that of the top \( n \) PISAs. It is not necessary to pursue these branches anymore. The absolute threshold \( g_{\text{min}} \) is also included in the code to show its versatility. One may set \( g_{\text{min}} = 0 \) if he only wants the top \( n \) \( P_j \)'s to be returned regardless of how low their goodness scores might be. On the other hand, if one is interested only in \( P_j \) with goodness scores above \( g_{\text{min}} \), he can set \( n = \infty \), or he may use the simpler IUA::GetGoodEnoughPisa algorithm.

```c
function IUA::TraverseMisaTree (M_root, q, collsel-param, collrank-param)
begin
   // collsel-param consists of \( n \) and \( g_{\text{min}} \).
   // \( Q \) is a list of Misas or Pisas in descending order of goodness
   Q := empty list
   append \( M_{\text{root}} \) to \( Q \)
   while (some of first \( n \) in \( Q \) are MISA)
      \( M_{\text{next}} \) := the highest score MISA in \( Q \), extract it from \( Q \)
      \( L_{\text{child}} \) := MISA::ProcessQuery(\( M_{\text{next}} \), q, collsel-param, collrank-param)
      append to \( Q \) each \( M_i \) or \( P_j \) in \( L_{\text{child}} \)
   end
   return the first \( n \) of \( Q \)
end
```

Figure 5.3: Navigation algorithm for MISA tree: general collection selection parameter

### 5.1.2 MISA tree as a hierarchical partitioning of collections

In the MISA search tree approach, the ARD \( ard_m \) uploaded by a MISA \( M_i \) typically represents the 'maximum' aspect of his acquaintance. The IUA can know that a 'good enough' \( P_j \) is likely to be under it. Nevertheless, the goodness score calculated from \( ard_m \) gives no hints of the number of 'good enough' \( P_j \)'s underneath. We are limited by the expressiveness of a scalar goodness score. In the MISA search tree approach, this single scalar refers to the goodness of a single 'best' \( P_j \).
Our second approach takes an alternative view. We see the MISA tree as a hierarchical partitioning of the universal document set, and try to design ARD merging so that the goodness score computed from a merged ARD $ard_m$ represents the total goodness of the $P_j$'s underneath the tree. We call this sum-based ARD merging and denote the goodness score as $G^{\text{sum}}(\cdot)$. Similar to the max-based case, $G^{\text{sum}}(P_j) = G(P_j)$ as there is no merging. Take the MISA tree in Fig. 5.1 as an example. Suppose the primary ISA $P_j$ hosts the collection $C_j$, $j = 1..6$, and that goodness scores are computed with the parameter $s_{\text{min}}$. Ideally, we would like $G^{\text{sum}}(M_1)$ to be the sum of the relevancy scores of the documents in the virtual collection $C_1 \cup C_2$ having relevancy scores above $s_{\text{min}}$. Similarly, $G^{\text{sum}}(M_2)$ should be equal to the score for $\bigcup_{j=3,5} C_j$.

Let $L$ be the set of $P_j$'s at the leaf node of the subtree rooted at $M_i$. For a given collection ranking method, the goodness score $G^{\text{sum}}(M_i)$ will be equal to $\sum_{P \in L} G(P)$ if for any non-leaf node $M$, we have $G^{\text{sum}}(M) = \sum_{A \in Acq(M)} G^{\text{sum}}(A)$. This property is called additivity. However, it is not always possible to merge ARDs such that the equality holds. We may need to drop back to an upper bound $G^{\text{sum}}(M) \geq \sum G^{\text{sum}}(A)$.

$G^{\text{sum}}(\cdot)$ alone might not be useful in traversing the MISA tree. For example, suppose $G^{\text{sum}}(M_2) = a \cdot g_{\text{min}}$. One cannot decide whether any $P_j$ underneath $M_2$ has $G(P_j) \geq g_{\text{min}}$ if $a < 4$. Similarly, $G^{\text{sum}}(M_2) > G^{\text{sum}}(M_1)$ does not imply that there is a $P_{j_2}$ under $M_2$ that a has goodness score larger than any $P_{j_1}$ under $M_1$. Instead sum-based ARD merging provides valuable assistant information in selecting the next MISA to expand in MISA tree traversal. In choosing the next $M_i$ to visit, the IUA can consider both $G^{\text{max}}(M_i)$ and $G^{\text{sum}}(M_i)$. He may prefer to first visit $M_i$ with approximately equal $G^{\text{max}}(\cdot)$ but with a much larger $G^{\text{sum}}(\cdot)$. This helps to retrieve most good enough $P_j$'s at the beginning of the tree traversal.

A second usage of $G^{\text{sum}}(\cdot)$ is to select an alternative starting point for MISA tree traversal. Suppose $M_b \in Acq(M_a)$. If $G^{\text{sum}}(M_b) \approx G^{\text{sum}}(M_a)$, then most resources with large enough relevancy scores (as defined by the collection ranking
parameters) appear in collections in the subtree rooted at $\mathcal{M}_b$. It is very likely that visiting $\mathcal{M}_a$ only results in query routing to $\mathcal{M}_b$. The IUA can save this routing if he can estimate in advance $G^\text{sum}(\cdot)$ for the two MISAs. Such mechanism can be implemented as a caching function of (fragment of) ARDs and collection ranking results in the IUA or in a common agent serving a community of users with common interest. This is similar to the concept domain name resolution caching in the DNS system [27].

We call the set of $\mathcal{P}_j$'s used in the computation of an ARD its coverage. As an $\mathcal{M}_i$ only uploads a single merged ARD, we will also call this the coverage of $\mathcal{M}_i$. The coverage of the ARDs involved in the previous case has a superset-subset relationship. $G^\text{sum}(\cdot)$ is also useful in the more general case of selecting abstract ISA with overlapping coverage. This will be discussed in later sections.

### 5.2 ARD merging

In the previous section, two kinds of ARD merging are identified. Max-based ARD merging should satisfy the overestimate principle, whereas sum-based ARD merging should observe additivity. Satisfactory ARD merging operation varies for different collection ranking methods.

We will only consider ARD merging for absolute collection ranking methods discussed earlier in Chapter 4. The relative goodness scores $G_i(\text{cori})$ and $G_i(\text{cvu})$ take advantage of the relative distribution of $\text{wdf}_{i,j}$ and $\text{df}_{i,j}$ among the collections being ranked. The goodness score of a certain ARD will vary if it is compared with different sets of ARDs. Therefore, one may not compare the relative goodness score computed in two subtrees of the MISA tree. Methods using relative goodness score needs to recompute the goodness scores every time the set of ARDs being compared changes.

Both kinds of merging operate on $\text{wdf}_{i,j}$ and $\text{df}_{i,j}$. The merged metadata for max-based and sum-based ARD merging are different. For brevity, we skip labeling the metadata for a specific kind of merging in the following discussion.
and use the name \( ard_i \) to encapsulate the necessary metadata. Different notations will be used to emphasize their difference. The operator \( \odot \) denotes max-based merging while \( \oplus \) sum-based merging.

### 5.2.1 Max-based ARD merging

Max-based ARD merging \( ard_m = \bigodot_i ard_i \) satisfies \( G(ard_m) \geq G(ard_i), \forall i \). It is desirable that the operation \( \bigodot \) is commutative and associative. This ensures that the order of merging is not significant. Moreover, it is important that the merging is also idempotent, i.e., \( ard_a \odot ard_a = ard_a \). This guarantees that repeated merging of some ARDs is not harmful. It can be verified easily that the merging operations we are to define in this section have these properties.

Max-based ARD merging for the IDF family is easy. \( G_i(idf), G_i(cidf) \) and \( G_i(gidf, r_{\text{min}}) \) have the common form \( \sum_{j \in q} wdf_{i,j} \cdot \delta_j \). Since \( \delta_j \) depends only on \( idf_j \) for \( j \in q \) and the parameter \( r_{\text{min}} \), the following natural definition of merging is satisfactory.

\[
wdf_{m,j} = \max_i wdf_{i,j} \quad df_{m,j} = \max_i df_{i,j}
\]  \hspace{1cm} (5.1)

Satisfactory ARD merging for \( G_i(weidf, r_{\text{min}}) = G_i(cidf) \cdot W_i(r_{\text{min}}) \) is also simple. It uses both \( wdf_{i,j} \) and \( df_{i,j} \). The merging in (5.1) also satisfies the optimistic principle. The proof is trivial when one notices that the set \( B = \{ j \in q : idf_j \geq r_{\text{min}} \times \max_{j \in q} idf_j \} \) depends only on the query and the \( idf_j \) for \( j \in q \), and so \( W_m(r_{\text{min}}) > W_i(r_{\text{min}}) \).

Lastly we consider \( G_i(gloss, s_{\text{min}}) \) and \( G_i(gloss', s_{\text{min}}) \). Both \( wdf_{i,j} \) and \( df_{i,j} \) are used in the ARD. (5.1) is not sufficient to guarantee \( G_m(\cdot, \cdot) \geq G_i(\cdot, \cdot) \) for these goodness scores. Here is a counter example. Let \( \text{Acq}(A_m) = \{ A_1, A_2 \} \) and the query \( q \) be the single term \( j \), and \( wdf_{1,j} = wdf_{2,j} = 2s_{\text{min}}, \, df_{1,j} = 2 \) and \( df_{2,j} = 4 \). The formula above will give \( wdf_{m,j} = 2s_{\text{min}} \) and \( df_{2,j} = 4 \). This results in \( G_m(gloss, s_{\text{min}}) = 0 < G_1(gloss, s_{\text{min}}) = wdf_{1,j} \cdot idf_j \).
However, notice that $G_m(idf)$ calculated from a merged ARD based on (5.1) is an upper bound of $\max_i G_i(gloss)$ since $G_i(idf) \geq G_i(gloss, s_{\min})$ for each $i$. In addition, $G_m(gloss)$ computed from a merged ARD by the following merge function (5.2) also is an upper bound. The proof is more difficult, partly because of the parameter $s_{\min}$ and the fact that the $df_{i,j}$ may be of different order for each collection. Note that this lemma also justifies the case of $G_i(gloss', s_{\min})$, because the adjustment to $df_{i,j}$ depends only on $idf_j$ for $j \in q$, and is therefore the same for both $ard_a$ and $ard_b$. If both ARDs satisfy the condition in the lemma, then both ARD after adjustment will also satisfy the condition.

\[
\begin{align*}
    df_{m,j} &= \max_i df_{i,j} \\
    wdf_{m,j} &= \max_i (wdf_{i,j}/df_{i,j}) \times \max_i df_{i,j}
\end{align*}
\] (5.2)

Which of the above two upper bounds gives a tighter bound depends on $s_{\min}$. Consider the example in Table 5.1. Let $Acq(A_m) = \{A_1, A_2\}$, and the query be the two terms $\{1, 2\}$ with $idf_1 = 2$ and $idf_2 = 1$. $G_m(idf)$ based on (5.1) is tighter than $G_m(gloss)$ based on (5.2) for $s_{\min} = 0$, while the latter is tighter for $s_{\min} = 3$. It is therefore desirable to keep two sets of $wdf$ and $df$ as the merged results of (5.1) and (5.2) and use the tighter one from the two bounds.

<table>
<thead>
<tr>
<th>ARDs of</th>
<th>$wdf_{1,1}, wdf_{1,2}$</th>
<th>$df_{1,1}, df_{1,2}$</th>
<th>$adf_{1,1}, adf_{1,2}$</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>2, 10</td>
<td>2, 10</td>
<td>1, 1</td>
<td>$G_1(3) = 6$, $G_1(0) = 14$</td>
</tr>
<tr>
<td>$A_2$</td>
<td>2, 10</td>
<td>2, 20</td>
<td>1, 0.5</td>
<td>$G_2(3) = 0$, $G_2(0) = 14$</td>
</tr>
<tr>
<td>$A_m$ from (5.1)</td>
<td>2, 10</td>
<td>2, 20</td>
<td>1, 0.5</td>
<td>$G_m(idf) = 14$</td>
</tr>
<tr>
<td>$A_m$ from (5.2)</td>
<td>2, 20</td>
<td>2, 20</td>
<td>1, 1</td>
<td>$G_m(3) = 6$, $G_m(0) = 24$</td>
</tr>
</tbody>
</table>

Table 5.1: An example comparing ARD merging functions for $G_i(gloss)$

**Theorem 5.1** Given two ARDs $ard_a$ and $ard_b$ and a query $q$. Denote $adf_{i,j} = wdf_{i,j}/df_{i,j}$. If $adf_{a,j} \geq adf_{b,j}$ and $df_{a,j} \geq df_{b,j}, \forall j \in q$, then $G_a(gloss, s_{\min}) \geq G_b(gloss, s_{\min}), \forall s_{\min}$.
In the following paragraphs, we will formulate the proof of Theorem 5.1 using the notion of sequence. For our purpose, a sequence $\langle a_k \rangle$ is a list of elements from a certain set with no duplicates. We will use the set $q \cup \{\alpha, \omega\}$, where $q$ is the set of terms in the query and $\alpha$ and $\omega$ are two artificial terms not in $q$. The notation $\langle a_k \rangle$ will also be used as the set of elements inside. The elements of $\langle a_k \rangle$ are named sequentially as $a_0, a_1, \ldots, a_k, \ldots$. Furthermore, the order of the elements in $\langle a_k \rangle$ induces an injective function $f_a : \langle a_k \rangle \to 0..|\langle a_k \rangle| - 1$. For example, if $a_5$ is term 3, then $f_a(\text{term 3}) = 5$.

Given two sequences $\langle a_k \rangle$ and $\langle b_k \rangle$ on the same set $T = q \cup \{\alpha, \omega\}$, and $|\langle a_k \rangle| = |\langle b_k \rangle| = |T|$, and $a_0 = b_0 = \alpha$ and $a_{|T|-1} = b_{|T|-1} = \omega$. In other words, $\langle a_k \rangle$ and $\langle b_k \rangle$ start and end with the same elements. Use the following algorithm to obtain a subsequence $\langle c_k \rangle$. The enumeration order of $a_k$'s in the for loop and the condition $f_b(j) > f_b(c_l)$ ensure that $\langle c_k \rangle$ is a common subsequence of $\langle a_k \rangle$ and $\langle b_k \rangle$. In other words, if $x, y \in \langle c_k \rangle, f_c(x) < f_c(y)$, then $f_a(x) < f_a(y)$ and $f_b(x) < f_b(y)$.

\[
\langle c_k \rangle = \text{empty list}
\]

append $c_0 = \alpha$ to $\langle c_k \rangle$

// invariant: terms in $\langle c_k \rangle$ appear sequentially in both $\langle a_k \rangle$ and $\langle b_k \rangle$

for each $j \in a_1, a_2, \ldots, a_{|T|}$

\[
c_l := \text{last term in } \langle c_k \rangle
\]

if $f_b(j) > f_b(c_l)$ then append $j$ to $\langle c_k \rangle$

append $c_n = \omega$ to $\langle c_k \rangle$

return $\langle c_k \rangle$

Lemma 5.1 Given $j \in \langle a_k \rangle$ but $j \not\in \langle c_k \rangle$. Suppose $x$ is the first element before $j$ in $\langle a_k \rangle$ that is also in $\langle c_k \rangle$, i.e. $x \in \langle c_k \rangle$ and $f_a(x) < f_a(j)$ and $\not\exists x' \in \langle c_k \rangle, f_a(x') < f_a(x') < f_a(j)$. Then $f_b(x) > f_b(j)$.

Proof: First, notice that generally $f_b(i) \neq f_b(j) \iff i \neq j \iff f_a(i) \neq f_a(j)$. We have to show that it is impossible to have $f_b(x) < f_b(j)$. Assume to the
contrary that $f_b(x) < f_b(j)$. Let $y$ be the first element after $j$ in $\langle b_k \rangle$ that is also in $\langle c_k \rangle$, i.e. $y \in \langle c_k \rangle$ and $f_b(j) < f_b(y)$ and $\forall y' \in \langle c_k \rangle, f_b(j) < f_b(y') < f_b(y)$. If $f_a(j) < f_a(y)$, then $j$ would have been appended to $\langle c_k \rangle$ after $x$ by the algorithm and so $j \in \langle c_k \rangle$, a contradiction. Alternatively, consider the case $f_a(y) < f_a(j)$. The minimality of $x$ prohibits $f_a(x) < f_a(y) < f_a(j)$. We also cannot have $f_a(y) < f_a(x) < f_a(j)$ as $f_b(x) < f_b(j) < f_b(y)$ and this means that $x$ and $y$ are in different order in $\langle a_k \rangle$ and $\langle b_k \rangle$. And $x \neq y$ because of the strict inequality in $f_b(x) < f_b(j) < f_b(y)$. □

**Lemma 5.2** Let $S(\langle a_k \rangle, j)$ be the set $\{a_k : a_k \in \langle a_k \rangle, k > f_a(j)\}$. Then for any $x \in \langle c_k \rangle$, $S(\langle b_k \rangle, x) \subset S(\langle a_k \rangle, x)$.

**Proof:** We prove by induction on the elements of $\langle c_k \rangle$. $S(\langle b_k \rangle, \alpha) = S(\langle a_k \rangle, \alpha)$ and so the base case is true. Now let $x, y \in \langle c_k \rangle$ be two consecutive elements in $\langle c_k \rangle$ ($f_c(y) = f_c(x) + 1$), and $S(\langle b_k \rangle, x) \subset S(\langle a_k \rangle, x)$. By lemma 5.1, for any $t \in \langle a_k \rangle$ with $f_a(x) < k < f_a(y)$, $f_b(t) < f_b(x)$, and so $t \notin S(\langle b_k \rangle, x)$ and $S(\langle b_k \rangle, x) \subset S(\langle a_k \rangle, t)$. Finally, the element $y$ is deducted from both $S(\langle b_k \rangle, x)$ and $S(\langle a_k \rangle, t)$ and so we have $S(\langle b_k \rangle, y) \subset S(\langle a_k \rangle, y)$. □

**Proof of Theorem 5.1:** First, we introduce some convention and notation.

Let the user query $q$ be a set of terms. For terms $j_0 \in q$ with $df_{a,j_0} = 0$, set $wdf_{a,j_0} = 0$ and $df_{a,j_0} = df_{a,max}$, where $df_{a,max} = \max_{j \in q} df_{a,j}$. This would not change the value of $G_a(gloss, s_{min})$. Name the terms in $q$ in non-decreasing order of $df_{a,j}$ as $a_1, a_2, ..., a_{|q|}$. For notational consistency, append an artificial term $a_0 = \alpha$ with $df_{a,\alpha} = 0$ at the front of the sequence. Call this sequence $\langle a_k \rangle$. Apply similar adjustment to $ard_a$ to obtain $\langle b_k \rangle$.

Using the two assumptions in the derivation of $G_a(gloss, s_{min})$, we obtain a virtual document set of staircase shape. We will call the $|q|$ bands of documents as band $a_0$, band $a_1$, ..., band $a_{|q|-1}$. The band $a_k$ consists of documents that do not contain the terms $\{a_{k'} : k' \leq k\}$. For example, documents in band $a_0$ contain all terms in $q$, whereas those in band $a_2$ contain all terms in $q$ except $a_1$ and
$a_2$. There are $df_{a_{a_k+1}} - df_{a_{a_k}}$ documents in band $a_k$, each having the following relevancy score.

$$sim_k^a = \sum_{j=a_v, k<k'<|q|} adf_{a,j} \times idf_j$$

Suppose the last band taken by $G_a(gloss, s_{\text{min}})$ is the band $a_v$. The band $a_{v+1}$ is the first one with $sim_{v+1}^a < s_{\text{min}}$. Denote the sum of the relevancy scores of the documents within bands $a_0, a_1, \ldots, a_u$ as $g_a(v)$. Then $G_a(gloss, s_{\text{min}}) = g_a(v)$. One can calculate $g_a(v)$ by definition as ‘sum of bands’. Alternatively, one can calculate it as ‘sum of strips’, shown in Fig. 5.4. The left diagram illustrates $g_a(v)$ calculation as the sum of relevancy scores of the documents within the bands up to band $a_v$. The vertical axis shows what terms are present in the band.

$$g_a(v) = \sum_{0 \leq k \leq v} sim_k^a \times \text{width of band } a_k$$

$$= \sum_{0 \leq k \leq v} sim_k^a \times (df_{a_{a_{k+1}}} - df_{a_{a_k}})$$

The right diagram illustrates the calculation of $g_a(v)$ by summing $|q|$ strips, where each strip $a_k$ denotes the contribution of the term $a_k$ to the sum $g_a(v)$. For $a_k$ with $k \leq v$, all the $df_{a_{a_k}}$ appearances of the term will be included. For $a_k$ with $k > v$, only $df_{a_{a_v}}$ out of $df_{a_{a_k}}$ appearances will be used. We formulate this with the factor $w_{a,j}$.

$$g_a(v) = \sum_{j \in q} adf_{a,j} \times idf_j \times w_{a,j} \quad \text{where}$$

$$w_{a,j} = \begin{cases} 
df_{a,j} & \text{if } f_a(j) \leq v \\
\text{otherwise. Note that } df_{a_{a_v}} \leq df_{a,j}.
\end{cases}$$

Assume that $G_b(gloss, s_{\text{min}})$ takes up to band $b_u$ and let the term $b_u = j^*$. If the order of terms in $\langle a_k \rangle$ and $\langle b_k \rangle$ are the same, documents in both band $a_u$ and $b_u$ contain the same set of terms from $q$. Since $adf_{a,j} \geq adf_{b,j}, \forall j \in q$,

$$sim_u^a \geq sim_u^b \geq s_{\text{min}}.$$  Hence $G_a(gloss, s_{\text{min}}) \geq g_a(u)$ and it is easy to derive $g_a(u) \geq g_b(u)$ by comparing $w_{a,j}$ and $w_{b,j}$ in the ‘sum of strips’ formula.

However, when the terms in the two sequences are in different order, we need some way to synchronize the bands $a_k$’s and $b_k$’s for the comparison. Append
another artificial term \( a_{|q|+1} = b_{|q|+1} = \omega \) to both sequences. Set \( wdf_{a,\omega} = wdf_{b,\omega} = 0 \), \( df_{a,\omega} = df_{a,\max} \), and \( df_{b,\omega} = df_{b,\max} \). Submit the resultant sequences \( \langle a_k \rangle \) and \( \langle b_k \rangle \) to the algorithm above to get the common subsequence \( \langle c_k \rangle \).

If \( j^* \in \langle c_k \rangle \), take \( v = f_a(j^*) \). Otherwise, take \( v = f_a(x) \) where \( x \) is the first element before \( j^* \) in \( \langle b_k \rangle \) that is also in \( \langle c_k \rangle \). In both cases, we have \( S(\langle b_k \rangle, j^*) \subseteq S(\langle a_k \rangle, a_v) \) by Lemma 5.2. In addition, \( adf_{a,j} \geq adf_{b,j}, \forall j \in q \) and so we have \( sim_a^b \leq sim^a_v \). \( G_a(gloss, s_{min}) \) will take bands at least up to band \( a_v \). This means that \( G_a(gloss, s_{min}) \geq g_a(v) \) and \( G_b(gloss, s_{min}) = g_b(u) \). We conclude the proof by showing \( w_{a,j} \geq w_{b,j} \), \( \forall j \in q \) and therefore \( g_a(v) \geq g_b(u) \).

For those terms \( j \) on or before \( b_u \) in \( aseqb \), \( f_a(j) \leq f_a(a_v) \) by Lemma 5.1. And for those terms \( j \) on or before \( a_v \) in \( \langle a_k \rangle \), i.e. \( f_a(j) \leq f_a(a_v) \), \( w_{a,j} = df_{a,j} \geq df_{b,j} \geq w_{b,j} \). We only need to consider those terms after \( a_v \), which have \( w_{a,j} = df_{a,a_v} \leq df_{a,j} \).

Consider the case \( j^* \in \langle c_k \rangle \). For any \( j \) after \( j^* \) in \( \langle a_k \rangle \), \( w_{a,j} = df_{a,j} \geq df_{b,j} \). If \( f_b(j) < f_b(j^*) \), \( w_{b,j} = df_{b,j} \leq df_{b,j} \). Otherwise, \( w_{b,j} = df_{b,j} \). In both cases we have \( w_{a,j} \geq w_{b,j} \).

Lastly, we are left with the case \( j^* \not\in \langle c_k \rangle \). Recall that \( x = a_v \) is the first element before \( j^* \) in \( \langle b_k \rangle \) that is also in \( \langle c_k \rangle \). By Lemma 5.1, for any \( j \in \langle b_k \rangle \) with \( f_b(x) < f_b(j) \leq f_b(j^*) \), \( f_a(j) < f_a(x) \) and therefore \( w_{a,j} = df_{a,j} \geq df_{b,j} = w_{b,j} \). In particular, \( w_{a,x} \geq w_{a,j} \geq w_{b,j} \) and so for any \( j \) that is both after \( a_v \) in \( \langle a_k \rangle \) and after \( j^* \) in \( \langle b_k \rangle \), \( w_{a,j} = w_{a,a_v} \geq w_{b,j} \). \( \Box \)

54
5.2.2 Sum-based ARD merging

Sum-based ARD merging \( ard_m = \bigoplus_i ard_i \) satisfies \( G(ard_m) = \sum_i G(ard_i) \). It is also desirable that the operation \( \bigoplus \) is commutative and associative to ensure that the order of merging is unimportant. However, we generally do not have idempotence and therefore special consideration has to be taken if the merging ARDs have overlapping coverage.

Suppose that \( ard_i \) is generated by RD aggregation from a collection \( C_i \). The ARD \( ard_m \) generated by RD aggregation from the union of the collections \( \bigcup_i C_i \) is the following (assuming no duplicated documents in \( C_i \)'s):

\[
wdf_{m,j} = \sum_i wdf_{i,j} \quad df_{m,j} = \sum_i df_{i,j} \tag{5.3}
\]

This simplistic summation formula (5.3) satisfies \( G(ard_m) = \sum_i G(ard_i) \) for the IDF family of collection ranking methods. \( G_i(idf) \), \( G_i(cidf) \) and \( G_i(gidf, r_{min}) \) have the general form of \( G_i = \sum_{j \in q} wdf_{i,j} \cdot \delta_j \), where \( \delta_j \) depends on \( idf_j \) for \( j \in q \) and \( r_{min} \) only. This gives a simple proof of the additivity.

\[
G(ard_m) = \sum_{j \in q} wdf_{m,j} \cdot \delta_j \\
= \sum_{j \in q} \left( \sum_i wdf_{i,j} \right) \cdot \delta_j \\
= \sum_i \sum_{j \in q} wdf_{i,j} \cdot \delta_j \\
= \sum_i G(ard_i)
\]

We believe it is not feasible to design sum-based ARD merging for the gGLOSS family that satisfies strict additivity if only one set of \( wdf_{i,j} \) and \( df_{i,j} \) is used in the merged ARD. Nonetheless, one may relax strict equality in the additivity and require only that the merged ARD gives an upper bound of the sum of goodness score of individual ARDs, i.e. \( G_m \geq \sum_i G_i \). With this relaxation, the above summation of \( wdf_{i,j} \) and \( df_{i,j} \) is also satisfactory for \( G_i(weidf, r_{min}) \). Recall that
\( G_i(\text{weidf}, r_{\text{min}}) = G_i(\text{eidf}) \cdot W_i(r_{\text{min}}) \). Since the set \( B \) in computing \( W_i(r_{\text{min}}) \) only depends on \( idf_j \) for \( j \in q \), it is the same for both \( ard_m \) and \( ard_i \). Hence \( W_m(r_{\text{min}}) = \sum_i W_i(r_{\text{min}}) \). This gives the following proof:

\[
G_m(\text{weidf}, r_{\text{min}}) = G_m(\text{eidf}) \cdot W_m(r_{\text{min}})
\geq \sum_i G_i(\text{eidf}) \cdot W_i(r_{\text{min}})
= \sum_i G_i(\text{weidf}, r_{\text{min}})
\]

However, \( G_m(\text{gloss}, s_{\text{min}}) \) computed from ARD \( ard_m \) in (5.3) is not always larger than or equal to \( G_i(\text{gloss}, s_{\text{min}}) \). Two counter examples follow. Let \( Acq(A_m) = \{A_1, A_2\} \) and the query \( q \) be the single term \( j \), and \( wdf_{1,j} = wdf_{2,j} = 2s_{\text{min}} \), \( df_{1,j} = 2 \) and \( df_{2,j} = 4 \). Then \( wdf_{m,j} = 4s_{\text{min}} \), and \( df_{2,j} = 6 \). This results in \( G_m(\text{gloss}, s_{\text{min}}) = 0 < G_1(\text{gloss}, s_{\text{min}}) = wdf_{1,j} \cdot idf_j \). For the second example, let \( wdf_{1,j} = wdf_{2,j} = 2s_{\text{min}} \), \( df_{1,j} = 1 \) and \( df_{2,j} = 3 \). Then \( wdf_{m,j} = 4s_{\text{min}} \), and \( df_{2,j} = 4 \). Now we have \( G_m(\text{gloss}, s_{\text{min}}) = 4 \cdot s_{\text{min}} \cdot idf_j > G_1(\text{gloss}, s_{\text{min}}) + G_2(\text{gloss}, s_{\text{min}}) = wdf_{1,j} \cdot idf_j + 0 = 2 \cdot s_{\text{min}} \cdot idf_j \). In other words, \( G_m(\text{gloss}, s_{\text{min}}) \) computed from the merged ARD may overestimate or underestimate the correct sum.

But we observe that \( G_i(\text{gloss}, s_{\text{min}}) \leq G_i(idf) = G_i(\text{gloss}, s_{\text{min}} = 0) \). \( G_m(idf) = \sum G_i(idf) \) is an upper bound of \( \sum_i G_i(\text{gloss}, s_{\text{min}}) \). This gives a first upper bound by ignoring the parameter \( s_{\text{min}} \).

The formula for max-based merging for \( G_m(\text{gloss}, s_{\text{min}}) \) inspires another merging formula.

\[
wdf_{m,j} = \max_i (wdf_{i,j} / df_{i,j}) \times \sum_i df_{i,j}
\]

\[
df_{m,j} = \sum_i df_{i,j}
\]

Nonetheless, the score computed from the above may sometimes give a bound too loose. Consider the situation in the second example above. The merging
formula would give \( wdf_{m,j} = 8s_{\text{min}} \) and \( df_{m,j} = 4 \). This is even larger than the upper bound \( G_i(idf) \). \( G_m(idf) = 4 \cdot s_{\text{min}} \cdot idf_j < G_m(gloss, s_{\text{min}}) = 8 \cdot s_{\text{min}} \cdot idf_j \). \( G_m(idf) \) will be a tighter bound when \( s_{\text{min}} \) is small. Moreover, we cannot prove the above merging satisfies additivity yet. Therefore, it makes sense to consider both upper bounds in estimating the sum-based goodness score for \( G_i(gloss, s_{\text{min}}) \).

5.3 Building a MISA tree

With proper ARD merging and propagation in the MISA tree, we are assured that searching the tree with the suitable algorithm will return all good enough \( \mathcal{P}_j \)'s regardless of the distribution of resources among \( \mathcal{P}_j \)'s. However, linking \( \mathcal{M}_i \)'s and \( \mathcal{P}_j \)'s recklessly can cause serious efficiency problem in the tree traversal. Moreover, it may also cause degradation in some collection ranking methods — an effect we called *dilution*.

After looking at the adverse effects of reckless tree growing, we will go into some useful ways to grow MISA tree 'healthily'.

5.3.1 Large branch-out

Hierarchical structure has been deployed successfully in many other fields in computer science to tackle scalability problems. Notable examples include distributed directory service like X.500 and the Internet DNS system. Tree traversal in name lookup or domain name resolution typically goes along a single branch on each level. However, one may need to pursue multiple branches in query processing in a MISA tree.

Consider a hypothetical MISA tree with second level MISAs serving the resources in certain geographical region, e.g. \( \text{Acq}(\mathcal{M}_{\text{root}}) = \{\mathcal{M}_{\text{asia}}, \mathcal{M}_{\text{europe}}, \mathcal{M}_{\text{usa}}, \ldots\} \), where \( \mathcal{M}_{\text{root}} \) is the global root MISA. Each geographical \( \mathcal{M}_g \) in turns has acquaintance \( \mathcal{M}_{g,\text{topic}} \) for specific topics like business, entertainment, natural science and social science. (Incidentally, both the geographical and topical layer can be
structured as a hierarchy.) Suppose a user query "distributed resource discovery in digital library" is presented to $M_{\text{root}}$. Thanks to the popularity of the research topic, relevant resources are likely to appear in the natural science MISA of many geographical regions. $M_{\text{root}}$ will return numerous $M_g$'s and thus many geographical MISAs will be consulted in the query processing. In general, much bandwidth will incur if such tree traversal branches to many MISAs on a layer.

5.3.2 The dilution problem

We use only a single set of $wdf_{i,j}$ and $df_{i,j}$ in each ARD, which imposes limit on its expressiveness. There is inevitable information loss when ARDs are merged. This loss is most serious when the merging ARDs are of widely different resource domains. We say that such merge dilutes the indexing information. Dilution results in degradation in quality of collection ranking methods.

Dilution occurs in two different ways. The first way is specific to relative ranking methods, which use a factor $\Delta_j$ to describe the discrimination power of a term $j$ among the collections compared $C_i$'s. It is $cvv_j$ in CVV and $icf_j$ in CORI. This factor changes according to the resource distribution among $C_i$'s. There are hidden problems behind the reliance of $\Delta_j$ on the distribution. As the contents of $C_i$'s become more and more homogeneous (e.g. among geographical MISAs), there will be a tendency for the specificity of a term to dilute. $\Delta_j$ will converge for terms with diverse $idf_j$.

Here is an example to illustrate this. Let the user query be "Perl data structures". Term 1 ('Perl') is likely to have a larger significance in the the query than the other two terms. This is captured in TF-IDF as $idf_1$ is much larger than $idf_2$ and $idf_3$. Now consider the 'geographical-topical' MISA tree in the previous section. It is likely the case that the three terms will have nearly the same $icf_j$. Even worse, the document frequency $df_{i,2}$ and weighted document frequency $wdf_{i,2}$ of term 2 are much larger than those of term 1. This causes $M_{\text{root}}$ to give a high goodness score to a $M_g$ with a lot of documents containing the word 'data'.
This is the consequence of the loss of discrimination power of $icf_j$ and the lack of proper scaling of document frequency by $idf_j$.

$cuv_j$ in $G_i(cuv)$ is less susceptible to this kind of dilution problem. One case in which dilution attacks $cuv_j$ is when the ratio $df_{i,j}/|C_i|$ for each collection is similar. $cuv_{i,j}$'s become approximately the same, resulting in similar $cuv_j$ for terms of very different $idf_j$. Alerted readers might notice that there is no apparent performance degradation for low $P_a$ in the experiments in Chapter 4. One can explain this by noting that we were averaging the results of many queries, and that the document collection used is relatively small. There may only be one or two documents about Perl and so, even when distributed evenly ($P_a = 0$), the factor $\Delta_j$ is not reduced too much.

The second kind of dilution is inherent in lossy ARD merging. Some collection ranking methods like $G_i(gloss)$ and $G_i(weidf)$ use collection specific statistics which is lost in the merged ARD. We generally only have $G^{\max}(ard_m) \geq \max_i G^{\max}(ard_i)$ and $G^{\sum}(ard_m) \geq \sum_i G^{\sum}(ard_i)$. The equality cannot be guaranteed. On the other hand, the equality for sum-based merging holds for the IDF family of collection ranking methods. These methods do not use collection specific statistics and are literally incorporating dilution degradation in the goodness scores. This results in generally poorer performance.

### 5.3.3 Topic specific MISA

In many cases, the large branch-out and dilution problems are caused by acquiring diverse collections in a MISA. These give rise to the necessity of topically specialized MISA.

One can build the MISA tree according to some hierarchical classification scheme, like Library of Congress Classification in library. In other words, each MISA is specialized in a certain topic, with MISAs for various subtopics being his acquaintances. Since a user query is likely to fall into some specific subject served by a few topical MISAs, this helps to reduce much branch-out.
However, the pure hierarchical architecture assumes a tree topology. This cannot model cross-linking of subtopics. An inter-disciplinary subject has resources distributed among several narrower subtopics. We will extend the pure hierarchical architecture to support these in Section 5.6.

Another problem is that the collection of $P_j$ may fall into the domain of more than one leaf MISA. A single ARD from $P_j$ is inherently diluted. This calls for clustering of the resources in a collection before making the ARD. Now $P_j$ can aggregate each cluster and upload the ARD to the suitable MISA.

5.4 Flat architecture

In the hierarchical architecture, all meta ISAs are not equal. They fall into a clear-cut ‘social class’ pyramid in which higher rank citizen orders lower rank citizens to work and extracts information from them — query routing goes downwards while ARD propagation goes upwards. Conflict of interests may arise in the responsibility of running the special class of ISAs.

Flat architecture introduces a different arrangement of rights, responsibility and cooperation. Each participant is more or less of equal capability. Examples of flat architectures abound in the physical world. Consider the research community example again. Some researchers are very knowledgeable. (A PISA with extensive RD base.) Some are very sociable in the research community and know the various subfields and the pertinent researchers well. (A MISA with a lot of acquaintance.) Nevertheless, a research newcomer usually does not go through the hierarchy top down to search for information. (Human beings are much more liable to scalability problems.) Instead, many of us roam among our peers and colleagues following their recommendation to knowledgeable persons to the topic involved.

We will first give a somewhat restricted version of the flat architecture in this section. Since this pure flat architecture is partly inspired by the work in [33], we will borrow the name Cordia and call the abstract ISAs involved Cordia ISAs, or CISAs for short. A set of CISAs $C_i$’s connected in a certain topology is known as a
CISA mesh accordingly. See Fig. 5.5. As ISAs in the abstract DRD model, CISAs are connected to each other through the acquaintance relationship. Unlike in the hierarchical architecture, no strict layered structure is assumed. Loops are allowed in the CISA mesh. In particular, two CISAs may be mutual acquaintances and the acquaintance relationship is literally bi-directional. This means that ARDs are transferred to and fro between CISAs. For simplicity, we will assume all acquaintance relation to be bi-directional in the pure flat architecture.

The CISA mesh cooperates with the IUA to find out good enough \( \mathcal{P}_i \)'s. Each \( \mathcal{C}_i \) is coupled with a unique ISA \( \mathcal{P}_i \), and serves as his representative in the CISA mesh. In terms of the abstract DRD model, \( \mathcal{P}_i \) is an acquaintance of \( \mathcal{C}_i \) and uploads an ARD of his collection. However, this ARD is different in nature to those ARDs transferred within the CISA mesh. The former has locator to \( \mathcal{P}_i \) (most likely a primary ISA) which may be a result of collection selection, while the latter guides the IUA to roam around the mesh. Hence, we reserve the notion \( \text{Acq}(\mathcal{C}_i) \) to the acquaintance within the CISA mesh from now on in discussing flat architecture and its variations.

The ARD \( \text{ard}_c \) transferred from \( \mathcal{C}_i \) to his acquaintance derives from the ARD \( \text{ard}_p \) uploaded by \( \mathcal{P}_i \). A locator to himself (of the form \texttt{cisa://cisa-name@which-community} which is piggybacked to \( \text{ard}_p \). The scheme instructs the IUA to follow suitable protocol in navigating the CISA mesh. In words, the ARD uploaded by \( \mathcal{C}_i \) consists of a locator to \( \mathcal{C}_i \) and \( \mathcal{P}_i \) and the metadata of the ARD uploaded from \( \mathcal{P}_i \). Because each \( \mathcal{C}_i \) uploads only this single ARD, we refer to this as the ARD of \( \mathcal{C}_i \). Each \( \mathcal{C}_i \) keeps in his RD base the ARDs of himself and his acquaintances.

Suppose the IUA submits a query to a certain CISA \( \mathcal{C}_a \) and \( \mathcal{C}_b \in \text{Acq}(\mathcal{C}_a) \). Since the ARD uploaded by each \( \mathcal{C}_b \) only derives from that of \( \mathcal{P}_b \), \( \mathcal{C}_a \) has little knowledge about other CISAs beyond his acquaintance. The flat architecture employs an alternative method to provide implicit knowledge about indirect acquaintance. Acquaintance relationship is established based on the similarity or coherency of the ARDs of two CISAs. "Birds of a feather flock together." If one
Figure 5.5: The pure flat architecture: CISA mesh and CISAs

finds that the collection in $\mathcal{P}_b$ is promising, then it is likely that the collection in $\mathcal{P}_c$ for $\mathcal{C}_c \in \text{Acq}(\mathcal{C}_b)$ will also be. Metrics for coherency will be discussed in Section 5.5.

IUA navigation of the CISA mesh follows a best-first strategy. This is described in the pseudocode $\text{IUA::NavigateCisaMesh}$ in Fig. 5.6. At any time instance, the IUA keeps a history of the visited CISAs and the ARDs from their RD bases. This is equivalent to the ARDs of all the visited CISAs and their acquaintances. For example, in the CISA mesh in Fig. 5.5, after visiting $\mathcal{C}_3$, the IUA would have the ARDs of $\mathcal{C}_i$ and $\mathcal{P}_i$, $i = 2, 3, 4, 5$. At each step, these collections are ranked by certain collection ranking method to find the best CISAs to route the query to. The process stops when the best CISA has already been visited. Notice that if absolute collection ranking is used, the IUA only needs to keep the locator to CISA or PISA and may leave the computation of the goodness score to the CISA being visited. Collection ranking done at the IUA only results in different ranking than that at CISAs for relative goodness scores, as for example, $G_i(cuv)$ in the original Cordia paper.

The call $\text{NavigateCisaMesh}(\mathcal{C}_\text{start}, q, n, \cdot)$ will stop eventually as there are a finite number of CISAs, each of them visited at most once. However, the algorithm does not guarantee that the best $n$ $\mathcal{P}_i$'s among the set of PISAs connected to the CISA mesh will be returned even if the mesh is a connected graph. The
function CISA::ProcessQuery (self, q)
begin
   // return the ARDs of myself and my acquaintances
   // all these ARDs are dual-homed: they have both a locator to CISA and PISA
   // an optimization. only return statistic for terms in q
end

function IUA::NavigateCisaMesh (C_start, q, n, collrank-param)
begin
   // Q is a list of CISAs in descending order of goodness
   Q := empty list
   C_next := C_start
   while (visited(C_next) == false)
      L := CISA::ProcessQuery(C_next, q)
      set visited(C_next) to true
      append each C_i in L to Q, duplicates discarded
      perform collection ranking on C_i in Q
      C_next := the highest rank CISA in Q
   return the top n in Q
end

Figure 5.6: Navigation algorithm for pure CISA mesh

Navigation algorithm will stop at ridge of relatively low goodness scores. Fig. 5.7 shows a mesh of CISAs connected in a rectilinear grid. The goodness scores w.r.t. the query q of the ARD of some CISAs are shown. An empty cell means that the CISA there has goodness score smaller than the neighboring cells with goodness scores shown. Suppose the IUA starts navigating the CISA mesh at C_a. Because a CISA only knows about direct acquaintance, but not that of the indirect ones, the navigation will not cross the low goodness boundary and therefore fails to recognizes the high goodness score CISA in the middle. Even worse, CISA mesh navigation sometimes can be trapped in a patch of ‘not so good’ nodes, as in the case if we start navigation at C_b. Since connected CISAs are coherent, the CISAs in the group at C_b may be equally ‘bad’ for the query. This illustrates the importance of connecting relevant CISAs together and choosing C_start wisely.
These topics will be examined in the next section.

Figure 5.7: Patches of high goodness score CISAs separated by low score ridges

5.5 Weaving a CISA mesh

Without propagation of merged ARD, proper acquaintance establishment becomes very important to the effectiveness of the CISA mesh. We refer to the action of establishing acquaintance as weaving a CISA mesh. Literatures on flat architectures often apply some similarity measure between collections as a guide in mesh weaving. Some examples and their pitfalls are examined first. Then we will analyze some requirements in proper weaving to ensure completeness of collection selection.

5.5.1 ARD coherency

In weaving the CISA mesh, we would like that CISAs hosting relevant resources to a user query $q$ are linked together. Because the IUA can only go among CISAs along their acquaintance relationship, proper connection of ‘relevant’ $C_i$’s is the most basic requirement. This is usually done by identifying $C_i$’s of similar topics and linking them together. One can of course perform these manually. Alternatively, we would like to automate this by using a measure of the similarity or coherency between the ARDs of two CISAs. We denote this function as $coh(ard_a, ard_b)$. The CISA mesh will be weaved such that $C_a$ and $C_b$ with high $coh(C_a, C_b)$ are linked together. One may take the cosine measure between the
$wdf_{i,j}$ vectors of the ARDs. This measure ranges from 0 to 1, where 1 denotes the highest coherence. Another proposal in [33] uses the summation of $cuv_j$ of the union of terms appearing in two collections as a measure of how different two collections are.

$$\text{cossim}(ard_a, ard_b) = \frac{\sum_j wdf_{a,j} \cdot wdf_{b,j}}{\sqrt{\sum_j wdf_{a,j}^2} \cdot \sqrt{\sum_j wdf_{b,j}^2}}$$

$$\text{scvv}(ard_a, ard_b) = \sum_{j, d_{a,j} > 0 \land d_{b,j} > 0} cuv_j$$

These measures suffer two shortcomings, nonetheless. First, the two ARDs may be very different according to the above measures but still give comparative goodness scores for the query $q$. For example, consider the collection of computer science research literature done in United States and the collection of natural science (which includes computer science) in Asia. The first collection is more focused in topic specificity. A gross similarity measure may rank the two collections to be incoherent even if they have comparative goodness scores for a query on computer science. This suggests adding a parameter $T$ as some representation of a topic. We will use a set $T$ of common terms in a certain topic. Furthermore, it is good to give larger weight to those significant terms in the cosine measure. We obtain (5.4) after these adjustments, where $\delta_j$ is a weight for term $j$. Incidentally this measure is also good for identifying similar collections for merging in the hierarchical architecture. If the ARDs from a set of MISAs are mutually highly coherent, or if the ARDs are highly coherent to an ARD predefined by the upstream MISA as his topic, then the merged ARD will also likely to be coherent.

Such ARD will suffer less from dilution.

$$\text{coh}(ard_a, ard_b, T) = \frac{\sum_{j \in T} wdf_{a,j} \cdot wdf_{b,j} \cdot \delta_j}{\sqrt{\sum_{j \in T} wdf_{a,j}^2} \cdot \sqrt{\sum_{j \in T} wdf_{b,j}^2}} \quad (5.4)$$

However, the above measures are not suitable for weaving CISA mesh because they ignore the magnitude of the goodness score of the compared ARDs w.r.t.
a user query. As an example, assume three artificial ARDs of nearly perfect mutual coherency. The $wdf_{i,j}$ vectors may be approximately a scalar multiple of each other. But their goodness scores may be very different, say, $G(C_a, q) = 100$, $G(C_b, q) = 20$ and $G(C_c, q) = 80$. If they are connected as $C_a \leftrightarrow C_b \leftrightarrow C_c$, the IUA will be trapped at $C_a$ and cannot cross the low goodness $C_c$ to reach $C_a$. Another minor deficit of the above measure is that the highest $\text{coh}(\cdot, \cdot, T)$ is not a constant, but depends on $T$.

5.5.2 An ideal weaving scheme

We now consider some necessary conditions in weaving the CISA mesh for a navigation algorithm that only uses local information of the CISAs visited. Recall that each CISA receives ARDs of the curiosity of his acquaintances only. When the IUA visits $C_i$, he obtains the ARDs of $C_i$ and $Acq(C_i)$. Generally, let $C_{\text{visited}}$ be the set of CISAs visited by the IUA, and $C_{\text{unvisited}}$ be the set of CISAs not yet visited but who are acquaintances of a visited CISA and therefore their ARDs are known to the IUA. Then at any moment the IUA makes the decision of query routing based on his knowledge of $C_{\text{visited}}$ and $C_{\text{unvisited}}$. The IUA has to determine whether he should stop, or if not, which of $C_{\text{unvisited}}$ should be visited next. This decision should be based on the information on hand only.

Let us first consider the absolute threshold $g_{\text{min}}$ for collection selection. A CISA $C_i$ is good enough if $G(C_i) \geq g_{\text{min}}$. Suppose the navigation starts at $C_{\text{start}}$ and at a certain moment, IUA has visited the set of CISAs $C_{\text{visited}}$. Consider the following simple rule: Stop if $G(C) < g_{\text{min}}, \forall C \in C_{\text{unvisited}}$. Otherwise, visit $C_{\text{best}} \in C_{\text{unvisited}}$ of largest $G(\cdot)$. This is encoded in the revised CISA mesh navigation algorithm in Fig. 5.9. Such navigation will be complete if and only if for any good enough $C_i$, there is a sequence of good enough CISAs connecting $C_{\text{start}}$ and $C_i$. It is desirable for the CISA mesh to have this property for all values of $g_{\text{min}}$. This seems to suggest a configuration of descending goodness scores from the highest goodness CISA. Consider the example in Fig. 5.8. Suppose the IUA
starts at $C_1$ with $g_{\text{min}} = 8$, and let $G(C_1) = 8$. The above navigation scheme will first drive the IUA towards the highest goodness CISA $C_{\text{best}}$ with $G(C_{\text{best}}) = 10$. Then the IUA will visit surrounding CISAs of lower and lower goodness scores until he reaches the boundary of $G(\cdot) < g_{\text{min}}$. If the navigation starts outside the boundary, say, at $C_2$, then the IUA may not be driven towards the highest goodness score node. This may be resolved by ensuring that the best CISA in $C_{\text{unvisited}}$ will always be visited.

![Diagram of Coherent CISAs with Descending Goodness Scores](image)

Figure 5.8: A group of coherent CISAs with descending goodness scores

The above goodness weaving scheme needs undesirable global coordination. The behavior of the IUA actually hints at a less strict weaving scheme. In order to drive the IUA to $C_{\text{best}}$ first, each $C_i$ should try hard to maintain an acquaintance $C_a \in \text{Acq}(C_i)$ with $G(C_a, q) > G(C_i, q)$. This ensures that wherever IUA starts, it will be driven to $C_{\text{best}}$. In fact, this also enables the IUA to visit every good enough CISAs thereafter. For an arbitrary good enough CISA $C_x \neq C_{\text{best}}$, he will have an acquaintance who is better than him. This traces a sequence of good enough CISAs recursively to $C_{\text{best}}$. The IUA can reach $C_x$ along this path. We phrase the above requirement in the following principle.

*Aggressive principle in CISA mesh weaving*: Given a query $q$. When he first joins a CISA mesh or when his acquaintances change (old acquaintances disconnected, or their ARDs updated), a CISA $C_a$ should
try to maintain that at least one $C_b \in Acq(C_a)$ has $G(C_b, q) > G(C_a, q)$
for a query $q$ he is interested in.

It is noteworthy that the weaving scheme also ensures completeness for collection selection with relative parameter $n$. Suppose the user wants the top $n$ collections. If at one moment, the IUA has visited $C_{visited}$, and finds that any $C_i \in C_{unvisited}$ has $G(C_i) < G(C_w)$ for the worst $C_w \in C_{visited}$, then he does not need to pursue $C_{unvisited}$ anymore to find other good $C_i$ outside. Assume to the contrary that such a good enough CISA $C_x$ exists outside the boundary $C_{unvisited}$. By the aggressive principle, there is a path of mutual acquaintances of better and better goodness scores from $C_x$ towards $C_{best}$. This path will intersect $C_{unvisited}$ at a certain $C_y$. $G(C_x) > G(C_w)$ leads to $G(C_y) > G(C_w)$, a contradiction.

The above weaving scheme is especially favorable because it only uses local information. When a new CISA joins the CISA mesh, it only needs to find an existing member in the CISA mesh that is better than him. Similarly, when the ARD of a CISA changes, only he and his acquaintances have to check whether they still have an acquaintance better than themselves.

5.5.3 Approximating the ideal weaving scheme

The aggressive principle ensures that CISA mesh navigation starting at any $C_{start}$ will reveal all good enough $C_i$'s if the mesh is weaved accordingly. However, the weaving scheme needs to compare $G(C_i, q)$'s, which may be very different for different queries $q$'s, and one cannot hope for weaving the mesh for all possible queries. In general, one cannot guarantee completeness for CISA mesh navigation.

An approximation is to use a topic term set $T$ to replace a query in the topic. A CISA that is interested in the topic will use $G(C_i, T)$ in the maintenance of his acquaintances. He may acquire different acquaintances for different $T$'s. In this sense, the acquaintances of $C_i$ are typed.

Using $T$ instead of various $q$'s in mesh weaving is motivated by the following assumption. If coh($C_a, C_b, T$) is very high, and $G(C_a, T) > G(C_b, T)$, then it is
function CISA::ProcessQuery (self, q, collsel-param, collrank-param)
begin
    // perform collection ranking of the ARDs in my RD base
    // that is, the ARD of myself and my acquaintances
    // all these ARDs are dual-homed: they have both a locator to CISA and PISA
    return the locators of the ARDs with their scores
end

function IUA::NavigateCisaMesh (C_{start}, q, collsel-param, collrank-param)
begin
    // collsel-param consists of n and g_{min}.
    // Q is a list of CISAs in descending order of goodness
    Q := empty list
    C_{next} := C_{start}
loop
    L := CISA::ProcessQuery(C_{next}, q, n, collrank-param)
    set visited(C_{next}) to true
    insert each C_i in L to Q, in descending order of G(\cdot), duplicates eliminated
    if best n CISAs in Q are all visited
        exit loop
    else
        C_{next} := the best unvisited CISA in Q
    endif
end
return the top n in Q
end

Figure 5.9: Revised navigation algorithm for pure CISA mesh

very likely that for q \subset T, G(C_a, q) > G(C_b, q). This holds for the IDF family of
collection ranking methods. A very high coh(C_a , C_b , T) implies that the two
vectors wdf_{a,j} and wdf_{b,j}, j \in T are approximately scalar multiple of each other.
Then for most j \in T, wdf_{a,j} > wdf_{b,j}. This gives G(C_a , q) > G(C_b , q) for the
collection ranking methods G(idf) and G(eidf). They use the generic formula
\sum_{j \in q} wdf_{i,j} \cdot \delta_j where \delta_j is independent of the collections. The same \delta_j should
also be used in coh(\cdot, \cdot, \cdot, \cdot). Using G(gidf, r_{min}), G(gloss, s_{min}) and G(weidf, r_{min})
in the calculation of G(C_a , T) is not as meaningful because of their assumption
of the distribution of query terms in a collection. One can only use G(idf) and
$G(eidf)$ to weave the CISA mesh.

However, not all CISAs are interested in all topics. Even if the assumption mentioned above works, navigation starting at a CISA not interested in answering topic $T$ will likely be trapped. A proper starting point for navigation is still very important. A user interested in certain topics can either set his IUA to start at CISAs interested in the same topics, or he can use a special agent for locating the starting point. A MISA tree will be proposed to perform this in Section 5.7.

5.6 Hybrid architectures

The hierarchical and flat architectures as described in the previous sections are pure and reside on two extremes of a spectrum of distributed collection selection subsystems based on ARD merging and query routing. In this section, we first contrast the two pure architectures. Then we will describe them using a common vocabulary. This serves as an elaboration of the abstract ISA mesh in the abstract DRD model. Then we are able to evolve one architecture towards the other to get two extended versions. The extended hierarchical architecture is presented first, along with the issues of overlapping coverage in ARD merging and query processing. We will see the importance of both max-based and sum-based ARD merging in this architecture. Afterwards, we add ARD propagation to the pure flat architecture. This architecture features similar issues of overlapping coverage in ARDs, plus some of its own.

5.6.1 Comparing and reconciling the hierarchical and flat architectures

Nodes in a MISA tree are grouped into disjoint layers in which MISAs on lower layers upload ARDs to those on upper layers. There are no loops. The other side of the coin is that query routing is only done from upper to lower layers. Because of this arrangement, one does not need to remember the previously visited MISAs
in traversing the MISA tree. On the other hand, navigation in the unstructured CISA mesh often encounters loop and therefore needs to remember the visited CISAs.

There is a big difference in query processing in the two architectures. The collection selection process in MISA tree is complete. All $P_j$’s that are good enough are returned at the end of the tree traversal. CISA mesh lacks this feature. It stems from the difference in ARD propagation. In MISA tree, ARD propagation is cumulative, meaning that a MISA merges all ARDs in his RD base for uploading to his upstream MISA. This property, together with proper max-based ARD merging, ensures that one can start query processing at the root of the tree, always trusting that within a certain number of query routing, he will end up with the good enough primary ISAs. The prohibition of loops and the layered structure also ensures termination.

CISA mesh, on the other hand, relaxes these requirements. The navigation in CISA mesh is generally not complete, although one can approximate the ideal weaving scheme by adding acquaintance links between coherent CISAs. Table 5.2 summarizes the differences between the pure hierarchical and flat architectures.

Diverse as they can be, the pure hierarchical and flat architectures can be described using the common vocabulary of indirect acquaintance, curiosity ARD and interest ARD. The discussion below applies to both MISA and CISA, and so we use generic abstract ISA $A_i$ in the formulation.

Given a set of primary ISA $P_j$’s. We are to define a collection ranking subsystem using a set of abstract ISA $A_i$’s. In order to be recommended by the abstract ISA mesh, each $P_j$ uploads an ARD of his RD base to one or more $A_i$’s. We call the set of $P_j$’s that upload ARDs to $A_i$ his curiosity, denoted as $\text{curio}(A_i)$. The ARDs uploaded by $P_j$’s are foreign to the abstract ISA mesh. The locators in these ARDs are typically locators to primary ISA $P_j$. (They can also be an ISA in another abstract ISA mesh. More details will be given in Section 5.7.) These ARDs are special to $A_i$ and are termed curiosity ARDs of $A_i$. Since we assume that each $P_j$ uploads only a single ARD, we also denote curiosity ARDs
<table>
<thead>
<tr>
<th></th>
<th>Hierarchical</th>
<th>Flat</th>
</tr>
</thead>
<tbody>
<tr>
<td>topology</td>
<td>MISA tree is a directed tree. Leaf MISAs have PISAs as acquaintances.</td>
<td>CISA mesh is a network. Bi-directional acquaintance links between CISAs. CISAs and PISAs are coupled in pairs.</td>
</tr>
<tr>
<td>indexing</td>
<td>cumulative, upward propagation</td>
<td>non-cumulative, no ARD merging</td>
</tr>
<tr>
<td>navigation</td>
<td>starts at the root. Downward query routing to the leaf of the MISA tree</td>
<td>starts at a CISA interested in the same topic. Roaming until a low goodness score ridge is hit.</td>
</tr>
<tr>
<td>coverage</td>
<td>ARDs of MISAs on the same level do not overlap</td>
<td>ARDs of any two CISAs may overlap</td>
</tr>
<tr>
<td>information</td>
<td>ARD merging and propagation</td>
<td>acquaintance linking of coherent CISAs</td>
</tr>
<tr>
<td>about indirect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>acquaintance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Comparison of pure hierarchical and flat architectures

as curio$(\mathcal{A}_i)$ when there is no ambiguity. We reserve the notion $Acq(\mathcal{A}_i)$ to those abstract ISAs connected to $\mathcal{A}_i$ in the abstract ISA mesh, and separate this from curio$(\mathcal{A}_i)$.

We define the indirect acquaintance of an abstract ISA, denoted as $Acq_n(\mathcal{A})$, to be the set of abstract ISAs reachable from $\mathcal{A}$ within $n$ acquaintance links in the abstract ISA mesh. For example, $Acq_0(\mathcal{A}) = \mathcal{A}$, $Acq_1(\mathcal{A}) = \{\mathcal{A}\} \cup Acq(\mathcal{A})$, and $Acq_2(\mathcal{A}) = Acq_1(\mathcal{A}) \cup \bigcup_{B \in Acq(\mathcal{A})} Acq(B)$.

We have been using the agent name $\mathcal{A}$ itself or phrase like 'the ARD of $\mathcal{A}$' to represent the ARD he uploads. This will create difficulty in notation in more general architectures. Therefore, we introduce the notion of interest ARD of an abstract ISA. An interest ARD of $\mathcal{A}$ is an ARD with locator pointing to $\mathcal{A}$. The metadata of the ARD derives from the curiosity ARDs of $\mathcal{A}$ and that of his (indirect) acquaintances. (This justifies our choice of the figurative terms.) The interest ARD is parameterized. In this thesis, we only use the depth of the indirect acquaintances taken. Note that we use the union operation on the
curio(·)'s to get rid of duplicates in the formula.

\[
\text{interest}(\mathcal{A}_i, \text{depth} = n) = \bigoplus_{\mathcal{P}_j \in \text{curio}_n} \mathcal{P}_j, \quad \text{curio}_n = \bigcup_{\mathcal{B} \in \text{Acqu}(\mathcal{A}_i)} \text{curio}(\mathcal{B}) \quad (5.5)
\]

With this notation, we can now formulate the pure hierarchical and flat architectures uniformly. In addition, we can use the general navigation algorithm \texttt{IUA::NavigateAbstractISAMesh} in both architectures.

The pure hierarchical architecture is a set of abstract ISAs \( \mathcal{M}_i \)'s with the acquaintance relationship forming a directed tree. Except for the leaf nodes, \( \text{curio}(\mathcal{M}_i) = \emptyset \). For leaf nodes, \( \text{curio}(\mathcal{M}_i) \subset \{ \mathcal{P}_j \text{'s} \} \) and these curiosities are disjoint. All indirect acquaintances are taken in the computation of interest ARDs, i.e. \( \mathcal{M}_i \) uploads \( \text{interest}(\mathcal{M}_i, \text{depth} = \infty) \). The \texttt{IUA::NavigateAbstractISAMesh} algorithm in Fig. 5.11 can be used instead of \texttt{IUA::TraverseMisATree}, although the checking for visited \( \mathcal{M}_i \) is not necessary.

The pure flat architecture consists of a set of abstract ISAs \( \mathcal{C}_i \). The special coupling relationship between \( \mathcal{C}_i \) and \( \mathcal{P}_j \) is now rephrased with the curiosity notion. Each \( \mathcal{C}_i \) has a single \( \mathcal{P}_j \) as his curiosity, and these curiosities are disjoint. The CISAs are connected according to the coherency of their curiosity ARDs. Each \( \mathcal{C}_i \) uploads \( \text{interest}(\mathcal{C}_i, \text{depth} = 0) \), i.e., the metadata of this ARD is the same as that of his single curiosity ARD. The generalized algorithm \texttt{IUA::NavigateAbstractISAMesh} in Fig. 5.11 can be used instead of \texttt{IUA::NavigateCisaMesh}. There is a minor technical change though. Both algorithms will visit the same set of CISAs in the same order. However, when the navigation stops, the pairs of \( \mathcal{C}_i \) and \( \text{curio}(\mathcal{C}_i) = \{ \mathcal{P}_i \} \) in \texttt{IUA::NavigateCisaMesh} are split into \( L_{\text{curio}} \) and \( L_{\text{int}} \) in \texttt{IUA::NavigateAbstractISAMesh}. The former holds \( \mathcal{P}_i \) who is the curiosity of a visited \( \mathcal{C}_i \), while the latter holds the interest ARDs of unvisited acquaintance \( \mathcal{C}_{\text{unvisited}} \) of visited CISAs. In the pure flat architecture, these ARDs have the same metadata as a curiosity \( \mathcal{P}_i \). However, in general, they can be a merged ARD of several curiosity ARDs and therefore have to be resolved by visiting \( \mathcal{C}_{\text{unvisited}} \). This step can be bypassed in the pure flat
architecture. To mimic the exact behavior the IUA::NavigateCisaMesh, one can change the last statement of IUA::NavigateAbstractISAMesh to return the first 
$n$ of the combined list of $L_{\text{curio}} + L_{\text{int}}$.

Also notice that the aggressive principle is satisfied in the hierarchical architecture observing optimistic principle in ARD merging. The upstream acquaintances are always better than a MISA for all queries. In other words, the principle that guarantees completeness in weaving the CISA mesh is a general case of the principle that guarantees completeness in ARD merging in a MISA tree.

function AbstractISA::ProcessQuery ($q$, collsel-param, collrank-param) 
begin
    // collsel-param consists of $n$ and $g_{\text{min}}$.
    // compute the goodness scores of the ARDs in my RD base
    // using the parameter 'collrank-param'.
    // return good enough curiosity ARDs with their scores
    // and interest ARDs of my acquaintances with their scores.
    // if IUA insists to do ranking himself, just return the ARDs,
    // else return the locators in the ARDs.
    $L_{\text{curio}} := \text{top } n \text{ of my curiosity with } G_{\max} (\cdot) \geq g_{\text{min}}$
    $L_{\text{int}} := \text{the interest ARDs of my acquaintances}$
    return $L_{\text{curio}}$ and $L_{\text{int}}$
end

Figure 5.10: Query processing at an abstract ISA

5.6.2 Hierarchical architecture with overlaps

The characteristics of hierarchical architecture is an acyclic topology. Because there are no loops, ARD merging can be cumulative and ARDs propagate from ‘downward’ to ‘upward’. In the extended hierarchical architecture, we will relax the strict layered structure in pure hierarchical architecture and introduce overlapping coverage.

The extended hierarchical architecture is formulated in our model as follows. The topology of the participating abstract ISAs, also called meta ISAs, is a di-
function IUA::ChooseNextIsaToVisit ()
begin
// choose the next abstract ISA in Q to visit
// in addition to choosing the one with largest $G^\text{max}(\cdot)$,
// one can also consider $G^\text{min}(\cdot)$ and the coverage signature
// in the ARD to handle overlap case better
if $Q$ is empty then return null
// choose the next best abstract ISA $A_b$ in $Q$ to visit
if $L_{\text{good}}$ is not empty
  $P_w :=$ the worst PISA in $L_{\text{good}}$
  if $G(A_b) \leq G(P_w)$ then return null
  extract $A_b$ from $Q$
end return $A_b$
end

function IUA::NavigateAbstractIsaMesh ($A_{\text{start}}$, $q$, collsel-param, collrank-param)
begin
// collsel-param consists of $n$ and $g_{\text{min}}$.
// $Q$ is a list of abstract ISA to be visited.
// visited($A_i$) == false and $G^\text{max}(A_i) \geq g_{\text{min}}$ for $A_i \in Q$.
// $L_{\text{good}}$ is a list of good enough $P_j$ collected so far. Maximum length is $n$
// Both lists are in descending order of goodness
$Q :=$ empty list
$L_{\text{good}} :=$ empty list
append $A_{\text{start}}$ to $Q$
loop
  $A_{\text{next}} :=$ IUA::ChooseNextIsaToVisit()
  if $A_{\text{next}}$ == null then exit loop
  ($L_{\text{curio}}$, $L_{\text{int}}$) :=
  AbstractISA::ProcessQuery($A_{\text{next}}$, $q$, collsel-param, collrank-param)
  set visited($A_{\text{next}}$) to true
  append each $P_j$ in $L_{\text{curio}}$ to $L_{\text{good}}$,
  discard duplicates, sort by goodness score, maintain maximum length $n$
  append each $A_i$ in $L_{\text{int}}$ with visited($A_i$) == false to $Q$,
  discard duplicates and sort by goodness score
  // only do this if IUA like to do ranking himself
  perform collection ranking on $A_i$ in $Q$
  and sort $Q$ by goodness, taking consideration of $g_{\text{min}}$ and $n$
end return $L_{\text{good}}$
end

Figure 5.11: Navigation algorithm for abstract ISA mesh
rected acyclic graph. Each MISA $\mathcal{M}_i$ may have his own curiosity $\text{curio}(\mathcal{M}_i) \subset \{\mathcal{P}_j\}'s$. Each MISA $\mathcal{M}_i$ merges the ARDs received from his downstream acquaintances and his own curiosities and uploads a single merged ARD to his own upstream acquaintances. ARD propagation is cumulative and the ARD uploaded by $\mathcal{M}_i$ is $\text{interest}(\mathcal{M}_i, \text{depth} = \infty)$. (We will also use the agent name $\mathcal{M}_i$ to denote the ARD when the depth is not emphasized.) Unlike the pure hierarchical architecture, there is no strict layering, and any two $\text{Acq}(\mathcal{M}_i)$ may overlap. Because the mesh is acyclic, there are some ‘top level’ MISA’s with no upstream acquaintance.

![Diagram of extended hierarchical architecture](image)

Figure 5.12: The extended hierarchical architecture: acyclic mesh with overlaps

We now inspect the effect of overlapping coverage. Consider the MISA mesh in Fig. 5.12. The departure from a directed tree to the more general directed acyclic graph brings two inter-related issues: how to handle duplicates in ARD merging, and how to choose the next MISA to visit in navigation.

We first consider max-based ARD merging. In the pure hierarchical architecture, the coverage of MISA’s on the same layer do not overlap. There is a unique path from the root to an arbitrary $\mathcal{P}_j$ and therefore in merging ARDs from acquaintances, a MISA does not have to worry about duplicates. Hence $\mathcal{M}_i$ can compute his interest ARD $\text{interest}(\mathcal{M}_i, \text{depth} = \infty)$ simply as
\( \bigoplus_{A \in \text{Acq}(M_i)} \) interest(\( A, \text{depth} = \infty \)). However, some \( P_j \) may be accessible through multiple paths from \( M_i \) in the extended hierarchical architecture. For example, in merging the ARDs from \( M_3 \) and \( M_4 \), \( M_1 \) is risking any undesirable effect of double counting \( P_2 \). Fortunately, max-based ARD merging \( \bigcirc \) is idempotent and therefore \( M_3 \bigcirc M_4 = \bigcirc_{j=1..4} P_j \). We only need to add a checking of visited MISA's in the IUA::NavigateMisaTree to handle any duplicate \( M_i \) and \( P_j \) returned by MISA::ProcessQuery. Then the navigation algorithm will work correctly for this architecture. It will not loop but is still complete. It can also prune the fruitless branches confidently.

However, the simple choice of the next MISA with the largest \( G^{\text{max}}(\cdot) \) in navigation might not work as well. Imagine that \( P_1, P_4 \) and \( P_5 \) are good enough. Then the three \( M_i, i = 3, 4, 5 \) are good enough by the optimistic principle. How should the IUA choose when \( M_1 \) return all three to him? A solution to minimize query routing is to choose \( M_3 \) and \( M_4 \). The IUA needs some heuristics to choose 'wisely' among MISAs with overlapping coverage.

We propose two heuristics for choosing a good set. First, it would be very useful if IUA can make an educated guess of the overlaps between \( M_i \)'s. Our solution uses bitstring signature similar to signature-file system [30]. A bitstring coverage signature is assigned to each \( P_j \) and included in the ARD he uploads. When a MISA merges these ARDs from downstream \( P_j \)'s, he superimposes the coverage signatures and attaches the result to the merged ARD he himself uploads. This is done recursively at each \( M_i \). In effect, each ARD now contains a signature giving hints to its coverage. The IUA can make a pretty good guess of the overlaps of the coverage by comparing their signatures. The coverage signature is also favorable because the superimposing operation is also idempotent.

In addition, the goodness score \( G^{\text{sum}}(\cdot) \) also provides useful hints. When the coverage of two MISAs have much overlaps (perhaps as guessed from the signatures), it might be a good idea for the IUA to first visit the one with a larger \( G^{\text{sum}}(\cdot) \), since it may already cover much of the good enough \( P_j \) in the other one.
Sum-based ARD merging $\oplus$ is not idempotent, however. A plain merging of $M_3$ and $M_4$ will double count $wdf_{i,j}$ and $df_{i,j}$ in $P_2$. This results in an overestimated $G^{sum}(M_1)$. $M_1$ may adjust the ARD by somehow ‘subtracting’ the double counted $P_j$ by inquiring $P_j$ for his ARD. Alternatively, one may require an individual MISA also uploads the curiosity ARDs he receives without merging. In this way, the ARD uploaded by $P_j$ is transferred intact to each $M_i$ that is reachable through the acquaintance relationship. Sum-based ARD merging is done at each $M_i$ with the original curiosity ARDs from $P_j$. This process can use less bandwidth by carefully selecting which acquaintance to upload which $P_j$. For example, $M_1$ in Fig. 5.11 may request $M_3$ and $M_5$ to upload an intact ARD from $P_1$ and $P_5$ respectively in addition to their own interest ARDs. $M_4$ only needs to upload his interest ARD $P_2 \oplus P_3 \oplus P_4$. Now $M_1$ himself can upload $P_1 \oplus M_4 \oplus P_5$.

We incorporate the above changes to the general algorithm IUA::NavigateAbstractISAMesh in Fig. 5.11. Query processing typically starts at one of the top level MISAs.

A major application of the extended hierarchical architecture is to implement topic specific MISA. As explained earlier, reckless linking of acquaintances can cause serious efficiency problem in query processing. This can be avoided if the domain of MISAs coincides with topical clustering of user queries. However, a pure hierarchy is usually not sufficient to model the topic-subtopic relationship of online resources. The extended hierarchical architecture is more suitable in this case.

### 5.6.3 Flat architecture with ARD propagation

In the hierarchical architecture, ARD merging and propagation inform MISAs on higher level information about good enough $P_j$’s underneath. This feature is missing in the pure flat architecture. We now attempt to extend the flat architecture by adding ARD propagation. An unresolved issue of the flat architecture
is that CISA mesh navigation can easily be stopped by a ridge of low goodness score CISAs. Some good enough $P_j$'s may be left behind if they are hidden behind such ridges. We will see that, in addition to the approximation of ideal weaving scheme, ARD propagation is another partial solution to this problem.

The extended flat architecture takes the same topology as the pure flat architecture. The abstract ISAs in the mesh, also called CISAs, are linked together with bi-directional acquaintance link based on the coherency of their curiosity ARDs. But now a CISA $C_i$ can take one or more curiosities. Similarly, a PISA $P_j$ can be a curiosity of one or more CISAs. Therefore curio($C_i$)'s $\subseteq \{P_j$'s$\}$ may overlap. We will also use the general abstract ISA mesh navigation algorithm $\text{IUA::NavigateAbstractISAMesh}$ for IUA interaction with the CISA mesh. The major change lies in the way an interest ARD is computed and propagated.

Recall that the CISA in the pure flat architecture is egoistic: the interest ARD uploaded by $C_i$ only constitutes his own curiosity. In symbol, interest($C_i$, depth = 0) = $\bigoplus_{P \in \text{curio}(C_i)} P$. We propose two schemes to enlarge the 'social circle' of $C_i$ without establishing new acquaintance relationships. The first way is to pass the interest ARDs of indirect acquaintances to $C_i$. The second way is to have the interest ARD of an acquaintance to include the curiosity ARDs of indirect acquaintances. This increases the tendency of the IUA to route a query to intermediate node which acts as a bridge to relevant acquaintances.

The first approach requires the CISAs to take the role of mediator — relaying the interest ARDs to others without merging. Recall that $\text{Acq}_n(C_i)$ denotes the set of CISAs reachable from $C_i$ within $n$ hops (acquaintance links). We would like $C_i$ to hold the interest ARDs of CISAs in $\text{Acq}_n(C_i)$. The radius of his 'social circle' is boosted from 1 to $n$. This can be done by having each CISA broadcasting his interest ARD to his acquaintances. The broadcast packet is tagged with a 'hop-left' counter which is decremented every time the ARD is transferred along an acquaintance link. By initializing the counter to $n$ and cooperatively relaying each others' interest ARDs until 'hop-left' = 0, every $C_i$ will hold interest($C$), $C \in \text{Acq}_n(C_i)$ in his RD base. The CISA mesh will be a more friendly community.
Some bandwidth can also be saved by exchanging messages like 'I have this ARD. Do you want it?' before transferring the relatively bulky ARDs. (Similar message 'IHAVE' can be found in NNTP [23])

The second approach is to include the curiosity ARDs of (indirect) acquaintances in the computation of an interest ARD. In other words, the interest ARD takes depth larger than 0. It would be nice if we can compute the interest ARD of arbitrary depth in (5.5) efficiently. Unfortunately, the presence of loops in the CISA mesh makes the calculation of interest(\(C_i, \text{depth} = n\)) complex even for small \(n\). Recall that in the extended hierarchical architecture, a MISA can safely perform max-based ARD merging and coverage signature superimposing even if the merging ARDs have overlapping coverage. Idempotence of the operations renders this harmless. Now suppose a CISA \(C_2\) attempts to compute interest(\(C_2, \text{depth} = n\)) for uploading to \(C_1\). The ARD would be awkward for \(C_1\) to use if it includes curio(\(C_1\)). For example, he cannot make sure that high \(G^{\text{max}}(C_2)\) actually is not due to high \(G^{\text{max}}(\text{curio}(C_1))\). The only way to find out is to route the query to \(C_2\) and end up looping back to himself. The worse thing is that \(C_2\) himself cannot presume interest(\(C_3, \text{depth} = n\)), \(C_3 \in \text{Acq}(C_2)\), to be free of curio(\(C_1\))! As a result of the loops, \(C_i\) cannot just merge the interest ARDs of his acquaintances to get his own interest ARD. We have to broadcast the curiosity ARDs curio(\(C_i\))’s to perform merely max-based ARD merging and coverage signature superimposing. It incurs the same cost as sum-based ARD merging. Hence, merging curiosity ARDs from indirect acquaintances takes two steps. First, \(C_i\)'s broadcast curio(\(C_i\))'s with ‘hop-left’ set to 0 similar to broadcasting of interest ARDs discussed previously. Then, after receiving the necessary curiosity ARDs from \(\text{Acq}_n(C_i)\) (and dropping duplicates), \(C_i\) can merge them himself.

Interest ARD broadcasting and increasing the depth of interest ARDs are expensive in nature. Fortunately, one can apply both of them selectively. One can broadcast interest(\(C_i, \text{depth} = 0\)) to a distance of \(n\), where \(n\) can be set stochastically. This enables the navigation algorithm \text{IUA::NavigateAbstractISAMesh}\) to cross a low goodness score ridge of width \(n - 1\). Another example is to select
some interest($C_i$, depth = $d$)’s for broadcasting. These interest ARDs summarize
a group of CISAs with similar curiosity ARDs. By transferring them to some
distant CISA $C_{far}$, we present an escape for an IUA trapped in the neighborhood
of $C_{far}$ of low goodness scores. This can be viewed as linking CISAs of dissimilar
curiosity ARDs. Consider for example the cubic CISA mesh in Fig. 5.13. Suppose
$C_1$ and $C_2$ can compute interest($C_i$, depth = 1), $i = 1, 2$, and a query $q$ is relevant
to the CISAs around $C_1$. By exchanging these ARDs between the two opposed
vertices, the IUA can jump from the probably low goodness score $C_2$ to $C_1$.

![Cubic CISA mesh](image)

Figure 5.13: A cubic CISA mesh

The cubic CISA mesh example also illustrates another important feature of the
extended flat architecture. With interest ARD broadcasting and/or $C_i$ uploading
interest($C_i$, depth > 0), the CISA mesh also functions as a partially replicated
mesh of MISAs. The whole CISA mesh is still able to perform properly when a
small number of CISAs fail.

5.7 Interoperation

Interoperation between the hierarchical and flat architectures is streamlined by
the common underlying model and the special scheme in the locator of ARDs.
We list two main approaches in this section.

The first approach is stacking a mesh on another by linking some ISAs on the
lower level mesh to some ISAs on the upper level as curiosity. Recall that a good starting CISA $C_{start}$ is important to the effectiveness of CISA mesh. A simplistic solution is to index all the CISAs in the mesh by a MISA tree. In fact, one does not need to index all CISAs. It is sufficient to index only some representative CISAs of high goodness scores. These CISAs $C_j$'s are taken as curiosities of the leaf nodes $M_i$'s of the MISA mesh in the extended hierarchical architecture and upload their interest ARDs interest($C_j$, depth = $d$) as the curiosity ARDs of $M_i$'s. These ARDs should also include coverage signature. Enabled with heuristics of choosing wisely among ARDs with overlapping coverage, the IUA can then use the MISA mesh to perform $C_{start}$ selection. On the other hand, one can stack a CISA mesh on top of a set of MISA trees by linking the roots of trees to the CISAs as curiosity. See Fig. 5.14.

![Diagram of MISA on CISA mesh on MISA trees](image)

Figure 5.14: MISA on CISA mesh on MISA trees

The access protocol is identified in the ARD as collection locator, the IUA can navigate the above composite mesh by installing suitable navigation modules and dispatching statement, as shown in the pseudocode `IUA::ProcessQuery` in Fig. 5.15.

The second approach is embedding. It is based on the recognition that the extended pure and flat architectures are mostly the same. One can introduce some MISAs $M_k$'s within a CISA mesh $C_i$'s. Such $M_k$ does not have curiosity himself, but serves mainly as a gatherer of ARDs from CISAs nearby. These
function IUA::ProcessQuery(q, collsel-param, collrank-param)
begin
    L := starting ISA
    // phase 1: collection selection
    while L contains only abstract ISA
        if L contains only MISA
            for each \( M_i \) in L
                // \( L_{leaf} \) is a list of leaf node of the Misa tree that is good enough
                \( L_{leaf} := \) IUA::TraverseMisaTree\( (M_i, q, \) collsel-param, collrank-param\)
                merge \( L_{leaf} \) to L
        else if L contains only CISA
            for each \( C_i \) in L
                // \( L_{curio} \) is a list of curiosity of the Cisa mesh that is good enough
                \( L_{curio} := \) IUA::NavigateCisaMesh\( (C_i, q, \) collsel-param, collrank-param\)
                merge \( L_{curio} \) to L
        else if L contains only Abstract ISA in the extended architectures
            for each \( A_i \) in L
                // \( L_{curio} \) is a list of curiosity of the Cisa mesh that is good enough
                \( L_{curio} := \) IUA::NavigateAbstractIsaMesh\( (A_i, q, \) collsel-param, collrank-param\)
                merge \( L_{curio} \) to L
        else
            error "Abstract ISA protocol not installed"
    endif
    // now L contains only Pisa
    // phase 2: submit query to \( P_j \) with relevant resource
    for each \( P_j \) in L // may be parallelized
        // \( R_j \) is the result list from \( P_j \)
        \( R_j := \) PISA::QueryPisa\( (P_j, q) \)
    // phase 3: result merging
    merge \( R_j \) to R, a final result list
    return R
end

Figure 5.15: General query processing algorithm for the IUA to interact with various kinds of abstract ISAs

83
\( M_k \)'s can be used as representatives for the starting CISA selection subsystem. Since they have higher goodness scores than the neighboring CISAs, they help to approximate the ideal CISA mesh weaving scheme. Such MISA mesh within CISA mesh is highly configurable. The IUA can use \texttt{IUA::NavigateAbstractISAMesh} without change.

5.8 Other issues

5.8.1 Decentralizing ISA navigation

It may seem unnatural that one operation in this work remains highly centralized. In our discussion up to this moment, query routing appears as message exchanges between the IUA and various abstract ISAs and primary ISAs. The IUA may easily become a bottleneck when remote ISAs in disparate locations have to be accessed.

Fortunately, abstract ISA mesh navigation can be decentralized easily. The centralization is only a convenience in the abstract model. In real-world deployment, query routing can be decentralized by routing intermediate results between abstract ISAs. Another way to do this is to use special agent that navigates an abstract ISA mesh for the IUA. For example, a MISA tree can provide such an agent \( N \) as a single point of service. The IUA only submits the query to \( N \), which will navigate the tree and return the results to the IUA.

Lastly, the navigation process can be distributed at the user end. A community of users with common interests may share a caching agent \( K \) that caches fragments of ARDs. The cache in \( K \) helps to locate a better starting point in a MISA tree or CISA mesh.

5.8.2 Result merging

After collection selection, the IUA will submit the query to the selected primary ISAs and merge the results into a single list. Result merging will be trivial if the
primary ISAs are homogeneous, using the same document ranking algorithm and the same value for global statistical information like inverse document frequency ($idf_j$). The relevancy scores from individual primary ISAs are of the same scale and can be compared without modification. As shown in [31], one does not need to maintain very accurate global statistical information to ensure document ranking effectiveness. Therefore, we have been assuming the availability of sufficiently accurate $idf_j$'s.

Result merging for heterogeneous primary ISAs is more complex. Typical solution is to scale the relevancy scores from a primary ISA with some function of its goodness score [35, 12].
Chapter 6

Conclusion

We have gone a long way from a centralized index server (in our model, a single primary ISA) in Chapter 1 to a general distributed resource discovery system of hybrid architectures in Chapter 5. All our discussion has been done under the the abstract DRD model in Chapter 3. Some existing and new collection ranking methods are described and compared experimentally in Chapter 4. These methods are sufficient for one to deploy a centralized collection subsystem as a single meta-indexer. We classify the methods into two categories: absolute and relative, and recognize that absolute collection ranking methods are more appropriate in a distributed collection selection subsystem. In submitting a query to the system, the user may specify the degree of minimum resource relevancy desired with collection ranking parameters ($s_{\text{min}}$ and $r_{\text{min}}$). In addition, he may state the collection selection criteria with collection selection parameters ($g_{\text{min}}$ and $n$). The information not only enables the system to return more accurate results, but also allows more efficient internal operation like pruning unpromising branches in the abstract ISA mesh.

Several architectures have been discussed in Chapter 5. The hierarchical architectures are suitable for well structured topic-specific and organizational hierarchies. A nice feature is its completeness in collection selection. The flat architectures, on the other hand, are better for linking a set of closely related
collections.

We start with the familiar hierarchical architecture and show how to design ARD merging so that the absolute collection ranking methods examined in Chapter 4 can be used in the distributed case. We then move to the flat architecture which connects ISAs with coherent collections. We see that both the similarity of the resource collections and their estimated goodness scores have to be considered in weaving the ISA mesh. Two hybrid architectures are then introduced. The extended hierarchical architecture permits MISAs with overlapping acquaintances and models topic specific MISAs better. The extended flat architecture introduces ARD merging and propagation which helps the user agent to overcome some pitfalls in the CISA mesh weaving. Finally, we see how these architectures interoperate and how the abstract ISA mesh navigation can be decentralized.

The resultant DRD system segregates metadata extraction to the Index Extraction Agents (IEAs) and index building and query processing of the resources of an information provider in individual primary ISA. The task of collection selection is decentralized among a set of abstract ISAs of certain architecture. This arrangement brings scalability. Efficient navigation and the relatively small size of ARD reduce the computing resource consumption in index building and query processing. Autonomy of information and service providers is respected by requiring them to only export ARDs in order for them to join the global DRD system. They do not need to expose their own indices.

The DRD system is also customizable to user demands by supporting topic specific ISAs. A global mesh of topic specific MISAs can be set up according to 'sub-topic super-topic' relationship in the extended hierarchical architecture. Alternatively, a group of closely-related collections may be connected according to their coherence with each other in the flat architectures. We do not work on the details of replication in the abstract ISA mesh. Instead, we enable the IUA to navigate a mesh with (partial) replicates with the coverage signatures in the ARDs. Two ARDs with exact coverage signature are very likely to be replicates. Besides, the extended flat architecture provides a delicate way of interlaced partial
replicates. All these make the system more robust under partial failure.

Here is some of the future work in our mind:

- Information filtering
  In information filtering, new incoming documents are matched against existing user profiles and dispatched to potential interested users. It is interesting to see how well the two architectures adapt to this paradigm.

- Experimental evaluation of the flat architectures
  The ideal weaving scheme in flat architectures can only be approximated in practical environment. Experimental evaluation of the weaving scheme needs to be done. There are other approaches to merge ARDs and mesh navigation not covered by our analysis. For example, one might give diminishing weights to ARDs from farther nodes in merging them. Search techniques from artificial intelligence like genetic algorithm and simulated annealing may also help to overcome the pitfalls in imperfect mesh weaving.

- Other retrieval models
  We have only worked on vector-based retrieval model. Although boolean-based retrieval model can be simulated in our model by having a boolean goodness score, some other models like the probabilistic model may not be formulated in this way. Moreover, some useful features of sophisticated information retrieval systems like relevance feedback are not considered.
Bibliography


91


