

**SEARCH ENGINE SELECTION AND RESULT  
MERGING IN METASEARCH**

*A dissertation submitted to the Jawaharlal Nehru University  
in partial fulfillment of the requirements  
for the award of the degree of*

**MASTER OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND TECHNOLOGY**

**BY**

**RAJESH KUMAR**



**SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI – 110067**

**JULY 2010**

*Dedicated*

*to*

*My Loving Parents*

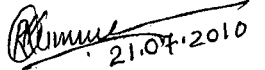


**SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI – 110067**

## **DECLARATION**

This is to certify that the dissertation entitled “**Search Engine Selection and Result Merging in Metasearch**” is being submitted to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science & Technology**, is a record of bonafide work carried out by me under the supervision of **Dr. T.V. Vijay Kumar**.

The matter embodied in the dissertation has not been submitted in part or full to any University or Institution for the award of any degree or diploma.

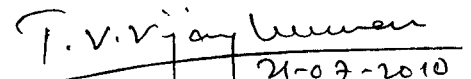
  
21.07.2010  
**Rajesh Kumar**  
**(Student)**




SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI – 110067

## CERTIFICATE

This is to certify that this dissertation entitled “**Search Engine Selection and Result Merging in Metasearch**” submitted by **Mr. Rajesh Kumar**, to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, for the award of degree of **Master of Technology in Computer Science & Technology**, is a research work carried out by him under the supervision of **Dr. T. V. Vijay Kumar**.

  
21-07-2010  
**Dr. T. V. Vijay Kumar**  
(Supervisor)

  
**Prof. S. Minz**  
(Dean)

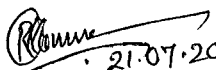
# ACKNOWLEDGEMENT

First, I would like to heartily thank my parents and brothers for their positive support at every moment in my life. Specially my elder brother **Sita Ram** has been a constant source of encourages and inspiration nurturing of my work and always supporting my decision all the way during my studies in spite social difficulties.

Sincerely express my appreciation to my supervisor **Dr. T.V. Vijay Kumar** for giving me an opportunity to work under his supervision. His moral support and encouraging nature to build a confidence to achieve our aim. Sir always appreciates to work hard during entire period of dissertation and constant support throughout completion of dissertation work. I have learned several incredible things from him including joyful positive attitude towards work and perfection at doing things. During the entire dissertation, I realized that Sir is not only a good teacher also a good human being specially his helping nature

I would like to thanks **Porf. S. Balasundaram** sir, **Dr. Zahid Raza** sir for their moral support and I would also like to thank **Prof. S. Minz** (Dean, SC&SS)

I would like to thanks to seniors **Mohammad Haider, Santosh Kumar, Ajay Kumar Verma** for his their right suggestion and guided in my dissertation work. I give special thanks my friend **Rajeev Kumar** for their appreciating to doing Master of Technology and support at every moment. I would like to thanks my lab mate **Anil Kumar Giri** for his contribution their knowledge which was helpful in my dissertation work and **Kumar Dilip** for encouraging me which increase my confidence to work hard. I would like to thanks **Manmohan** for their cooperative and helping nature during my dissertation work.

  
21.07.2010  
**Rajesh Kumar**

# Table of Contents

<b>Chapter 1: Introduction</b> .....	<b>1-22</b>
<b>1.1 Information Retrieval</b> .....	<b>3</b>
1.1.1 Basic Concept of IR .....	4
1.1.2 IR Model.....	5
1.1.3 IR Problem.....	9
1.1.4 IR Performance Evaluation.....	9
<b>1.2 Metasearch Engine</b> .....	<b>10</b>
1.2.1 Architecture of Metasearch Engine.....	10
1.2.2 Advantage of Metasearch Engine.....	12
<b>1.3 Search Engine Selection</b> .....	<b>13</b>
1.3.1 D-WISE (Distributed Web Index Search Engine).....	14
1.3.2 Collection Retrieval Interface Network (CORI-Net).....	15
1.3.3 Query Similarity (qSim).....	17
1.3.4 Modeling Relevant Document Distribution (MRDD).....	17
1.3.5 Profusion Approach .....	18
<b>1.4 Result Merging</b> .....	<b>19</b>
1.4.1 CORI Merging Technique.....	19
1.4.2 Semisupervised Learning Approach.....	20
1.4.3 LMS Merging Approach .....	21
1.4.4 Abstract Merging Approach.....	21
1.4.5 OWA-Based Merging Approach.....	22
<b>1.5 Aim of the Dissertation</b> .....	<b>22</b>
<b>1.6 Organization of the Dissertation</b> .....	<b>22</b>
<b>Chapter 2: Search Engine Selection</b> .....	<b>23-51</b>
<b>2.1 Query Similarity</b> .....	<b>24</b>
2.1.1 An Example.....	26
<b>2.2 Modeling Relevant Document Distribution (MRDD) Approach</b> .....	<b>33</b>
2.2.1 An Example.....	34
<b>2.3 Example Based Comparison</b> .....	<b>40</b>
<b>2.4 Experimental Results</b> .....	<b>50</b>
<b>Chapter 3: Result Merging</b> .....	<b>52-66</b>
<b>3.1 OWA-Based Merging Approach</b> .....	<b>53</b>
3.1.1 An Example.....	55
<b>3.2 Abstract Merging</b> .....	<b>57</b>
3.2.1 An Example.....	58
<b>3.3 Example Based Comparison</b> .....	<b>60</b>
<b>3.4 Experimental Results</b> .....	<b>65</b>
<b>Chapter 4: Conclusion</b> .....	<b>67-68</b>
<b>Reference</b> .....	<b>69-72</b>

# CHAPTER 1

## Introduction

In recent years, the web has become a huge source of information, which is mostly unstructured in the form of text or image. The people all over the world pose queries using their favorite search engine to find relevant information. However, every search engine use their own method or algorithm to ranking the retrieved results. The metasearch engines usually send the query simultaneously to different search engines resulting in queries being processed in parallel thereby saving time. Metasearch engine [L98] is a tool that allows searching multiple search engines at the same time and returning more comprehensive and relevant document that satisfy the information needs of the user, efficiently. In other words, metasearch engine is a system that provides unified access to multiple existing search engines[MYL02]. When user poses a query to the metasearch through the user interface, the metasearch engine is responsible to identify appropriate underlying search engine which have relevant document with respect to user query. Each search engine has a text (unstructured data) database, defined by the set of documents that can be searched by search engine. All underlying search engines retrieve most of the

relevant documents, which the metasearch engine combine into single ranked list and displays them to the user. Ranking of the document is based on user query, top rank document have high query weight. The main goal of metasearch over the single search engine is increased coverage and a consistent interface [SE97]. A consistent interface is necessary for the metasearch engine to ensure that result from several places can be meaningful combined while user is not aware about underlying search engine. Two types of the search engines exist[MYL02] namely general purpose search engine and special purpose search engine. The general purpose search engine aims to provide capability to search all type of the web page with respect to the user query like Google, Altavista, Excite, Lycos and HotBot etc[MYL02]. The special purpose search engine retrieves the document for a defined domain such as specific subject area. Example, Cora search engine focus on computer science search paper and Medical World Search focus to retrieved the medical information. It is believed that hundreds of thousands of special purpose search engines currently exist on the web [BM]. The motivation for the metasearch [MW] includes

- increase in the search coverage as rate at which the web has been increasing is much faster than the indexing capability of a single search engine. Also, the metasearch engine effectively combines the coverage of all underlying search engines.
- retrieves relevant document by merging documents retrieved from underlying search engine and ranking them with respect to the user query thereby making it convenient and reliable for the user to retrieve relevant information.
- facilitate the invocation of multiple search engines. All relevant documents are stored in database of each search engine. The user in order to retrieve most relevant



document needs to first identify search engines with most relevant information followed by sending queries to each search engine in the corresponding query format. The documents retrieved will not be ranked and it would become difficult for the user to determine the relevant documents amongst the retrieved document. This problem would be compounded if there are large number of search engines each returning documents as per its ranking algorithm. As a result, it would be more difficult to choose the relevant document. Metasearch search engine solves this problem by select the appropriate search engines for a given user query and merge the results retrieved from them.

Before discussing metasearch engine and its component in detail, a short overview of some basic concepts of information retrieval (IR) is given. These concepts would be of importance while discussing search engine selection and result merging in metasearch.

## **1.1 Information Retrieval**

Information Retrieval (IR) is a system that finds relevant information from the unstructured documents that satisfies user information needs. Unstructured documents [AMS00] can be natural language text, audio, image, photographic image, video, etc. The records that Information Retrieval (IR) addresses are often called “documents”. These documents can structured, semi-structured, unstructured or a combination of these.

According to [S89], “Information-retrieval system process files of records and requests for information, and identifies and retrieves from the files certain records in response to the information requests. The retrieval of particular documents depends on the similarity between the documents and the queries, which is measured by comparing the values of certain attributes to records and information requests.” Traditional IR assumes that the basic information unit is a document, and a large collection of documents is available to

form the text database. In IR system, user sends a query according to their requirement. Query has multiple forms [L98] of which one is passed to the information system. Query may be a Keyword query, Boolean query, Phrase query, full document query or Proximity query. Query format is a list of keywords, called “terms” which provides the semantic to the documents. Ranking of the relevance documents uses the weight of the query.

### 1.1.1 Basic Concept of IR

Information retrieval (IR) [L98] helps users to find information matching their needs. Technically, IR studies the learning, organization, storage, retrieval, and distribution of information. Historically, IR is all about document retrieval, emphasizing document as the basic unit. A general architecture of Information Retrieval [L98] is given in Figure 1.1.

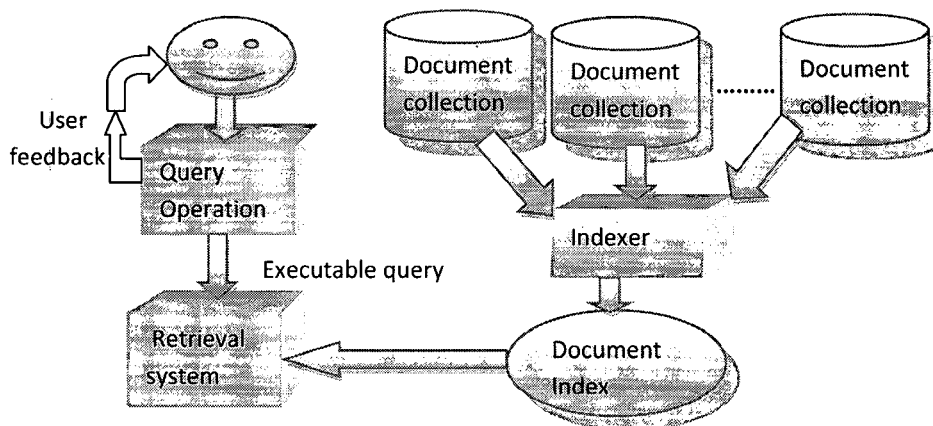


Figure 1.1: Architecture of IR Model [L98]

In Figure 1.1, the user with information need poses a query called user query, to the IR system through the query operations module. The retrieval module uses the document index to retrieve those documents that contain some query terms and compute relevance scores for them. This is followed by ranking them according to the scores. The ranked documents are then presented to the user. The document collection is also called the text database, which is indexed by the indexer for efficient retrieval. A user query represents the user’s information needs, which is in one of the following forms [L98] like Boolean

query, phrase query, keyword query, proximity query and full document query. Query will perform some preprocessing operation before it is sent to the information retrieval system. The query operations module can range from very simple to very complex. In the former, it does nothing but just pass the query to the retrieval engine after some simple preprocessing like stopword removal, stemming, digit, hyphen and number removal etc.

### **1.1.2 IR Models**

An IR model [L98] governs how a document and a query are represented and how the relevance of a document to a user query is defined. Several information retrieval model exist, namely Boolean model, Vector Space model, Probabilistic model, Language model, Inference model, Impact factor model, Connectivity model, Mutual citation model, Page Rank model, HITS, SALSA model, Associative Interaction model and Bayesian model. In information retrieval (IR) system each query and document are associated with a weight [SK06]. Boolean Model is based on set theory and Boolean algebra. In the Boolean model [L98], the weight of the queries and documents are represented as the set of terms with each term either present or absent in a document. The weight  $w_{ij}$  ( $\in \{0, 1\}$ ) of term  $t_i$  in document  $d_j$  is 1 if term  $t_i$  is present in document  $d_j$  otherwise weight of term  $t_i$  is 0. Query terms are combined using Boolean operators AND, OR and NOT which have their usual meaning. Document retrieval using Boolean query depends on discrete value say true or false i.e relevant or irrelevant. If for the given query it is true then relevant documents are retrieved else irrelevant document will be retrieved. Boolean model has some drawback like there is no partial matching, it is based on exact matching of the query. Ranking of the relevant document is not provided by this model. Translating information need to Boolean model. These limitations are overcome by the vector space model.

In vector space model [L98] the document and query are presented as vector of query term weight and document term weight. In this model, ranking of the retrieved document is based on degree of the relevance to a query. The degree of relevance to the query is computed by similarity between query and document. Let  $q = (q_1, q_2, q_3, \dots, q_m)$ , where  $q_1, q_2, q_3, \dots, q_m$ , are query term weights and  $d = (w_1, w_2, w_3, \dots, w_m)$  where  $w_1, w_2, w_3, \dots, w_m$  are document term weight then similarity between document and query is computed by cosine similarity [L98] and is given by

$$Sim(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \times \|q\|} = \frac{\sum_{i=1}^{|M|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|M|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|M|} w_{iq}^2}}$$

The ranking of the document is done using their similarity value. The document having higher similarity value would be ranked higher amongst documents. In vector space model the weight  $w_{ij}$  of term  $t_i$  in the document  $d_j$  are computed as  $w_{ij} = idf_i \times tf_{ij}$  where  $tf_{ij}$  is normalized term frequency and  $idf_i$  is the inverse document frequency.

The term frequency [L98] is defined as term  $t_i$  in the document  $d_j$  is the number of times that the term  $t_i$  appear in the document  $d_j$  called term frequency. The normalized term frequency is given by

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \dots, f_{ij}\}}$$

where the denominator is the frequency of the term that occur the maximum number of times in a document  $d_j$  and  $f_{ij}$  is the frequency of the term  $t_i$  in the document  $d_j$ . If term  $t_i$  does not appear in  $d_j$  then normalized term frequency is zero i.e.  $tf_{ij} = 0$ .

Inverse document frequency [L98] is computed by

$$idf_i = \log \frac{N}{df_i}$$

where  $N$  is the total number of document in the system and  $df_i$  is the total number of documents in which term  $t_i$  appears at least once.

The weight  $w_{ij}$  of term  $t_i$  in the document  $d_j$  is computed as

$$w_{ij} = \log \frac{N}{df_i} \times tf_{ij}$$

The term weight  $w_{ij}$  of a query  $q$  can also be calculated as in [SB88] by

$$w_{iq} = \left( 0.5 + \frac{0.5 \text{freq}_{iq}}{\max \{f_{1q}, f_{2q}, \dots, f_{|v|q}\}} \right) \times \log \frac{N}{df_i}$$

where  $\text{freq}_{iq}$  is the raw frequency of term  $t_i$  in collection document for query  $q$ .

The main advantage of vector space model is

- Partial matching is allowed to retrieve the relevant information.
- Term weight strategy improves information retrieval performance
- Ranking of the relevant documents based on cosine similarity between query and document also improves the performance of retrieving relevant information.

Probabilistic models [BR04] used probability theory to evaluate probability of relevance of a document to a user query. The probability of relevance is estimated using rigorous experiments with innovative ideas. Unlike similarity-based models such as Boolean and Vector Space Model, Probabilistic models are theoretically sound and realistically cater to the IR problem. For a given query  $q$  and document  $d_j$  in a collection, the model tries to find the probability of the document being relevant with respect to a given user query.

Let  $R$  be the set of relevant documents and  $\bar{R}$  be the set of non-relevant documents then the probability of that the document  $d_j$  is relevant to query  $q$  is define by  $P\left(\frac{R}{d_j}\right)$  and

probability that document  $d_j$  is non-relevant to query  $q$  is define by  $P\left(\frac{\bar{R}}{\bar{d}_j}\right)$ . The

similarity between document  $d_j$  and query  $q$  is define as [BR04]

$$sim(d_j, q) = \frac{P\left(\frac{R}{\bar{d}_j}\right)}{P\left(\frac{\bar{R}}{\bar{d}_j}\right)}$$

Now finding the similarity between document  $d_j$  and query  $q$  using the Baye's Rule [BR04] is defined as

$$sim(d_j, q) = \frac{P\left(\frac{\bar{d}_j}{R}\right) \times p(R)}{P\left(\frac{\bar{d}_j}{\bar{R}}\right) \times p(\bar{R})}$$

Where  $P\left(\frac{\bar{d}_j}{R}\right)$  is the probability of randomly selecting the document  $d_j$  from relevant

document  $R$ . and  $P\left(\frac{\bar{d}_j}{\bar{R}}\right)$  is the probability of randomly selecting document  $d_j$  from the

non relevant document set  $\bar{R}$ .  $p(R)$  is the probability that a document randomly selected from the entire collection is relevant and  $p(\bar{R})$  is the probability that a document randomly selection from the entire collection is non-relevant.

Since the values of  $p(R)$  and  $p(\bar{R})$  are same for all document in collection therefore similarity between document  $d_j$  and query  $q$  is calculated as in [BR04] by

$$Sim(d_j, q) \sim \frac{P\left(\frac{\bar{d}_j}{R}\right)}{P\left(\frac{\bar{d}_j}{\bar{R}}\right)}$$

### 1.1.3 IR Problem

Problem in IR system is to identify the most relevant documents among retrieved documents for a given user query i.e. ranking of documents is the key. The Boolean model of IR system only returns matching documents. All relevant documents are not found in a single search engine as it is specific to retrieve information of an organization and may not return all relevant documents to a user query. One major problem in IR system is handling the documents dynamically. Mostly information are retrieved from static database. These problems are addressed by the metasearch engine.

### 1.1.4 IR Performance Evaluation

While considering retrieval performance evaluation for IR system [BR04], the retrieval task to be evaluated is considered first. The retrieval task is user specifying his information need through an interactive process in the IR system. The retrieval effectiveness needs to be evaluated using an measure that aims to estimate the nature, quality, ability, extent, or significance of the information retrieval system The performance of IR system depends on value of precision and recall. A good IR system have high precision (retrieve very few non-relevant document) value and high recall (retrieve as many relevant documents as possible) value.

Let  $|A|$  be the number of relevant documents and  $|B|$  be the number of retrieved documents then Precision is the fraction of retrieved documents that are relevant and Recall is the fraction of relevant documents that are retrieved. These are given by [BR04]

$$\text{Precision} = \frac{|A \cap B|}{|B|} \qquad \text{Recall} = \frac{|A \cap B|}{|A|}$$

## **1.2 Metasearch Engine**

Metasearch engine is a tool that sends the user requests to multiple primary search engines combine the results retrieved from them together and display them to the user. The information on the web has increased rapidly over time. When query is submitted to the metasearch engine, it decides which underlying search engine will be selected for a user query, how to preprocess the submitted query with respect to best utilization of the underlying search engine and how to merge result retrieved from the search engines. All decision is taken by the metasearch engine based on keyword of user query. The challenging aim of metasearch engine is the selection of appropriate search engines and merging results retrieved from them, with respect to the user query.

### **1.2.1 Architecture of Metasearch Engine**

In recent, WWW is use as largest digital library [KP+05] and people all over world use the digital library to find the relevant information according to their information needs. Information on the digital library is stored in disparate sources and each of these sources has their own search capability. In general any organizations have their own web site and also have their own search engine. When user poses a query to find relevant information, the metasearch engine finds such information using its components. The main components of a metasearch engine are search engine selector, document selector, query dispatcher and result merger. The aim of metasearch is to maximize the precision or retrieval effectiveness while minimizing the cost. In other words, the aim of a metasearch engine is to retrieve most relevant information as much possible while retrieving few irrelevant information for a user query. In order to carry out this, the metasearch engines select the most appropriate search engine containing relevant information. Each selected search engine should retrieve as much relevant information as possible. The architecture of a



metasearch engine, as in [MYL02], is shown in Figure 1.2. The metasearch engine consist of four components namely Search Engine Selector, Document Selector, Query Dispatcher and Result Merger and these are discussed next [MYL02]

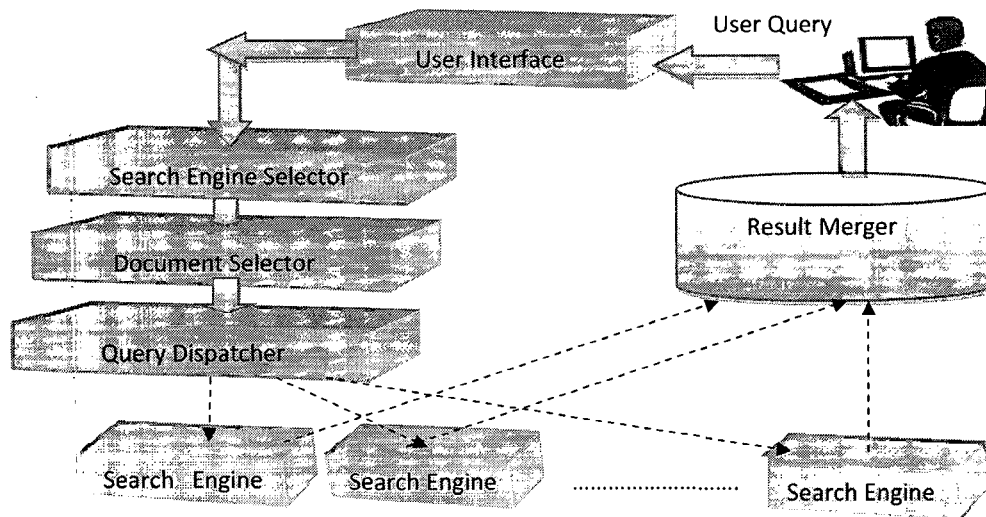


Figure 1.2 Architecture of Metasearch Engine [MYL02]

**Search Engine Selector:** The search engine selector selects the appropriate underlying search engine with respect to the user query. A good search engine selector should correctly identify search engines while minimizing identifying irrelevant search engines. The approaches for selecting search engines are discussed later in this chapter.

**Document Selector:** The document selector determines what documents to retrieve from the selected search engines. The aim is to retrieve more relevant documents with few irrelevant documents. To find out the relevant information different similarity measure are used which estimate the relevance between document and user query. The similarity is measured based on a pre-defined threshold value. The high similarity value shows that the information is more relevant with respect to the user query.

**Query Dispatcher:** The query dispatcher has a mechanism to establish a connection of a server with each selected search engine in order to dispatch query to each of these search engines. In general, the user query will be sent to the search engine after preprocessing. Every search engine may or may not have the same query as posed on the metasearch engine.

**Result Merger:** The result merger merges document retrieved from the selected search engines. The result merger combines all the result into a single ranked list and arranges the documents in descending order with their global similarity with respect to the user query. The top most documents having higher global similarity in the raked list are returned to the user through the interface.

### **1.2.2 Advantage of Metasearch Engines**

There are several advantages [LBE] of the metasearch engine over the single search engine

- Retrieves the documents from many underlying search engines which may be missed by a single search engine
- Retrieves the documents from various underlying search engine in parallel way as compared to retrieving documents from each search engine one at a time.
- Retrieves the documents by eliminating duplicates from various search engines as compared to retrieving documents from each search engine separately resulting in duplicate documents.
- Retrieves the documents from many underlying search engine enables exploring how to best combine the separate result lists.

Next, Search Engine Selection and Result Merging are discussed in the subsequent sections

### 1.3 Search Engine Selection

When a user poses a query to the metasearch engine, it invokes the search engine selector to select the appropriate search engines that satisfies the information needs of the user query. A good search engine selector uses the different selection algorithms to identify potentially useful search engine for a given user query. The search engine selection approaches can be classified into three categories [MYL02].

**Rough Representative Approaches:** In rough representative approaches, the content of each underlying search engines are often describe by using few keywords or paragraph [MYL02]. The rough representative are often generated manually and provides a general idea on what the search engine is about. The rough representative approaches is described in ALIWEB [KM94]

**Statistical Representative Approaches:** The statistical representation approaches describes the content of the search engine using different kind of statistical information. The statistical information can be documents frequency of the term, average weight of term in the documents [MYL02]. There are many kind of approaches based on statistical information like Web Index and Search Engine (D-WISE) [YL96], Collection Retrieval Interface Network CORI-Net [CLC95], Generalization Glossary of Servers' Server gGLOSS[GGM95].

**Learning Based Approaches:** The learning based approaches determine the utility of a search engine for a new user query based on retrieval experience result from the past queries. The experience result may be obtained using the training query. Using a training query [VGL95a] to build a model of the distributions of the relevant documents retrieval by all search engines and using this model to obtain the maximum number of documents from each search engines. There are three type of learning approaches namely static,

dynamic and combined or hybrid [MYL02]. In the static learning approach, the retrieval knowledge, once learned, will not be changed therefore it is not suitable for search engines and query patterns that changes frequently. Modeling Relevant Document Distribution (MRDD) approach [VGL95b][MYL02] is a static learning approach. On the other hand dynamic learning approaches the retrieval knowledge change with changes in the content of the search engines and the query patterns. Savvy Search engine [DH97] is a dynamic learning approach. The hybrid approach encompasses both static and dynamic learning. ProFusion approach [FG99] is used to select the search engine employing the combined learning approach i.e. static as well as dynamic approach

Some of the search engine selection approaches are discussed next.

### 1.3.1 D-WISE (Distributed Web Index Search Engine)

D- WISE approach [MYL02] is based on static approach in which selects appropriate search engines using the document frequency of each term as well as documents in each underlying search engine. For a user query  $q$ , it computes the ranking score of the search engine with the respect to the user query. The search engines with higher score have most relevant documents compare to the lower score search engines. Ranking score of the search engine is computed in the following way:

First compute the  $CV_{ij}$  [MYL02]

$$CV_{ij} = \frac{\frac{df_{ij}}{n_i}}{\frac{df_{ij}}{n_i} + \frac{\sum_{k \neq i}^N df_{kj}}{\sum_{k \neq i}^N n_k}}$$

where

$df_{ij}$  = documents frequency of  $j^{th}$  term in the  $i^{th}$  search engine

$n_i$  = total number of document in  $i^{th}$  search engine

$CV_{ij}$  = cue validity of  $j^{th}$  term for the  $i^{th}$  search engine

$N$  = total number of distributed search engine

$AVC_j$  = average of  $CV_{ij}$  for all search engines

$CVV_j$  = variance of the  $CV_{ij}$  for each  $j^{th}$  term over all search engines

Second compute the variance of the  $CV_{ij}$  for each  $j^{th}$  term over all search engines

[MYL02]

$$CVV_j = \sum_{i=1}^N \frac{(CV_{ij} - AVC_j)^2}{N}$$

Let two terms  $t_u$  and  $t_v$ , if  $CVV_u \geq CVV_v$  then term  $t_u$  is more weighted than term  $t_v$  for different search engine

Third compute the ranking of the  $i^{th}$  search engine with respect to the user query [MYL02]

$$r_i = \sum_{j=1}^M CVV_j \cdot df_{ij}$$

where  $M$  is number of query terms and  $r_i$  is the ranking score of  $i^{th}$  search engine.

The search engines with higher value of  $r_i$  would be selected by the metasearch engine.

### 1.3.2 Collection Retrieval Inference Network (CORI-Net)

In CORI-Net approach [CLC95], the selection of search engine is carried out using two pieces of information for each distinct term i.e. document frequency and search engine frequency. The technique of documents ranking with respect to the user query is known as inference network [CLC95]. If a term appears in  $k$  document in the search engine, the

term is repeated  $k$  times in the super document. Super document containing all distinct term in the search engine, As a result, the document frequency of a term in the search engine becomes the term frequency in the super document [CLC95]. Let  $N$  be the number of search engines,  $df_{ij}$  is document frequency of  $j^{th}$  term in  $i^{th}$  component search engine and  $se_j$  is search engine frequency of  $j^{th}$  term [SJ04]. If  $D_i$  is the super documents of all search engines, the relevant document due to the  $j^{th}$  term is computed as [SJ04]

$$P\left(\frac{t_j}{D_i}\right) = C_1 + (1 - C_1) \cdot T_{ij} \cdot I_j$$

where

$$T_{ij} = C_2 + (1 - C_2) \frac{df_{ij}}{df_{ij} + K},$$

$$I_j = \frac{\log\left(\frac{N + 0.5}{dbf_j}\right)}{\log(N + 1.0)}$$

$$K = C_3 \cdot (1 - C_4) + C_4 \cdot \frac{dw_i}{adw_i}$$

Where  $dw_i$  is number of words in  $D_i$  and  $adw_i$  is average number of words in  $D_i$ . The value of  $C_1, C_2, C_3, C_4$  are estimated empirically by performing experiment on the actual text collection [SJ04].  $P\left(\frac{t_j}{D_i}\right)$  is given by  $tfw \times idfw$  denoting the weight of term  $t_j$  in the super document corresponding to search engine  $D_i$  and can be estimated, for query term weight  $t_j$  in  $q$ . Now ranking score of each search with respect to given query can be calculated as [SJ04]:

$$\sigma_i = P\left(\frac{q}{D_i}\right) = \sum_{j=1}^K P\left(\frac{q}{t_i}\right) \cdot P\left(\frac{t_j}{D_i}\right)$$

where  $k$  is total number of query terms. The search engines are ranked based on  $\sigma_i$  and the top-ranked search engine are thereafter selected by the metasearch engine.

### 1.3.3 Query Similarity (qSim)

qSim [SLH09] algorithm utilize the retrieved results of past queries for selecting the appropriate search engines for a specific user query. The selection of the search engines are based on the value of relevance between user query and the search engine say  $rel(s_j|q)$ , which indicate the how likely it is for the search engine to be relevant to the user query  $q$  or what percentage of the relevant documents for the user query  $q$  is in the search engine  $s_j$ . Ranking of the search engines is carried out according to the value of  $rel(s_j|q)$ . Higher value of  $rel(s_j|q)$  means it is more appropriate search engine contain more relevant information for the user query. The value of  $rel(s_j|q)$  is depend on  $rel(s_j|p_i)$  and  $sim(p_i|q_i)$ , where  $rel(s_j|p_i)$  is the relevance between search engines and past queries and  $sim(p_i|q_i)$  is the similarity between all past queries with user query. The search engines with higher value for  $rel(s_j|q)$  are selected by the metasearch engine.

### 1.3.4 Modeling Relevance Document Distribution (MRDD)

Modeling relevance document distribution algorithm [VGL95B][MYL02] is a static learning based approach, which uses a set of training queries for learning. With the help of training queries, it identifies all the relevant documents returned from every search engine and arrives at a distribution vector for each relevant document. Similarly, it finds the distribution vector for each training query. With the help of cosine distance similarity function it finds the similarity between user query and all training queries and identifies the k-most similar training query and find the average relevant document distribution vector over k vector corresponding to the k-most similar training queries. Finally, average distribution vector is used to identify the appropriate search engines.

### 1.3.5 ProFusion Approach

ProFusion approach is a hybrid learning approach, which combines both static and dynamic learning approach. In the ProFusion approach, when a user query is received by the metasearch engine, the query is first mapped to one or more categories [FG99]. The query is mapped to a category that have at least one term that belong to the user query. The categories can be science and engineering, “Computer Science”, “Medical and Biotechnology”, “Business and Finance” and so on. The ProFusion approach considers thirteen categories. If C is a category and S is a search engine and set of training queries are identify for search engine then a score reflecting the performance of search engine S and category C is computed by [MYL02]

$$C \times \frac{\sum_{i=1}^{10} N_i}{10} \times \frac{R}{10}$$

Where C is the constant and R is the number of relevant documents among top-10 retrieved documents. Value of  $N_i$  is calculated as

$$N_i = \frac{1}{i}, \text{ if } i^{\text{th}} \text{ ranked document is relevant} \\ = 0, \text{ otherwise}$$

Similarly the score for all training queries associated with category C is averaged for search engine S and this average is the confidence factor. The sum of the confidence factor of each search engine with respect to query q be called ranking score of the search engine categories. The score of the search engine with respect to a given query is also updated dynamically [MYL02] based on retrieved result. If the user considers document d to be relevant and d is not the top most documents then the score of the search engine having document d is increased and score of all other search engine whose documents are ranked higher then d is decreased. The search engines with higher score are selected by the metasearch engine.



## 1.4 Result Merging

After the search engines are selected for the user query, the queries are processed against the selected search engines and documents are retrieved from each of them. These retrieved documents are merged into a single ranked list and are arranged in descending order of their global similarities. The result merging is a difficult task because the documents are stored in different type of search engines and document scores returned by the different search engine cannot be compared directly. The result merging of documents depend on the degree of overlap among documents retrieved from the selected search engine for a user query. If search engines are [MYL02] identical then many ranking algorithms are applied on same search engine to improve the retrieved effectiveness. This result merging problem is called data fusion [VGL95a]

Some of the results merging approaches are discussed next.

### 1.4.1 CORI Merging Technique

The CORI merging algorithms is associated with the CORI search engine selection algorithm. It uses a simple heuristic to normalize the search engine document scores. The Normalized score suitable for merging is calculated as in [NF03] as

$$C' = \frac{C_i - C_{\min}}{C_{\max} - C_{\min}}$$

which shows that the normalized  $i^{\text{th}}$  search engine and normalized search engine score is within the range [0, 1]. To find the value of  $C_{\max}$  and  $C_{\min}$  using [SC03a] formula as given below. For  $C_{\max}$ , set the value of  $T=1$  and for  $C_{\min}$  set the value of  $T=0$  for each query term.

$$T = \frac{df}{df + 50 + 150 \times cw / avg\_cw}$$

where

$df$  is the number of documents in  $i^{th}$  search engine that contain  $r_k$ ,

$cw$  is the number of terms occurrences in  $i^{th}$  search engine

$avg\_cw$  is the average of  $cw$  for the search engine to be ranked

If  $D_{max}$  is the maximum score of document and  $D_{min}$  is the minimum score of the document then normalized score of the search engine is calculated as in [NF03] as

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

The document score, called global normalized score of each document for result merging is calculated as in [NF03] as

$$D'' = \frac{1.0 * D' + 0.4 * C' * D'}{1.4}$$

The documents are ranked based on the value of the document scores. The documents with higher score are displayed to the user.

## 1.4.2 Semisupervised Learning Approach

The semi-supervised learning approach [SCO03a] for result merging broadcast the user query to all the underlying search engines and query is also sent in parallel to the centralized sample search engine and centralized search engine [SC03b] which returns the ranked list of documents score and this document score list is provided to the result merging. The result merging is based on two assumptions. First, when the user poses a query to the selected search engine then some of the documents retrieved from underlying search engine will be also be retrieved from the centralized sample search engine [SC03b]. Second, given specific search engine and search engine independent score for a small number of documents it maps the entire specific search engine score to their corresponding search engine score using the learning function.

### 1.4.3 LMS Merging Technique

LMS merging [RAS03] is use the number of document retrieved from search engines to calculate the Merging Score and it is used to calculate the score of every search engine. In LMS technique, the search engine score is calculated using the proportion of documents retrieved from each search engine. If  $|C|$  is the number of search engine and  $l_i$  is the total number of documents return from  $i^{th}$  search engine then score of the  $i^{th}$  search engine is calculated, as in [RAS03], as

$$S_i = \log \left( 1 + \frac{l_i * K}{\sum_{j=1}^{|C|} l_j} \right)$$

where K is a constant

The score is then used to calculate the weight of the search engine as in [RAS03] as

$$w_i = 1 + \left[ \frac{S_i - \bar{S}}{\bar{S}} \right]$$

Where  $\bar{S}$  is the mean score of search engine scores and  $S_i$  is the score of  $i^{th}$  search engine. The ranked document RD is calculated as  $RD = S_i \times w_i$ . The documents with higher value of  $RD$  are displayed to the user.

### 1.4.4 Abstract Merging approach

The abstract merging approach [LZ+08] is used to merge results retrieved from the underlying search engine with the relevance between user query and abstract information of each search engine. In this approach, first the relevance between abstract with all term in query is calculated. If the term is not present in the query then the relevance between abstract and term will be zero otherwise relevance will be calculated. Using the relevance between term and abstract, the relevance between user query and abstract is computed.



TH-19206

The relevance between user query and abstract is used to rank the documents. The high ranked documents are then displayed to the user.

#### **1.4.5 OWA-Based Merging Approach**

The OWA is an order weight average operator which is use the rank the result return from underlying search engines. The OWA-Based Merging Approach [DDR05] is basically based on fuzzy set theory that uses the quantifier Ordered Weighted Averaging (OWA) operators. The value of quantifier is always greater than zero. In this approach, the position of all documents returned by the search engine is determined and the weight of the search engine is computed. These are then used to generate the rank list of documents and higher ranked documents are displayed to the user.

### **1.5 Aim of the Dissertation**

The dissertation aims to study, implement and compare the following existing algorithms:

1. Search engine selection algorithm query Similarity (qSim) and Modeling Relevance Document Distribution (MRDD)
2. Result merging algorithms Abstract Merging and Order Weight Average

### **1.6 Organization of the Dissertation**

The dissertation is organized as follows: Chapter 2 compares search engine selection algorithm query Similarity (qSim) and Modeling Relevance Document Distribution (MRDD). The result merging algorithms Abstract Merging and Order Weight Average are discussed in chapter 3. Chapter 4 is the conclusion.

## CHAPTER 2

# Search Engine Selection

Search engine selection main task is to identify the most useful search engines that are likely to contain relevant documents for the user query. The objective of search engine selection is to improve efficiency as it would result in sending query to only potentially useful underlying search engines. Several algorithms exist for search engine selection like qSim approach[SLH09], Modeling Relevant Document Distribution (MRDD) [VGL95b][MYL02], Generalization Glossary of Server' Server (gGLOSS)[GGM95], Distributed Web Index Search Engine (D-WISE)[YL96], Collection Retrieval Interface Network (CORI-Net)[CLC95], Savvy Search engine [DH97], ProFusion [FG99], Neural Network approach[RD+86], Relevant Document Distribution Estimation Method for Resource Selection (ReDDE) [SC03b], Light Weight Probes (LWP) Approaches [HT99], Decision theoretic framework (DTF) approach [F97], Learning from Past Queries for resource selection (LPQ) approach [SLH09]. Few of these approaches have been discussed in Chapter 1. A good search engine selection algorithm should identify potential useful search engine that satisfies the information needs of the user query .In this chapter,

the search engine selection algorithms like query Similarity (qSim), Modeling Relevant Document Distribution (MRDD) algorithms are discussed and compared.

### Query Similarity (qsim)

The qSim algorithm [SLH09] use a set of past queries say  $p = \{p_1, p_2, p_3, \dots, p_m\}$ , where  $p_i$  indicate  $i^{th}$  past query and set of search engine  $s = \{s_1, s_2, s_3, \dots, s_n\}$  for a user  $q$  as an input. In step 1, it selects the search engine for every past query and generate a ranked list using round robin merging [VT97] technique of documents return from search engine for each past query  $p_i$ . In step 2, the algorithm computes the percentage of relevant documents for the query  $p_i$  in search engine  $s_j$  i.e.  $Rel\left(\frac{s_j}{p_i}\right)$ . In the next step, the ranked list of documents with respect to the query is generated. In step 4, the similarity  $Sim\left(\frac{p_i}{q}\right)$  between user query and the past queries is computed. This similarity is normalized in step 5. In step 6, the relevance between the search engine and the user query  $Rel\left(\frac{s_j}{q}\right)$  is computed using  $Rel\left(\frac{s_j}{p_i}\right)$  and  $Sim\left(\frac{p_i}{q}\right)$  as

$$Rel\left(\frac{s_j}{q}\right) = \sum Rel\left(\frac{s_j}{p_i}\right) \times Sim\left(\frac{p_i}{q}\right)$$

The search engines are ranked in descending order of  $Rel\left(\frac{s_j}{q}\right)$ . A higher value of  $Rel\left(\frac{s_j}{q}\right)$  implies that the search engine contains most relevant documents with respect to user query  $q$ . Thus, the search engines having higher value for  $Rel\left(\frac{s_j}{q}\right)$  are selected.

The algorithm based on qSim [SLH09] is given in Figure 2.1. The algorithm takes the past queries, user query and the search engines as input and produces ranked list of top-most search engines as output.

**Input:** Let set H [p, q, s], where p is the number of past query, q is the user query, and s is the number of search engine.

**Output:** Ranked list of topmost search engine.

**Method:**

**Step1:** For each  $i^{\text{th}}$  past query

Rank the documents retrieved from all search engines into single ranked list

**Step2:** Compute

$$\text{Rel}\left(\frac{s_j}{p_i}\right) = \sum_{\substack{\text{topT doc in the merged} \\ \text{ranked list}}} \frac{\text{Rel}(s_j/\text{doc})}{T}$$

where T is a pre-defined number of top documents.

$$\text{if } (doc \in s_j) \text{ then } \text{Rel}\left(\frac{s_j}{\text{doc}}\right) = 1$$

$$\text{Otherwise } \text{Rel}\left(\frac{s_j}{\text{doc}}\right) = 0$$

**Step3:** Generate the ranked list  $R_q$  of documents returned from the search engines for the user query q

**Step 4:** For each  $i^{\text{th}}$  past query  $p_i$  and user query q, compute the similarity using

$$\text{Sim}(p_i/q) = \frac{1}{|R_{p_i}|} \sum_{\text{doc} \in R_{p_i} \cap R_q} \text{Score}(\text{doc}, R_{p_i}, R_q)$$

where  $R_{p_i}$  and  $R_q$  are ranked list of past query and user query returned from the search engines and Score function is calculated as

$$\text{Score}(\text{doc}, R_{p_i}, R_q) = 1 - \left| \frac{\text{doc ranked in } R_{p_i}}{|R_{p_i}|} - \frac{\text{doc ranked in } R_q}{|R_q|} \right|$$

**Step5:** Normalized the value of  $\text{Sim}(p_i/q)$  using

$$\text{MAXSIM}_q = \max_i \text{Sim}(p_i/q)$$

$$\text{CUTSIM}_q = 0.8 * \text{MAXSIM}_q$$

$$\text{Normalized } \text{Sim}(p_i/q) = \begin{cases} 0 & \text{if } \text{Sim}(p_i/q) < \text{CUTSIM}_q \\ \frac{\text{Sim}(p_i/q) - \text{CUTSIM}_q}{\text{MAXSIM}_q - \text{CUTSIM}_q}, & \text{otherwise} \end{cases}$$

**Step 6:** For user query q

For (each  $j^{\text{th}}$  search engine) compute

$$\text{Rel}(s_j/q) = \sum \text{Rel}(s_i/p_i) \times \text{Sim}(p_i/q)$$

**Step7:** Ranked the search engine according to the value of  $\text{Rel}(s_j/q)$ . A larger value of  $\text{Rel}(s_j/q)$  is more likely means it contain most relevant documents with respect to the user query q.

Figure 2.1 Query Similarity algorithms based on [SLH09]

### 2.1.1 An Example:

Let the set of training queries be  $PQ = \{PQ1, PQ2, PQ3, \dots, PQ6\}$  and let the set of search engines be  $SE = \{SE1, SE2, \dots, SE10\}$  and a user query be  $q$  are input to the algorithm.

**Step1:** Training queries along with the terms in them are shown in Table 2.1.

PQ1	14	8	7	28	12	30
PQ2	14	26	17	5	6	30
PQ3	2	26	23	22	20	4
PQ4	2	29	11	18	19	7
PQ5	3	15	14	13	10	1
PQ6	18	7	20	27	25	19

Table 2.1

Let the user query UQ have the following terms 8, 10, 21, 38, 42, 40.

For every past query  $PQ_i$ , the search engines selected are shown in Table 2.2.

PQ1	SE1	SE2	SE3	SE4	SE6	SE8	
PQ2	SE1	SE3	SE4	SE5	SE7	SE9	SE10
PQ3	SE2	SE3	SE4	SE6	SE8	SE9	
PQ4	SE2	SE3	SE4	SE5	SE7	SE10	
PQ5	SE3	SE4	SE5	SE8	SE9		
PQ6	SE2	SE4	SE5	SE7	SE9	SE10	

Table 2.2

Past Query **PQ1**= (14, 8, 7, 28, 12, 30) is apply on all search engines and relevant documents are retrieved and are shown in Table 2.3.

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
SE1	38	50	37	23	36	19	44	50	33	46	43	11	10	30	
SE2	47	18	14	33	13	22	20	10	9	45	7	15	49	1	31
SE3	26	38	5	42	16	24	44	48	25	18	13	30	36	26	
SE4	13	6	23	16	15	29	2	28	20	25	16	44	1		
SE6	13	30	33	21	14	25	17	16	45	37	49	6	35		
SE8	38	22	7	45	1	47	14	29	3	2	11	48			

Table 2.3

Single merge list of documents return by six search engine for past query PQ1 using Round Robin algorithms [VT97] is shown in Table 2.4.



RD	38	47	26	13	50	18	6	30	22	37	14	5	23	33	7	42	.	.	TD
----	----	----	----	----	----	----	---	----	----	----	----	---	----	----	---	----	---	---	----

Table 2.4

Now the top 15 documents are selected from the ranked list of documents as shown in Table 2.5.

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
38	47	26	13	50	18	6	30	22	37	14	5	23	33	7

Table 2.5

Next Step is to compute  $Rel(s_j|PQ_1)$ , between search engine and past query PQ1. These are shown in Table 2.6 for  $T=15$

$Rel(s_1 PQ_1)$	$Rel(s_2 PQ_1)$	$Rel(s_3 PQ_1)$	$Rel(s_4 PQ_1)$	$Rel(s_6 PQ_1)$	$Rel(s_8 PQ_1)$
0.4000	0.4000	0.466	0.200	0.4000	0.2666

Table 2.6

**PQ2= (14, 26, 17, 5, 6, 30)** is applied on all search engines and documents are retrieved as shown in Table 2.7

	D1	D	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
SE1	44	16	36	43	39	9	8	25	28	11	13	20	40	27	17
SE3	6	40	50	41	26	32	43	27	48	2	20	47	23	22	18
SE4	18	40	9	10	2	29	17	44	30	23	22	38	43	11	41
SE5	13	40	6	50	38	14	37	41	1	42	20	17	27		
SE7	29	18	46	3	26	8	1	34	50	24	42	48	37	6	27
SE9	31	3	25	17	18	41	9	45	48	23	26	29	19	39	
SE10	3	44	12	2	40	45	9	27	15	25	31	16			

Table 2.7

Single merge list of documents return by the seven search engines for past query PQ2 using Round Robin algorithms [VT97] is shown in Table 2.8

RD	44	6	18	13	29	31	3	16	40	36	50	9	46	25	12	.	.	.	TD
----	----	---	----	----	----	----	---	----	----	----	----	---	----	----	----	---	---	---	----

Table 2.8

Top 15 documents from ranked list of documents are shown in Table 2.9

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
44	6	18	13	29	31	3	16	40	36	50	9	46	25	12

Table 2.9

Next Step computes  $Rel(s_j|PQ_2)$  between search engine and past query. This is shown in Table 2.10 for  $T=15$  for past query PQ2

$Rel(s_1 PQ_2)$	$Rel(s_3 PQ_2)$	$Rel(s_4 PQ_2)$	$Rel(s_5 PQ_2)$	$Rel(s_7 PQ_2)$	$Rel(s_9 PQ_2)$	$Rel(s_{10} PQ_2)$
0.466	0.266	0.333	.266	0.400	0.333	0.533

Table 2.10

**PQ3= (2, 26, 23, 22, 20, 4)** is applied to all six search engines. The documents retrieved are shown in Table 2.11

	D1	D	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
SE2	7	10	45	1	12	39	9	21	41	44	6	29	42		
SE3	32	34	26	36	11	9	4	12	40	42	37	48	17		
SE4	18	46	14	26	41	48	37	27	24	36	49	31	44		
SE6	17	10	27	20	16	8	47	11	16	30	5	40	26	14	
SE8	29	15	4	3	17	37	43	35	38	28	40				
SE9	44	25	21	18	19	11	35	38	34	50	21	41	24		

Table 2.11

Single merge list of documents return by six search engines for past query PQ3 using Round Robin algorithms [VT97] are shown in Table 2.12

RD	7	32	18	17	29	44	10	34	46	15	25	45	26	14	27	21	.	.	LD
----	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	----

Table 2.12

The top 15 documents selected from the ranked list of documents are shown in Table 2.13

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
7	32	18	17	29	44	10	34	46	15	25	45	26	14	27

Table 2.13

In the next step, the  $Rel(s_j|PQ_3)$  is computed between search engines and past query PQ3. This is shown in Table 2.14 for  $T=15$ .

$Rel(s_2 PQ_3)$	$Rel(s_3 PQ_3)$	$Rel(s_4 PQ_3)$	$Rel(s_6 PQ_3)$	$Rel(s_8 PQ_3)$	$Rel(s_9 PQ_3)$
0.333	0.266	0.400	0.333	0.200	0.266

Table 2.14

**PQ4= (2, 29, 11, 18, 19, 7)** is applied on the search engines and documents are retrieved and are shown in Table 2.15

	D1	D	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
SE2	42	14	35	19	29	23	30	13	36	44	28	38	27		
SE3	47	48	42	29	24	25	5	23	36	15	38	37	43		
SE4	17	4	50	20	26	14	29	44	15	1	7	43	36	34	
SE5	41	36	33	32	46	29	27	34	8	19	20	12	9	7	
SE7	27	12	36	19	11	44	39	18	46	41	16	23			
SE10	24	20	47	21	17	4	12	18	38	16	50	6	26	48	

Table 2.15

Single merge list of documents returned by six search engines for past query PQ4 using Round Robin algorithms [VT97] is shown in Table 2.16

RD	42	47	17	41	27	24	14	48	4	36	12	20	35	50	33	TD
----	----	----	----	----	----	----	----	----	---	----	----	----	----	----	----	----

Table 2.16

To select top 15 documents from ranked list of documents are shown in Table 2.17

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
42	47	17	41	27	24	14	48	4	36	12	20	35	50	33

Table 2.17

Next Step computes  $Rel(s_j|PQ_4)$  between search engine and past query PQ4. This is shown in Table 2.18 for  $T=15$ .

$Rel(s_2 PQ_4)$	$Rel(s_3 PQ_4)$	$Rel(s_4 PQ_4)$	$Rel(s_5 PQ_4)$	$Rel(s_7 PQ_4)$	$Rel(s_{10} PQ_4)$
0.333	0.333	0.400	0.400	0.266	0.533

Table 2.18

PQ5 = (3, 15, 14, 13, 10, 1) applied on search engines and documents are retrieved as shown in Table 2.19

	D1	D	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
SE3	4	31	8	42	26	21	38	27	10	22	28	20			
SE4	28	17	8	40	16	31	42	12	45	25	36	6	21	39	
SE5	15	45	8	36	40	7	47	25	5	35	30	19	11	48	
SE8	25	7	5	24	12	20	4	33	31	14	43	46			
SE9	35	6	23	36	12	15	3	47	41	50	34	11	13	44	20

Table 2.19

Single merge list of documents return by six search engine for past query PQ5 using Round Robin algorithms [VT97] is shown in Table 2.20

RD	4	28	15	25	35	31	17	45	7	6	8	5	23	42	40	12	.	.	TD
----	---	----	----	----	----	----	----	----	---	---	---	---	----	----	----	----	---	---	----

Table 2.20

Top 15 documents are selected from ranked list of documents as shown in Table 2.21

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
4	28	15	25	35	31	17	45	7	6	8	5	23	42	40

Table 2.21

Next Step Computes  $Rel(s_j|PQ_5)$  between search engine and past query PQ5 are shown in Table 2.22 for  $T=15$

$Rel(s_3 PQ_5)$	$Rel(s_4 PQ_5)$	$Rel(s_5 PQ_5)$	$Rel(s_8 PQ_5)$	$Rel(s_9 PQ_5)$
0.333	0.600	0.533	0.333	0.266

Table 2.22

**PQ6= (18, 7, 20, 27, 25, 19)** is applied to six search engines and documents are retrieved as shown in Table 2.23

	D1	D	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
SE2	40	30	24	9	41	5	26	21	8	45	48	14	1	15	
SE4	6	47	14	13	36	33	48	43	26	3	38	29	34	45	5
SE5	8	29	13	10	43	11	20	47	31	7	50	49	21	30	
SE7	18	1	25	31	40	13	30	38	36	44	12	34	16	8	
SE9	43	42	12	11	17	31	15	20	22	5	21	48	4		
SE10	30	31	3	41	50	10	36	40	37	21	20	50	27	24	

Table 2.23

Single merge list of documents return by six search engines for past query PQ6 using Round Robin algorithms [VT97] is shown in Table 2.24

RD	40	6	8	18	43	30	47	29	1	42	31	24	14	13	25	12	.	.	TD
----	----	---	---	----	----	----	----	----	---	----	----	----	----	----	----	----	---	---	----

Table 2.24

Top 15 documents from among the ranked list of documents are shown in Table 2.25

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
40	6	8	18	43	30	47	29	1	42	31	24	14	13	25

Table 2.25

In the next step,  $Rel(s_j|PQ_6)$  is computed between search engines and past query PQ6 are shown in Table 2.26 for  $T=15$

$Rel(s_2 PQ_6)$	$Rel(s_4 PQ_6)$	$Rel(s_5 PQ_6)$	$Rel(s_7 PQ_6)$	$Rel(s_9 PQ_6)$	$Rel(s_{10} PQ_6)$
0.400	0.400	0.466	0.533	0.200	0.266

Table 2.26

**Step3:** UQ= (8, 10, 21, 38, 42, 40) applied on the selected search engines and documents retrieved are shown in Table 2.27

	D1	D	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
SE1	32	30	25	31	11	27	25	33	8	37	43	2	23	6	
SE3	34	47	43	19	8	35	21	9	37	40	2	3	50		
SE4	37	30	11	3	10	25	44	32	4	26	16	43	9	27	
SE6	45	1	28	25	3	27	14	48	29	44	9	15	34	36	
SE9	50	7	32	10	23	11	5	3	47	20	17	43			
SE10	39	44	2	7	15	29	6	47	50	24	18	45			

Table 2.27

Single merge list of documents return by the six search engines for the user query UQ using Round Robin algorithms [VT97] are shown in Table 2.28

RD	32	34	37	45	50	39	30	47	1	7	44	25	43	11	28	2	.	.	TD
----	----	----	----	----	----	----	----	----	---	---	----	----	----	----	----	---	---	---	----

Table 2.28

Top 15 documents from ranked list of documents is shown in Table 2.29

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
32	34	37	45	50	39	30	47	1	44	25	43	11	28	2

Table 2.29

**Step4:** Now find the similarity between user query and all past queries using is shown in Table 2.30

$$\text{Sim}(PQ_i/q) = \frac{1}{|R_{PQ_i}|} \sum_{\text{doc} \in R_{PQ_i} \cap R_q} \text{Score}(\text{doc}, R_{PQ_i}, R_q),$$

and the score is defined as

$$\text{Score}(\text{doc}, R_{PQ_i}, R_q) = 1 - \left| \frac{\text{doc ranked in } R_{PQ_i}}{|R_{PQ_i}|} - \frac{\text{doc ranked in } R_q}{|R_q|} \right|,$$

where  $|R_{PQ_i}|$  and  $|Rq|$  are the number of documents in merged ranked list with respect to the training queries and user query respectively.

Here  $|R_{PQ_i}|=15$  and  $|Rq|=15$

$Sim(PQ_1 q)$	$Sim(PQ_2 q)$	$Sim(PQ_3 q)$	$Sim(PQ_4 q)$	$Sim(PQ_5 q)$	$Sim(PQ_6 q)$
0.2044	0.1200	0.2488	0.0666	0.097	0.2755

Table 2.30

**Step 5:**  $Sim(PQ_i|q)$  is normalized as shown in Table 2.31 using the following

$$MAXSIM_q = 0.2755$$

$$CUTMAX_q = 0.8 * 0.2755 = 0.2204$$

$$\text{Normalized } Sim(PQ_i|q) = \begin{cases} 0 & \text{if } Sim(PQ_i|q) < CUTSIM_q \\ \frac{Sim(PQ_i|q) - CUTSIM_q}{MAXSIM_q - CUTSIM_q}, & \text{otherwise} \end{cases}$$

<i>Normalized Sim(PQ<sub>1</sub> q)</i>	0
<i>Normalized Sim(PQ<sub>2</sub> q)</i>	0
<i>Normalized Sim(PQ<sub>3</sub> q)</i>	0.5154
<i>Normalized Sim(PQ<sub>4</sub> q)</i>	0
<i>Normalized Sim(PQ<sub>5</sub> q)</i>	0
<i>Normalized Sim(PQ<sub>6</sub> q)</i>	1

Table 2.31

**Step 6:** Find the value of  $Rel(s_j/q)$  are shown in Table 2.32 using

$$Rel(s_j/q) = \sum Rel(s_j/p_i) \times Sim(PQ_i/q)$$

Rel(s <sub>1</sub> /q)	0.000
Rel(s <sub>2</sub> /q)	0.5716
Rel(s <sub>3</sub> /q)	0.1325
Rel(s <sub>4</sub> /q)	0.6061
Rel(s <sub>5</sub> /q)	0.4000
Rel(s <sub>6</sub> /q)	0.1716
Rel(s <sub>7</sub> /q)	0.5333
Rel(s <sub>8</sub> /q)	0.1030
Rel(s <sub>9</sub> /q)	0.3370
Rel(s <sub>10</sub> /q)	0.2660

Table 2.32

The search engines are ranked according to the value of  $Rel(s_j/q)$  in descending order.

The search engines as per their rank are shown in Table 2.33

SE4	SE2	SE7	SE5	SE9	SE10	SE6	SE3	SE8	SE1
0.6061	0.5716	0.5333	0.4000	0.3370	0.2660	0.1716	0.1325	0.1030	0.0000

Table 2.33

The top ranked search engines are selected for retrieving the relevant documents for the user query.

## 2.2 Modeling Relevant Document Distribution (MRDD) Approach

The Modeling Relevant Document Distribution (MRDD) approach[VGL95b][MYL02] is a learning based approach for search engine selection, which uses a set of training queries to identify the most appropriate search engines. Each training query is applied to each search engine and documents are retrieved from the underlying search engine. The relevant documents among them form the distribution vector. Using the cosine distance similarity function, the similarity between user query and training queries is computed and then k- most similar training queries are considered. For each search engine average relevant document distribution vector over the k vector corresponding k-most similar

training queries and search engine is obtained. The average distribution vectors are use to identify the appropriate search engine for retrieving documents for the user query. The algorithm based on MRDD [VGL95b][MYL02] is shown in Figure 2.2

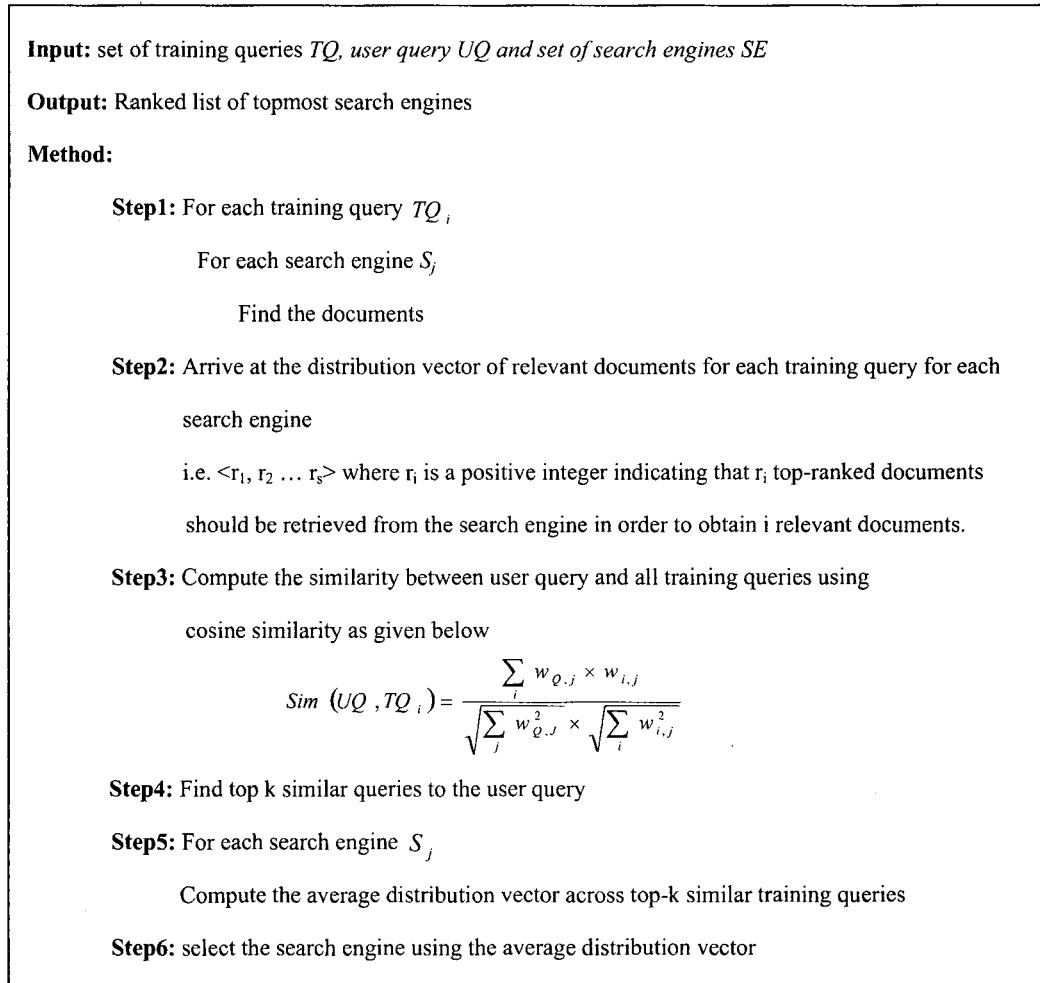


Figure 2.2 MRDD algorithms based on [VGL95b] [MYL02]

### 2.2.1 An Example

Let us consider four search engines each containing fifteen documents and let there be eight training queries with six terms as shown in Table 2.34. Each training query is applied to every search engine and documents are retrieved.



Let eight training queries be as shown in Table 2.34

TQ1	1	0	1	0	0	1
TQ2	1	1	1	1	1	0
TQ3	1	0	1	1	0	0
TQ4	0	0	0	1	1	0
TQ5	0	0	1	0	0	1
TQ6	0	1	1	1	1	0
TQ7	1	0	0	0	0	1
TQ8	0	0	1	0	1	1

Table 2.34

Training Query **TQ1**= (1, 0, 1, 0, 0, 1) is applied on four search engines and the retrieved documents are shown in each search Engine is shown in Table 2.35

SE1	1	1	0	1	0	1	1	1	1	1	0	0	1	0	1
SE2	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
SE3	1	0	0	1	1	1	0	0	0	1	0	0	0	1	0
SE4	1	0	1	0	0	0	0	1	1	0	0	1	1	0	1

Table 2.35

Distribution vector (DV) for training query **TQ1**= (1, 0, 1, 0, 0, 1) for each search engine is shown in Table 2.36

Search Engine	Distribution Vector
SE1	1, 2, 4, 6, 7, 8, 9, 10, 13, 15
SE2	3, 4, 5
SE3	1, 4, 5, 6, 10, 14
SE4	1, 3, 8, 9, 12, 13, 15

Table 2.36

Training Query **TQ2** = (1, 1, 1, 1, 1, 0) is applied on four search engines and the retrieved documents are shown in Table 2.37.

SE1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	0
SE2	0	1	0	0	0	0	1	1	1	1	1	0	1	0	0
SE3	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1
SE4	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0

Table 2.37

Distribution vector (DV) for training query **TQ2** = (1, 1, 1, 1, 1, 0) for each search engine is shown in Table 2.38

Search Engine	Distribution Vector
SE1	1, 3, 5, 8, 9, 10, 11
SE2	2, 7, 8, 9, 10, 11, 13
SE3	1, 2, 5, 6, 7, 10, 11, 12, 13, 14, 15
SE4	1, 2, 11, 12, 13, 14

Table 2.38

Training Query **TQ3**= (1, 0, 1, 1, 0, 0) is applied on the four search engines and the retrieved documents are shown in Table 2.39

SE1	1	0	0	0	0	0	0	1	1	0	0	0	0	1	1
SE2	1	1	1	1	0	0	1	0	1	0	1	1	1	1	0
SE3	0	1	1	1	0	0	1	1	1	0	0	1	1	1	1
SE4	0	1	0	1	1	1	1	1	1	0	0	0	1	0	1

Table 2.39

The distribution vector for training query **TQ3**= (1, 0, 1, 1, 0, 0) for each search engine is shown in Table 2.40

Search Engine	Distribution Vector
SE1	1, 8, 9, 14, 15
SE2	1, 2, 3, 4, 7, 9, 11, 12, 13, 14
SE3	2, 3, 4, 7, 8, 9, 12, 13, 14, 15
SE4	2, 4, 5, 6, 7, 8, 9, 13, 15

Table 2.40

Training Query **TQ4**= (0, 0, 0, 1, 1, 0) is applied on the four search engines and retrieved documents are shown in Table 2.41

SE1	1	0	1	0	1	1	1	1	1	0	0	1	1	0	1
SE2	0	0	1	1	1	0	1	0	0	0	0	0	0	1	1
SE3	0	0	1	0	0	0	0	0	0	1	1	0	1	0	1
SE4	0	0	1	0	1	0	1	1	0	0	0	0	0	1	1

Table 2.41

Distribution vector (DV) for training query **TQ4**= (0, 0, 0, 1, 1, 0) over all search engine is shown in Table 2.42

Search Engine	Distribution Vector
SE1	1, 3, 5, 6, 7, 8, 9, 12, 13, 15
SE2	3, 4, 5, 7, 14, 15
SE3	3, 10, 11, 13, 15
SE4	3, 5, 7, 8, 14, 15

Table 2.42

Training Query  $TQ5 = (0, 0, 1, 0, 0, 1)$  is applied on the four search engines and the retrieved documents are shown in Table 2.43

SE1	0	1	0	1	1	1	0	1	1	1	0	0	1	0	1
SE2	0	1	0	1	0	0	1	1	1	0	1	1	1	0	1
SE3	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1
SE4	0	1	1	1	0	0	1	1	0	0	1	0	1	1	1

Table 2.43

Distribution vector for the training query  $TQ5 = (0, 0, 1, 0, 0, 1)$  for each search engine is shown in Table 2.44

Search Engine	Distribution Vector
SE1	2, 4, 5, 6, 8, 9, 10, 13, 15
SE2	2, 4, 7, 8, 9, 11, 12, 13, 15
SE3	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15
SE4	2, 3, 4, 7, 8, 11, 13, 14, 15

Table 2.44

Training Query  $TQ6 = (0, 1, 1, 1, 1, 0)$  is applied on the four search engines and the retrieved documents are shown in Table 2.45

SE1	1	0	1	1	1	1	1	0	0	1	1	0	1	0	1
SE2	1	0	1	0	0	1	0	0	0	1	0	0	0	1	0
SE3	0	0	0	0	0	1	1	0	0	0	1	1	0	1	1
SE4	1	0	0	1	0	0	1	1	1	0	0	1	0	1	0

Table 2.45

Distribution vector for training query  $TQ6 = (0, 1, 1, 1, 1, 0)$  for each search engine is shown in Table 2.46

Search Engine	Distribution Vector
SE1	1, 3, 4, 5, 6, 7, 10, 11, 13, 15
SE2	1, 3, 6, 10, 14
SE3	6, 7, 11, 12, 14, 15
SE4	1, 4, 7, 8, 9, 12, 14

Table 2.46

Training Query  $TQ7 = (1, 0, 0, 0, 0, 1)$  is applied on the four search engines and the retrieved documents are shown in Table 2.47

SE1	1	0	1	0	1	1	0	0	1	1	0	0	0	0	0
SE2	1	1	1	1	0	0	0	1	0	0	0	0	1	1	0
SE3	1	1	1	0	1	0	0	1	1	1	1	0	1	1	1
SE4	0	1	1	0	0	1	0	1	1	0	1	1	0	0	0

Table 2.47

Distribution vector for the training query  $TQ7 = (1, 0, 0, 0, 0, 1)$  for each search engine is shown in Table 2.48

Search Engine	Distribution Vector
SE1	1, 3, 5, 6, 9, 10,
SE2	1, 2, 3, 4, 8, 13, 14
SE3	1, 2, 3, 5, 8, 9, 10, 11, 13, 14, 15
SE4	2, 3, 6, 8, 9, 11, 12

Table 2.48

Training Query  $TQ8 = (0, 0, 1, 0, 1, 1)$  is applied on the four search engines and the retrieved documents are shown in Table 2.49

SE1	1	1	1	0	1	0	0	0	0	1	0	0	1	1	0
SE2	0	1	1	0	0	1	0	1	1	1	0	0	1	1	0
SE3	0	0	1	1	1	0	0	1	1	1	1	1	1	0	0
SE4	0	1	0	0	1	1	0	0	0	1	0	1	0	0	0

Table 2.49

Distribution vector for training query  $TQ8 = (0, 0, 1, 0, 1, 1)$  for each search engine is shown in Table 2.50

Search Engine	Distribution Vector
SE1	1, 2, 3, 5, 10, 13, 14
SE2	2, 3, 6, 8, 9, 10, 13, 14
SE3	3, 4, 5, 8, 9, 10, 11, 12, 13
SE4	2, 5, 6, 10, 12

Table 2.50

In next step, the cosine similarity between user query and training queries are shown in Table 2.51 using the cosine similarity measure given by

$$Sim(UQ, TQ_i) = \frac{\sum_j w_{Q,j} \times w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2} \times \sqrt{\sum_j w_{i,j}^2}}$$

User Query	0	1	1	0	1	0	Cosine similarity
TQ1	1	0	1	0	0	1	$Sim(UQ, TQ_1)=0.6667$
TQ2	1	1	1	1	1	0	$Sim(UQ, TQ_2)=0.2254$
TQ3	1	0	1	1	0	0	$Sim(UQ, TQ_3)=0.6667$
TQ4	0	0	0	1	1	0	$Sim(UQ, TQ_4)=0.5918$
TQ5	0	0	1	0	0	1	$Sim(UQ, TQ_5)=0.5918$
TQ6	0	1	1	1	1	0	$Sim(UQ, TQ_6)=0.1340$
TQ7	1	0	0	0	0	1	$Sim(UQ, TQ_7)=1.0000$
TQ8	0	0	1	0	1	1	$Sim(UQ, TQ_8)=0.3333$

Table 2.51

The top-k (k=4) training queries selected are shown in Table 2.52

TQ	Cosine Similarity
TQ6	0.1340
TQ2	0.2254
TQ8	0.3333
TQ4	0.5918

Table 2.52

Now find the distribution vector across the all search engine over top-k training queries are shown in Table 2.53

	DV of SE1	DV of SE2	DV of SE3	DV of SE4
TQ6	1, 3, 4, 5, 6, 7, 10, 11, 13,	1, 3, 6, 10, 14	6, 7, 11, 12, 14, 15	1, 4, 7, 8, 9, 12,
TQ2	1, 4, 5, 8, 9, 10, 11	2, 7, 8, 9, 10, 11, 13	1, 2, 5, 6, 7, 10, 11, 12, 13, 14,	1, 2, 11, 12, 13,
TQ8	1, 2, 3, 5, 10, 13, 14	2, 3, 6, 8, 9, 10, 13,	3, 4, 5, 8, 9, 10, 11, 13	2, 5, 6, 10, 12
TQ4	1, 3, 5, 6, 7, 8, 9, 12, 13, 15	3, 4, 5, 7, 14, 15	3, 10, 11, 13, 15	3, 5, 7, 8, 14, 15

Table 2.53

In the next step, the average distribution vector (ADV) for all search engine are shown in

Table 2.54

ADV1	1.0, 1.25, 2.25, 2.75, 3.5, 4.0, 4.5, 5.0, 5.5, 6.25, 6.5, 6.75, 7.5, 7.75, 8.25
ADV2	0.25, 0.5, 1.25, 1.5, 1.75, 2.5, 3.0, 3.25, 3.75, 4.5, 4.5, 4.5, 5.0, 5.75, 6.0
ADV3	.25, .50, 1.0, 1.5, 1.75, 2.25, 2.75, 3.0, 3.5, 4.25, 5.25, 5.5, 6.25, 6.75, 7.5
ADV4	.50, .75, 1.0, 1.25, 1.75, 2.0, 2.5, 3.0, 3.25, 3.5, 3.75, 4.5, 4.75, 5.5, 5.75

Table 2.54

The list of selected search engines for top-k (where k= 4, 5, 6, 7, 8) documents are shown

in Table 2.55

For top k document	List of Selected Search Engines
K=4	SE2, SE3
K=5	SE2, SE3, SE4
K=6	SE2, SE3, SE4
K=7	SE1, SE2, SE3, SE4
K=8	SE1, SE2, SE3, SE4

Table 2.55

### 2.3 Example Based Comparison

Let the five past queries along with six terms each is shown in Table 2.56. Let us consider

in all there are six search engines with eight documents in each of them.

PQ1	2	1	5	2	7	3
PQ2	0	5	3	1	7	8
PQ3	3	1	0	2	0	2
PQ4	2	0	4	3	3	0
PQ5	5	8	1	8	3	7

Table 2.56

Let the user query UQ have the following terms 3, 0, 1, 6, 8, 5

Let us first apply the qSim algorithm.

Let the search engine selected from amongst six search engines be as shown in Table 2.57.

PQ1	SE1	SE2	SE4	SE5	SE6
PQ2	SE2	SE3	SE4	SE5	SE6
PQ3	SE1	SE2	SE3	SE4	SE5
PQ4	SE1	SE3	SE4	SE5	SE6
PQ5	SE1	SE2	SE4	SE5	SE6

Table 2.57

Past query  $PQ1 = (2, 1, 5, 2, 7, 3)$  is apply to all selected search engines and the relevant documents retrieved are as shown in Table 2.58

SE1	6	7	2	4	4	1	7	0
SE2	6	8	2	1	0	5	4	5
SE4	5	6	0	1	2	6	6	5
SE5	1	2	4	3	0	4	2	5
SE6	8	4	2	0	3	3	7	8

Table 2.58

The top-8 documents from the single merge list of documents return by five search engines for past query PQ1 using Round Robin algorithms [VT 97] are shown in Table 2.59

RD	6	5	1	8	7	2	4	3
----	---	---	---	---	---	---	---	---

Table 2.59

Next  $Rel(s_j|PQ_1)$ , is computed between search engine and the past query PQ1. These are shown in Table 2.60 where  $T=8$ .

$Rel(s_1 PQ_1)$	$Rel(s_2 PQ_1)$	$Rel(s_4 PQ_1)$	$Rel(s_5 PQ_1)$	$Rel(s_6 PQ_1)$
0.6250	0.7500	0.5000	0.6250	0.6250

Table 2.60

Past query  $PQ 2 = (0, 5, 3, 1, 7, 8)$  is applied to all selected search engines and the documents retrieved are shown in Table 2.61

SE2	6	0	6	1	8	4	4	2
SE3	2	8	5	4	8	2	8	5
SE4	4	1	2	0	7	6	7	8
SE5	0	7	1	7	5	1	7	7
SE6	0	1	7	0	2	7	5	8

Table 2.61

The top-8 documents from single merge list of documents returned by the five search engines for past query PQ2 using Round Robin algorithms [VT 97] are shown in Table 2.62

RD	6	2	4	8	1	7	3	0
----	---	---	---	---	---	---	---	---

Table 2.62

Next,  $Rel(s_j|PQ_i)$  is computed between the search engines and the past query PQ2. These are shown in Table 2.63 where  $T=8$ .

$Rel(s_2 PQ_2)$	$Rel(s_3 PQ_2)$	$Rel(s_4 PQ_2)$	$Rel(s_5 PQ_2)$	$Rel(s_6 PQ_2)$
0.6250	0.5000	0.7500	0.3750	0.6250

Table 2.63

Past query PQ3= (3, 1, 0, 2, 0, 2) is applied on all the selected search engines and the documents retrieved are shown in Table 2.64

SE1	5	1	2	7	5	5	0	1
SE2	5	3	1	7	8	6	3	4
SE3	4	5	4	7	6	5	1	5
SE4	7	6	3	5	0	3	3	3
SE5	5	1	2	5	2	4	1	4

Table 2.64

The top-8 documents from the single merge list of documents returned by the five search engines for past query PQ3 using Round Robin algorithms [VT 97] are shown in Table 2.65

RD	5	4	7	1	3	6	8	2
----	---	---	---	---	---	---	---	---

Table 2.65

Next,  $Rel(s_j|PQ_i)$  is computed between the selected search engines and the past query PQ3 are shown in Table 2.66 where  $T=8$ .



$Rel(s_1 PQ_3)$	$Rel(s_2 PQ_3)$	$Rel(s_3 PQ_3)$	$Rel(s_4 PQ_3)$	$Rel(s_5 PQ_3)$
0.5000	0.8750	0.6250	0.5000	0.5000

Table 2.66

Past query **PQ4= (2, 0, 4, 3, 3, 0)** is applied to all search engines and the documents retrieved are shown in Table 2.67

SE1	1	5	1	5	7	3	0	5
SE3	8	8	2	7	5	4	7	7
SE4	8	6	8	5	5	3	0	5
SE5	3	8	1	6	3	2	4	3
SE6	7	7	6	4	7	8	0	6

Table 2.67

The top-8 documents from the single merge list of documents returned by the five selected search engines for past query PQ4 using Round Robin algorithms [VT 97] are shown in Table 2.68

RD	1	8	3	7	5	6	2	4
----	---	---	---	---	---	---	---	---

Table 2.68

Next,  $Rel(s_j|PQ_4)$  is computed between the selected search engines and the past query PQ4 are shown in Table 2.69 where  $T=8$

$Rel(s_1 PQ_4)$	$Rel(s_3 PQ_4)$	$Rel(s_4 PQ_4)$	$Rel(s_5 PQ_4)$	$Rel(s_6 PQ_4)$
0.5000	0.6250	0.5000	0.7500	0.5000

Table 2.69

Past query **PQ5= (5, 8, 1, 8, 3, 7)** is applied on all the selected search engines and the documents retrieved are shown in Table 2.70

SE1	1	6	0	3	7	4	8	2
SE2	8	7	8	2	0	7	6	5
SE4	8	7	7	6	7	3	1	1
SE5	6	4	7	0	0	3	1	7
SE6	2	3	2	8	5	1	5	0

Table 2.70

The top-8 documents from the single merge list of documents returned by the five selected search engines for past query PQ5 using Round Robin algorithms [VT 97] are shown in Table 2.71

RD	1	8	6	2	7	4	3	5
----	---	---	---	---	---	---	---	---

Table 2.71

Next,  $Rel(s_j|PQ_i)$  is computed between the selected search engines and the past query PQ5 and are shown in Table 2.72 where  $T=8$ .

$Rel(s_1 PQ_5)$	$Rel(s_2 PQ_5)$	$Rel(s_4 PQ_5)$	$Rel(s_5 PQ_5)$	$Rel(s_6 PQ_5)$
0.8750	0.6250	0.6250	0.6250	0.6250

Table 2.72

User query UQ= (3, 0, 1, 6, 8, 5) is applied on the selected search engines and the documents retrieved are shown in Table 2.73

SE1	8	3	8	2	1	2	1	2
SE2	4	7	2	2	3	5	6	1
SE3	3	7	4	4	3	8	6	5
SE4	1	6	8	7	2	3	5	4
SE5	6	2	5	1	8	7	4	5
SE6	1	3	3	4	6	5	6	3

Table 2.73

The top-8 documents from the single merge list of documents returned by the six search engines for the user query UQ using Round Robin algorithms [VT 97] are shown in Table 2.74

RD	8	4	3	1	6	7	2	5
----	---	---	---	---	---	---	---	---

Table 2.74

Now the similarity between the user query and all the past queries are shown in Table 2.75 where

$$|R_{PQ_i}| = 8 \text{ and } |R_q| = 8$$

$Sim(PQ_1 q)$	$Sim(PQ_2 q)$	$Sim(PQ_3 q)$	$Sim(PQ_4 q)$	$Sim(PQ_5 q)$
0.4178	0.4000	0.4444	0.4622	0.4533

Table 2.75

Using step 5, the normalized value of  $Sim(PQ_i|q)$  is computed and are shown in Table 2.76

<i>Normalized Sim</i> ( $PQ_1 q$ )	0.5192
<i>Normalized Sim</i> ( $PQ_2 q$ )	0.3269
<i>Normalized Sim</i> ( $PQ_3 q$ )	0.8077
<i>Normalized Sim</i> ( $PQ_4 q$ )	1.0000
<i>Normalized Sim</i> ( $PQ_5 q$ )	0.9038

Table 2.76

In the next step, the  $Rel(s_j/q)$  is computed and are shown in Table 2.77

$Rel(s_1/q)$	2.2236
$Rel(s_2/q)$	2.4495
$Rel(s_3/q)$	2.0745
$Rel(s_4/q)$	2.1659
$Rel(s_5/q)$	1.9976
$Rel(s_6/q)$	1.8918

Table 2.77

The search engines are ranked according the value of  $Rel(s_j/q)$ . The search engine in descending order of their ranks are shown in Table 2.78

SE2	SE1	SE4	SE3	SE5	SE6
2.4495	2.2236	2.1659	2.0745	1.9976	1.8918

Table 2.78

The top ranked search engines are then selected by the metasearch engine.

Let us now apply the MRDD algorithm.

Training Query  $TQ1 = (2, 1, 5, 2, 7, 3)$  is applied to all the search engines and the retrieved documents from each search engine are shown in Table 2.79

SE1	0	4	4	0	7	5	0	6
SE2	3	4	5	5	6	6	7	6
SE3	8	4	2	0	0	3	1	8
SE4	0	3	5	3	4	5	1	3
SE5	7	6	8	8	2	5	4	4
SE6	5	8	7	0	5	0	3	6

Table 2.79

Distribution vector (DV) for training query  $TQ1 = (2, 1, 5, 2, 7, 3)$  for each search engine is shown in Table 2.80

	DV
SE1	2, 3, 5, 6, 8
SE2	1, 2, 3, 4, 5, 6, 7, 8
SE3	1, 2, 3, 6, 7, 8
SE4	2, 3, 4, 5, 6, 7, 8
SE5	1, 2, 3, 4, 5, 6, 7, 8
SE6	1, 2, 3, 5, 7, 8

Table 2.80

Training Query  $TQ2 = (0, 5, 3, 1, 7, 8)$  is applied to all the search engines and the retrieved documents from each search engine are shown in Table 2.81

SE1	7	6	1	3	6	8	8	8
SE2	8	5	5	3	3	6	7	1
SE3	3	2	2	8	3	8	2	2
SE4	3	6	3	5	3	1	4	5
SE5	1	2	4	4	4	1	5	4
SE6	3	0	2	2	6	6	8	0

Table 2.81

Distribution vector (DV) for training query  $TQ2 = (0, 5, 3, 1, 7, 8)$  for each search engine is shown in Table 2.82

	DV
SE1	1, 2, 3, 4, 5, 6, 7, 8
SE2	1, 2, 3, 4, 5, 6, 7, 8
SE3	1, 2, 3, 4, 5, 6, 7, 8
SE4	1, 2, 3, 4, 5, 6, 7, 8
SE5	1, 2, 3, 4, 5, 6, 7, 8
SE6	1, 3, 4, 5, 6, 7

Table 2.82

Training Query **TQ3= (3, 1, 0, 2, 0, 2)** is applied on all the search engines and the retrieved documents by each search engine are shown in Table 2.83

SE1	1	1	2	6	8	7	0	3
SE2	2	8	6	8	3	2	0	7
SE3	0	5	1	3	4	4	0	8
SE4	5	5	0	3	8	6	0	8
SE5	3	0	4	6	1	4	4	0
SE6	5	4	3	6	0	2	2	8

Table 2.83

Distribution vector (DV) for training query **TQ3= (3, 1, 0, 2, 0, 2)** for each search engine is shown in Table 2.84

	DV
SE1	1, 2, 3, 4, 5, 6, 8
SE2	1, 2, 3, 4, 5, 6, 8
SE3	2, 3, 4, 5, 6, 8
SE4	1, 2, 4, 5, 6, 8
SE5	1, 3, 4, 5, 6, 7
SE6	1, 3, 4, 6, 7, 8

Table 2.84

Training Query **TQ4= (2, 0, 4, 3, 3, 0)** is applied on all the search engines and the retrieved documents by each search engine are shown in Table 2.85

SE1	2	2	5	7	2	2	0	8
SE2	0	7	5	4	7	6	6	5
SE3	7	3	0	4	0	4	8	0
SE4	5	2	1	4	5	8	4	7
SE5	5	6	1	3	6	4	3	5
SE6	0	1	6	1	6	2	6	4

Table 2.85

Distribution vector (DV) for training query **TQ4= (2, 0, 4, 3, 3, 0)** for each search engine is shown in Table 2.86

	DV
SE1	1, 2, 3, 5, 6, 8
SE2	2, 3, 4, 5, 6, 7, 8
SE3	1, 2, 4, 6, 7
SE4	1, 2, 3, 4, 5, 6, 7, 8
SE5	1, 2, 3, 4, 5, 6, 7, 8
SE6	2, 3, 4, 5, 6, 7, 8

Table 2.86

Training Query **TQ5= (5, 8, 1, 8, 3, 7)** is applied to all the search engines and the retrieved documents by each search engine are shown in Table 2.87

SE1	2	4	7	8	8	2	6	1
SE2	4	5	2	0	3	2	4	6
SE3	6	5	2	6	8	2	0	4
SE4	3	5	6	2	3	1	7	6
SE5	8	1	4	7	1	3	3	1
SE6	5	2	3	3	5	5	3	2

Table 2.87

Distribution vector (DV) for training query **TQ5= (5, 8, 1, 8, 3, 7)** for each search engine is shown in Table 2.88

	DV
SE1	1, 2, 3, 4, 5, 7, 6, 8
SE2	1, 2, 3, 5, 6, 7, 8
SE3	1, 2, 3, 4, 5, 6, 8
SE4	1, 2, 3, 4, 5, 6, 7, 8
SE5	1, 2, 3, 4, 5, 6, 7, 8
SE6	1, 2, 3, 4, 5, 6, 7, 8

Table 2.88

In next step, similarity between user query and training queries using cosine distance similarity is computed and is shown in Table 2.89

UQ	3	0	1	6	8	5	Cosine similarity
TQ1	2	1	5	2	7	3	$Sim(UQ, TQ_1)=0.1565$
TQ2	0	5	3	1	7	8	$Sim(UQ, TQ_2)=0.2572$
TQ3	3	1	0	2	0	2	$Sim(UQ, TQ_3)=0.3711$
TQ4	2	0	4	3	3	0	$Sim(UQ, TQ_4)=0.2740$
TQ5	5	8	1	8	3	7	$Sim(UQ, TQ_5)=0.2729$

Table 2.89

The top-k training queries, where k=3, selected are shown in Table 2.90

TQ	Cosine Similarity
TQ1	0.1565
TQ2	0.2572
TQ5	0.2729

Table 2.90

Now, the distribution vector for each search engine with each top-k selected training query is shown in Table 2.91

	TQ1	TQ2	TQ5
DV Of SE1	2, 3, 5, 6, 8	1, 2, 3, 4, 5, 6, 7, 8	1, 2, 3, 4, 5, 7, 6, 8
DV of SE2	1, 2, 3, 4, 5, 6, 7, 8	1, 2, 3, 4, 5, 6, 7, 8	1, 2, 3, 5, 6, 7, 8
DV of SE3	1, 2, 3, 6, 7, 8	1, 2, 3, 4, 5, 6, 7, 8	1, 2, 3, 4, 5, 6, 8
DV of SE4	2, 3, 4, 5, 6, 7, 8	1, 2, 3, 4, 5, 6, 7, 8	1, 2, 3, 4, 5, 6, 7, 8
DV of SE5	1, 2, 3, 4, 5, 6, 7, 8	1, 2, 3, 4, 5, 6, 7, 8	1, 2, 3, 4, 5, 6, 7, 8
DV of SE6	1, 2, 3, 5, 7, 8	1, 3, 4, 5, 6, 7	1, 2, 3, 4, 5, 6, 7, 8

Table 2.91

In the next step, average distribution vector (ADV) for each search engine is computed and is shown in Table 2.92

ADV1	0.5000	1.2500	2.0000	2.5000	3.2500	4.0000	4.5000	5.2500
ADV2	0.7500	1.5000	2.2500	2.7500	3.5000	4.2500	5.0000	5.7500
ADV3	0.7500	1.5000	2.2500	2.7500	3.2500	4.0000	4.5000	5.2500
ADV4	0.5000	1.2500	2.0000	2.7500	3.5000	4.2500	5.0000	5.7500
ADV5	0.7500	1.5000	2.2500	3.0000	3.7500	4.5000	5.2500	6.0000
ADV6	0.7500	1.2500	2.0000	2.5000	3.2500	3.7500	4.5000	5.0000

Table 2.92

The list of selected search engines for top-k (where k= 4, 5, 6, 7, 8) documents are shown in Table 2.93

For top k document	List of Selected Search Engines
K=4	SE1, SE4, SE2, SE3
K=5	SE1, SE4, SE2, SE3, SE5
K=6	SE1, SE4, SE2, SE3, SE5, SE6
K=7	SE1, SE4, SE2, SE3, SE5, SE6
K=8	SE1, SE4, SE2, SE3, SE5, SE6

Table 2.93

In order to compare the number of search engines in common selected by the two algorithms, the two algorithms qSim and MRDD were implemented using MATLAB – R2007a. The experimental results are discussed next.

## 2.4 Experimental Results

First, the qSim and MRDD are compared on the common search engines selected by qSim and MRDD among the top-K search engines. For this, a graph is plotted and is shown in Figure 2.3. The graph shows that the number of common search engines increases with increase in the selection of number of search engines. Further, to ascertain the percentage of search engines in common selected by qSim and MRDD among the top-K search engines, a graph is plotted and is shown in Figure 2.4. The graph shows percentage of search engines in common selected by qSim and MRDD is high for higher values of K. Also, a reasonable percentage of search engines selected by qSim and MRDD are in common.



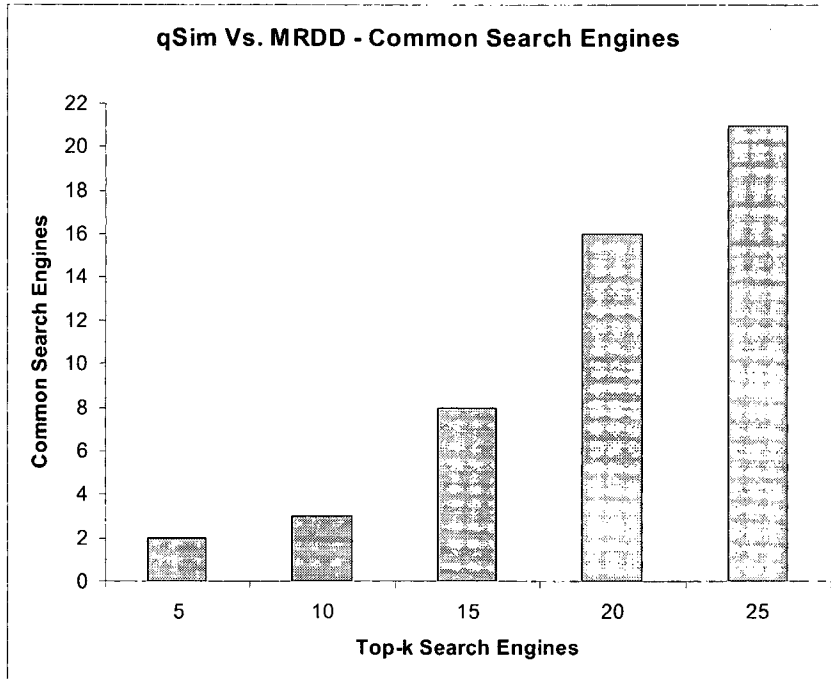


Figure 2.3 qSim Vs. MRDD – Common Search Engines

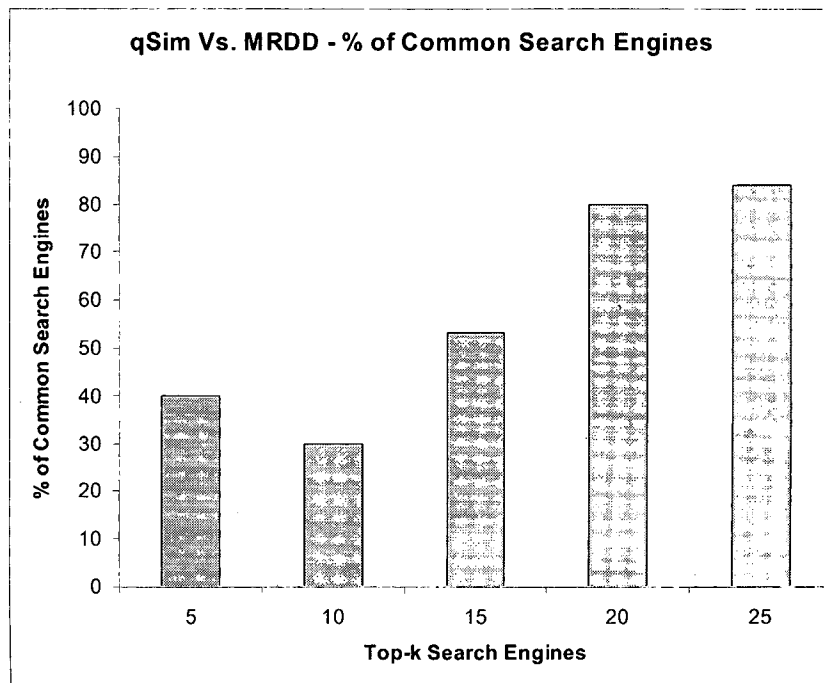


Figure 2.3 qSim Vs. MRDD – % of Common Search Engines

## CHAPTER 3

# Result Merging

Result merging is one of the key components of the metasearch engine. User poses a query to the metasearch engine through user interface. The metasearch engine selects the appropriate set of search engines based on the query. The query is rewritten and processed against the selected search engines. The results retrieved are merged and ranked into a single ranked list. Merging of the results into a single ranked list is a challenging task due to the various heterogeneous underlying search engines. Generally the relevant document returns from underlying search engine are ranked based on these document local ranking score or similarity. The result merging of documents depends on the local similarity and global similarity approaches [MYL02]. Result merging in the local similarity approach, manages the local similarity using identifying search engine. First, if the selected search engines have minimum overlap, as in the case of special purpose search engine, for a user query then all the documents retrieved will be identical from every search engine and these search engines will have their own normalized local similarity. In this case all local similarity is normalized based on common range. For example, if normalized similarity for one search engine is between 0 and 10 and other search engine is between 0 and 100 then

local similarities is renormalized on a common range, say 0 and 10, to make local similarity comparable [LMS]. Second, if the selected search engines are not identical, as in case of general purpose search engine, then all the documents retrieved would be different to each other and normalized in its own local similarity range. There are many algorithms for merging the retrieved relevant results like CORI merging [NF03], LMS Merging Technique [RAS03], Semi supervised Learning merge algorithms [SCO03a], Abstract merge algorithms [LZ+08], OWA- Based framework merging algorithm [DDR05]. In this chapter, the result merging algorithms like Abstract merging and OWA based result merging are discussed and compared.

### **3.1 OWA-Based Result Merging Approach**

The OWA Based result merging algorithm [DDR05] is based on fuzzy set theory that considers the quantifier, Order Weight Average (OWA) operator. In the OWA merging algorithm, first the position value of each document in different search engines is determined. Next, the position values of each of the documents are arranged in descending order. These is followed by calculating the weight of each search engine and then apply the OWA function to get the rank of each document and arrange them in descending order of their ranks. The algorithms based on OWA based merging [DDR05] is given in Figure 3.1. The algorithm takes as input the set of selected search engines, set of documents retrieved from all the search engines, the user query and the list of documents produced by each search engine. The algorithm is used to merge the documents retrieved from the search engines using the OWA operator and produces list of ranked documents as output.

**Input:** Let  $R[S_k, D_j, Q, L_{ij}]$ , where  $S_k$  is the set of search engines,  $D_j$  is the set of documents retrieved by search engine,  $Q$  is the user query,  $L_{ij}$  is the list of documents produced by set of search engine

**Output:** List of rank documents.

**Method:**

**Step1:** Find the set of documents  $D = (d_1, d_2, \dots, d_n)$  using

$$D = \left\{ d \mid d \in \bigcup_{j=1,2,\dots,k} L_{iq} \right\},$$

where  $L_{iq}$  is the list of documents return by search engine  $S_i$  for the given user query  $q$ .

If  $(|L_{iq}| = 0)$  then  $L_{iq}$  list is empty and  $l_{iq}(d_j) = 0 \forall d_j \in D$

**Step2:** For each search engine  $S_k$

Compute the scores  $C_{j1}, C_{j2}, \dots, C_{jk}$  using the position of the document in  $K^{\text{th}}$  ranked list  $L_{iq}$  using

$$C_{ij} = |L_{iq}| - p_{ji} + 1,$$

where  $p_{ji}$  is position of document  $d_j$  in search engine  $S_j$ .

**Step3:** For each search engine  $S_k$  arrange the score  $C_{ij}$  of each of the document in descending order

**Step4:** For each search engine  $S_k$  compute the weight using

$$w_i = Q\left(\frac{i}{K}\right) - Q\left(\frac{i-1}{K}\right), 1 \leq i \leq K$$

where  $K$  is the total no of search engines and  $Q$  is the quantifier function.

Weight of quantifier  $Q(r) = (r)^\alpha$ , where  $\alpha \geq 0$

**Step5:** Using OWA (Ordered Weighted Averaging) function, calculate the rank of each document and arrange in descending order of their list. The OWA is defined as

$$F(d) = \sum_{j=1}^K w_j \times C_{ij}$$

**Step6:** Display the list of ranked documents

Figure 3.1 Algorithm based on OWA-Based Merging [DDR05]

### 3.1.1 An Example

Let there be five search engines for user query  $q$ , and the number of documents retrieved by search engines using the first step in the algorithms, are shown in the given Table 3.1.

	1	2	3	4	5	6
SE1	D3	D4	D1	D6	D2	D5
SE2	D1	D6	D2	D4	D5	D3
SE3	D4	D2	D3	D1	D5	D6
SE4	D2	D3	D5	D4	D1	D6
SE5	D5	D2	D1	D4	D6	D3

Table 3.1

In step2, the score values of each documents is computed in the five search engines using formulae  $C_{ij} = |L_{iq}| - p_{ji} + 1$ , where  $p_{ji}$  is position of document  $d_j$  in search engine  $S_i$ ,  $L_{iq}$  is the list of documents return by search engine  $S_i$  for the given user query  $q$ . The score of the documents for each search engine is shown in Table 3. 2

	D1	D2	D3	D4	D5	D6
SE1	4	2	6	5	1	3
SE2	6	4	1	3	2	5
SE3	3	5	4	6	2	1
SE4	2	6	5	3	4	1
SE5	4	5	1	3	6	2

Table 3.2

The score values are arranging in descending order in step 3 and are shown in Table 3.3

D1	D2	D3	D4	D5	D6
6	6	6	6	6	5
4	5	5	5	4	3
4	5	4	3	2	2
3	4	1	3	2	1
2	2	1	3	1	1

Table 3.3

In step4, the weight of each search engine is computed using formula  $w_i = Q\left(\frac{i}{K}\right) - Q\left(\frac{i-1}{K}\right)$ ,  $1 \leq i \leq K$  where  $K$  is the total no of search engines and  $Q$  is the quantifier function. Weights of quantifier  $Q(r) = (r)^\alpha$ , where  $(\alpha = 0.5, 0.7, 0.9)$  are shown in Table 3.4.

	W1	W2	W3	W4	W5
$\alpha = 0.5$	0.4472	0.1852	0.1421	0.1198	0.1056
$\alpha = 0.7$	0.3241	0.2024	0.1728	0.1560	0.1446
$\alpha = 0.9$	0.2349	0.2035	0.1931	0.1866	0.1819

Table 3.4

In step5, the OWA (Ordered Weighted Averaging) function  $F(d) = \sum_{j=1}^K w_j \times C_{ij}$  is used to compute the rank of each document and are shown in Table 3.5

	D1	D2	D3	D4	D5	D6
$\alpha = 0.5$	4.5635	5.0107	4.4035	4.7121	4.0538	3.3015
$\alpha = 0.7$	4.2030	4.7343	3.9488	4.3772	3.5568	2.8742
$\alpha = 0.9$	3.9193	4.5025	3.5676	4.1117	3.1647	2.5397

Table 3.5

The merging list of documents is retrieved by all search engines in step 6 and are shown in Table 3.6

	D2	D4	D1	D3	D5	D6
$\alpha = 0.5$	5.0107	4.7121	4.5635	4.4035	4.0538	3.3015
$\alpha = 0.7$	4.7343	4.3772	4.2030	3.9488	3.5568	2.8742
$\alpha = 0.9$	4.5025	4.1117	4.1117	3.5676	3.1647	2.5397

Table 3.6

The ranked merge list does not depend on the value of  $\alpha$  as shown above in the example.

### 3.2 Abstract Merging

The abstract merging algorithm [LZ+08] is used to rank the documents return from the underlying search engines using the relevance between the abstract information of retrieved documents and the user query. In this algorithm, the relevance between the query term and the abstract is computed first and then the relevance between the query and the abstract is computed. The abstract ranked list is computed and used to rank the documents. Algorithm based on Abstract Merging [LZ+08] is shown in Figure 3.2. This algorithm takes the terms in the user query, terms in the abstract as input and produces single merged ranked list as output.

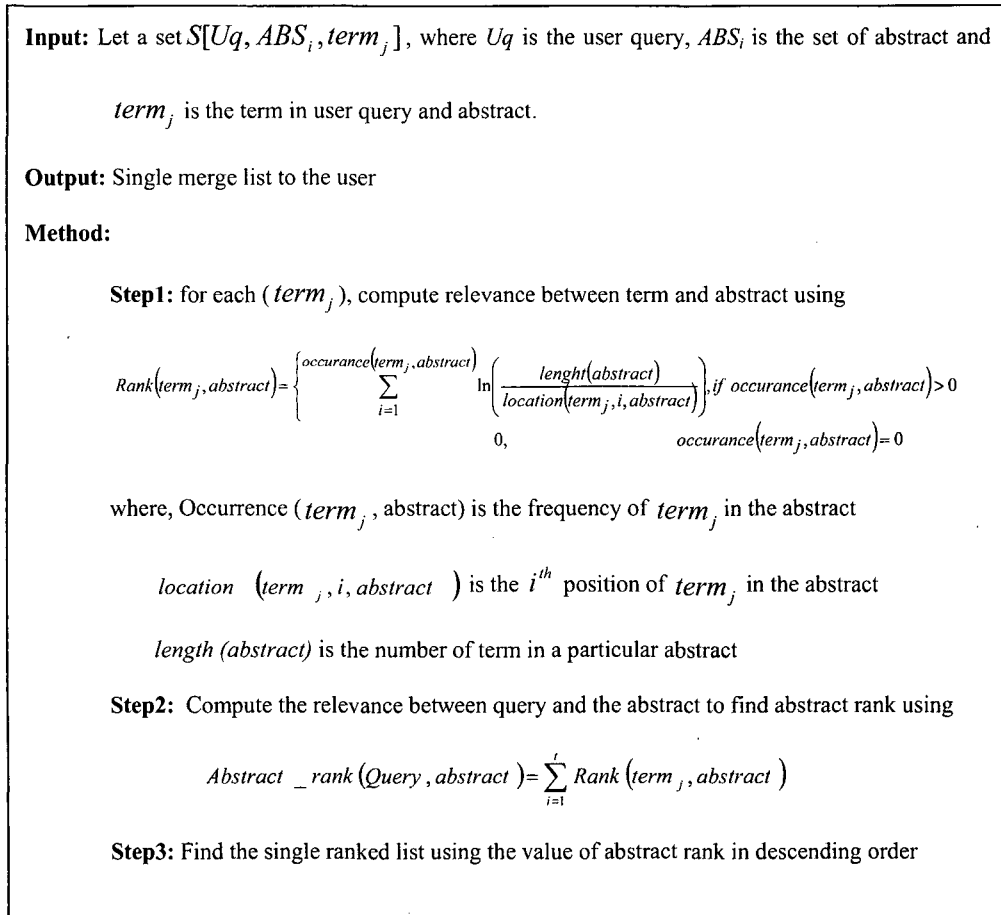


Figure 3.2 Abstract merging algorithm based on [LZ+08]

### 3.2.1 An Example

Let there be five abstracts with ten terms in each. Length of each abstracts are shown in Table 3.7.

ABS1	4	3	1	3	4	5	1	1	1	2
ABS2	3	2	4	3	2	2	2	2	4	3
ABS3	4	5	5	4	3	3	5	2	1	3
ABS4	2	5	5	4	1	5	2	1	3	1
ABS5	1	2	1	4	1	1	2	3	4	3

Table 3.7

Let the user query UQ have five terms as shown in Table 3.8

UQ	1	3	2	5	4
----	---	---	---	---	---

Table 3.8

In Step1, the relevance between each term with each abstract is computed using formula as

$$Rank(term_j, abstract) = \begin{cases} \sum_{i=1}^{occurrence(term_j, abstract)} \ln \left( \frac{length(abstract)}{location(term_j, i, abstract)} \right), & \text{if } occurrence(term_j, abstract) > 0 \\ 0, & \text{if } occurrence(term_j, abstract) = 0 \end{cases}$$

where occurrence ( $term_j$ , abstract) is the frequency of  $term_j$  in the abstract and  $location(term_j, i, abstract)$  is the  $i^{th}$  position of  $term_j$  in abstract

The  $Rank(term_j, abstract)$  of all the terms with respect to the abstract

ABS1= (4, 3, 1, 3, 4, 5, 1, 1, 1, 2) is shown in Table 3.9

Rank(term 1, ABS1)	1.8892
Rank(term 3, ABS1)	2.525
Rank(term 2, ABS1)	0.000
Rank(term 5, ABS1)	0.5108
Rank(term 4, ABS1)	2.9957

Table 3.9



The  $Rank(term_j, abstract)$  of all the terms with respect to the abstract  $ABS2=(3, 2, 4, 3, 2, 2, 2, 2, 4, 3)$  is shown in Table 3.10

Rank(term 1, ABS2)	0.00
Rank(term 3, ABS2)	3.2189
Rank(term 2, ABS2)	3.3932
Rank(term 5, ABS2)	0.00
Rank(term 4, ABS2)	1.3093

Table 3.10

The  $Rank(term_j, abstract)$  of all the terms with respect to the abstract is  $ABS3=(4, 5, 5, 4, 3, 3, 5, 2, 1, 3)$  is shown in Table 3.11

Rank(term 1, ASB3)	0.1054
Rank(term 3, ASB3)	1.2040
Rank(term 2, ABS3)	0.2231
Rank(term 5, ABS3)	3.1701
Rank(term 4, ASB3)	3.2189

Table 3.11

The  $Rank(term_j, abstract)$  of all the terms with respect to the abstract  $ABS4=(2, 5, 5, 4, 1, 5, 2, 1, 3, 1)$  is shown in Table 3.12

Rank(term 1, ABS4)	0.9163
Rank(term 3, ABS4)	0.1054
Rank(term 2, ABS4)	2.6593
Rank(term 5, ABS4)	3.3242
Rank(term 4, ABS4)	0.9163

Table 3.12

The  $Rank(term_j, abstract)$  of all the terms with respect to the abstract  $ABS5=(1, 2, 1, 4, 1, 1, 2, 3, 4, 3)$  is shown in Table 3.13

Rank( term 1, ABS5)	4.7105
Rank( term 3, ABS5)	0.2231
Rank( term 2, ABS5)	1.9661
Rank( term 5, ABS5)	0.0000
Rank( term 4, ABS5)	1.0217

Table 3.13

In step2, the relevance between the query and the abstract is computed using the formula which gives below and the values are shown in Table 3.14

$$Abstract\_rank(Query, abstract) = \sum_{i=1}^l Rank(term_i, abstract)$$

Abstract rank(query Q,ABS1)	7.9214
Abstract rank(query Q,ABS2)	7.9214
Abstract rank(query Q,ABS3)	7.9215
Abstract rank(query Q,ABS4)	7.9215
Abstract rank(query Q,ABS5)	7.9214

Table 3.14

In step3, the ranked documents are in descending order according to their relevance value between query and abstract and are shown in Table 3.15

ABS3	ABS4	ABS1	ABS2	ABS5
7.9215	7.9215	7.9214	7.9214	7.9214

Table 3.15

The ranking of documents are according to the value of similarity between abstract and the user query.

### 3.3 Example Based Comparison

Let eight search engines are selected by search engine selector. Assume each of the search engines retrieves six documents based on the user query, these are shown in Table 3.16.

SE1/DOC	6	3	4	5	1	2
SE2/DOC	4	2	1	3	6	5
SE3/DOC	4	6	5	3	1	2
SE4/DOC	2	4	3	6	5	1
SE5/DOC	5	3	1	6	2	4
SE6/DOC	6	3	2	5	1	4
SE7/DOC	1	4	3	6	2	6
SE8/DOC	3	5	2	6	4	1

Table 3.16

Let us first apply the OWA-Based Merging algorithm.

The position value of all documents is computed using formulae  $C_{ij} = |L_{iq}| - p_{ji} + 1$ , where  $p_{ji}$  is position of document  $d_j$  in search engine  $S_i$ ,  $L_{iq}$  is the list of documents return by search engine  $S_i$  for the given user query  $q$ . These are shown in Table 3.17

	D1	D2	D3	D4	D5	D6
SE1	2	1	5	4	3	6
SE2	4	5	3	6	1	2
SE3	2	1	3	6	4	5
SE4	1	6	4	5	2	3
SE5	4	2	5	1	6	3
SE6	2	4	5	1	3	6
SE7	6	2	4	5	1	3
SE8	1	4	6	2	5	3

Table 3.17

The score values of all documents in descending order are shown in Table 3.18

D1	D2	D3	D4	D5	D6
6	6	6	6	6	6
4	5	5	6	5	6
4	4	5	5	4	5
2	4	5	5	3	3
2	2	4	4	3	3
2	2	4	2	2	3
1	1	3	1	1	3
1	1	3	1	1	2

Table 3.18

The weight of each search engine is computed using formula  $w_i = Q\left(\frac{i}{K}\right) - Q\left(\frac{i-1}{K}\right)$ ,

$1 \leq i \leq K$ . Where  $K$  is the total no of search engines and  $Q$  is the quantifier function.

Weight of quantifier  $Q(r) = (r)^\alpha$ , where  $\alpha \geq 0$ . The weights are shown in Table 3.19

SE1	SE2	SE3	SE4	SE5	SE6	SE7	SE8
0.1539	0.1333	0.1265	0.1222	0.1192	0.1168	0.1149	0.1132

Table 3.19

The rank of each document is computed and are shown in Table 3.20

D1	D2	D3	D4	D5	D6
2.9070	3.2847	4.4617	3.9051	3.2817	4.0012

Table 3.20

The ranked documents according to their weight are shown in Table 3.21

D3	D6	D4	D2	D5	D1
4.4617	4.0012	3.9051	3.2847	3.2817	2.9070

Table 3.21

The top ranked documents are displayed in the above mentioned order.

**Next, let us apply the Abstract merging Algorithm**

A user query UQ with five terms is shown in Table 3.22

User_Q	3	1	6	4	2	5
--------	---	---	---	---	---	---

Table 3.22

The  $Rank(term_j, abstract)$  of all the terms with respect to the abstract **ABS1**= (6, 3, 4, 5, 1, 2) is shown in Table 3.23

Rank(term 3, ABS1)	1.0986
Rank(term 1, ABS1)	0.1823
Rank(term 6, ABS1)	1.7918
Rank(term 4, ABS1)	0.6931
Rank(term 2, ABS1)	0.0000
Rank(term 5, ABS1)	0.4055

Table 3.23

The  $Rank(term_j, abstract)$  of all the terms with respect to the abstract **ABS2**= (4, 2, 1, 3, 6, 5) is shown in Table 3.24

Rank(term 3, ABS2)	0.4055
Rank(term 1, ABS2)	0.6931
Rank(term 6, ABS2)	0.1823
Rank(term 4, ABS2)	1.7918
Rank(term 2, ABS2)	1.0986
Rank(term 5, ABS2)	0.0000

Table 3.24

The  $\text{Rank}(\text{term}_j, \text{abstract})$  of all the terms with respect to the abstract **ABS3**= (4, 6, 5, 3, 1, 2) is shown in Table 3.25

Rank(term 3, ABS3)	0.4055
Rank(term 1, ABS3)	0.1823
Rank(term 6, ABS3)	1.0986
Rank(term 4, ABS3)	1.7918
Rank(term 2, ABS3)	0.0000
Rank(term 5, ABS3)	0.6931

Table 3.25

The  $\text{Rank}(\text{term}_j, \text{abstract})$  of all the terms with respect to the abstract **ABS4**= (2, 4, 3, 6, 5, 1) is shown in Table 3.26

Rank(term 3, ABS4)	0.6931
Rank(term 1, ABS4)	0.0000
Rank(term 6, ABS4)	0.4055
Rank(term 4, ABS4)	1.0986
Rank(term 2, ABS4)	1.7918
Rank(term 5, ABS4)	0.1823

Table 3.26

The  $\text{Rank}(\text{term}_j, \text{abstract})$  of all the terms with respect to the abstract **ABS5**= (5, 3, 1, 6, 2, 4) is shown in Table 3.27

Rank(term 3, ABS5)	1.0986
Rank(term 1, ABS5)	0.6931
Rank(term 6, ABS5)	0.4055
Rank(term 4, ABS5)	0.0000
Rank(term 2, ABS5)	0.1823
Rank(term 5, ABS5)	1.7918

Table 3.27

The  $\text{Rank}(\text{term}_j, \text{abstract})$  of all the term with respect to the abstract **ABS6**= (6, 3, 2, 5, 1, 4) is shown in Table 3.28

Rank(term 3, ABS6)	1.0986
Rank(term 1, ABS6)	0.1823
Rank(term 6, ABS6)	1.7918
Rank(term 4, ABS6)	0.0000
Rank(term 2, ABS6)	0.6931
Rank(term 5, ABS6)	0.4055

Table 3.28

The  $Rank(term, abstract)$  of all the term with respect to the abstract **ABS7**= (1, 4, 3, 6, 2, 5) is shown in Table 3.29

Rank(term 3, ABS7)	0.6931
Rank(term 1, ABS7)	1.7918
Rank(term 6, ABS7)	0.4055
Rank(term 4, ABS7)	1.0986
Rank(term 2, ABS7)	0.1823
Rank(term 5, ABS7)	0.0000

Table 3.29

The  $Rank(term, abstract)$  of all the terms with respect to the abstract **ABS8**= (3, 5, 2, 6, 4, 1) is shown in Table 3.30

Rank(term 3, ABS8)	1.7918
Rank(term 1, ABS8)	0.0000
Rank(term 6, ABS8)	0.4055
Rank(term 4, ABS8)	0.1823
Rank(term 2, ABS8)	0.6931
Rank(term 5, ABS8)	1.0986

Table 3.30

In step 2, the relevance between query and abstract is computed and is shown in Table 3.31

Abstract rank(query Q, ABS1)	4.1713
Abstract rank(query Q, ABS2)	4.1713
Abstract rank(query Q, ABS3)	4.1713
Abstract rank(query Q, ABS4)	4.1713
Abstract rank(query Q, ABS5)	4.1713

Table 3.31

Ranked list of documents are shown in Table 3.32

ASB1	ABS2	ABS3	ABS4	ABS5
4.1713	4.1713	4.1713	4.1713	4.1713

Table 3.32

The top-K documents selected by algorithm OWA and AM for the same data set given in the example of section 3.3 are shown in Table 3.33

Top-K Documents	OWA	AM
Top-1 document	D3	D1
Top-2 documents	D3, D6	D1, D2
Top-3 documents	D3, D6, D4	D1, D2, D3

Table 3.33

The Top-3 documents selected by merging algorithms OWA and AM are different and this may be due to the heuristic used by the two algorithms.

In order to compare the quality of the documents selected by the two algorithms, the two algorithms OWA and AM were implemented using MATLAB – R2007a. The experimental results are discussed next.

### 3.4 Experimental Results

First, the OWA and AM are compared on the average document score achieved by top-K documents. For this, a graph is plotted and is shown in Fig. 3.3. The graph shows that the average document score of documents retrieved by OWA is consistently higher than those retrieved by AM. This shows that the algorithm OWA is able to retrieve better documents.

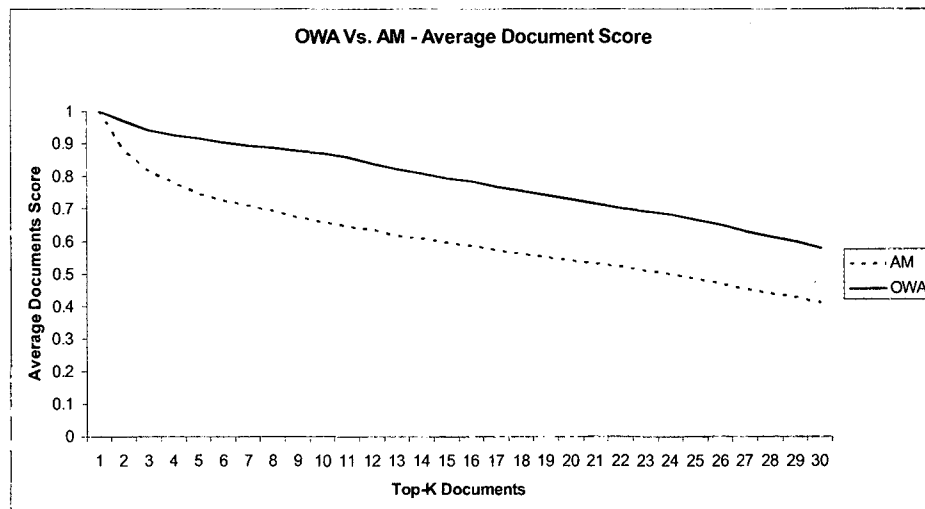


Figure 3.3 OWA Vs. AM – Average Document Score

Further, to ascertain the rise in the document score with every document retrieved by algorithm OWA and AM, a graph showing cumulative document score against top-K documents is plotted. This graph is shown in Figure 3.4. The graph shows that the rise in the document score with every document retrieved is greater for OWA in comparison to AM. This further implies that OWA retrieves better quality documents.

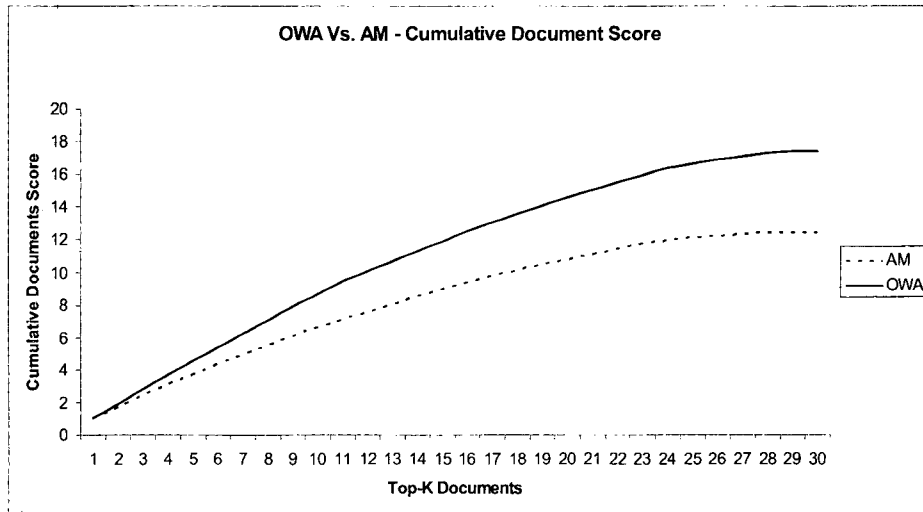


Figure 3.3 OWA Vs. AM – Cumulative Document Score



## **CHAPTER 4**

### **Conclusion**

The information on the Web is mostly semi-structure or unstructured in nature. The search engines are available to provide access to such data. These search engines are not scalable as they have a limited set of data sources associated with them. This problem has been addressed by the metasearch engine, which acts as search engine over many search engines thereby is able to access large number of databases. As a result, the metasearch engine addresses the scalability problem. There are two major components of a metasearch engine namely search engine selector and result merger. The search engine selector is responsible for selecting a set of search engines that are relevant to the user query. The user query is posed against these selected search engines and results are retrieved from them. The result merger then merges the retrieved results into a single ranked list. The documents as per their ranking in the rank list are displayed to the user.

Several search engine selection approaches exist in the literature and in this dissertation the approach query Similarity (qSim) and Modeling Relevant Document Distribution (MRDD) has been discussed and compared. Further, qSim and MRDD has been implemented and compared on the number of search engines in common selected by them.

It is observed that the two algorithms tend to select a reasonably high number of search engines in common when a reasonable high number of search engines are selected by them.

Several result merging approaches exist in literature and in this dissertation the approach Order Weight Average (OWA) and Abstract Merging (AM) has been discussed and compared. Further, OWA and AM has been implemented and compared on the average document score and cumulative document score of the documents retrieved by them. The results show that the documents selected by OWA have a higher average document score than those selected using AM. Furthermore, with every document selection, the increase in documents score is greater for documents selected using OWA in comparison to those selected using AM. This shows that in comparison to AM, OWA is able retrieve better top-k documents for a given user query.

## References

- [AMS00] A. Smeulders, M.Worring, S. Santini, A. Gupta, and R. Jain. Content-based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, (December 2000)
- [BM] Bergman, M. The deep Web: Surfacing the hidden value. Bright Planet, [www.completeplanet.com/Tutorials/DeepWeb/index.asp](http://www.completeplanet.com/Tutorials/DeepWeb/index.asp).
- [BR04] Baeza-Yates, R. and Ribeiro-Neto, B. Modern Information Retrieval, Pearson Education
- [CLC95] Callan, J. Lu, Z. and Croft, W.1995b. Searching distributed collection with inference networks. In Proceeding of the ACM SIGIR Conference (Seattle, WA), Page no. 21-28, (July1995)
- [DDR05] D.D.Elizabeth, D.Arijit and R.Vijay. 2005. A Comprehensive OWA-Based Framework for Result Merging in Metasearch. In Springer –Verlag Berlin Heidelberg. pages 193-201, (2005)
- [DH97] Drelinger, D. and Howe, A. 1997. Experience with Selecting Search engine Using Metasearch.ACM Trans. Inform. Syst. Page no. 195-222. (July 1997)

- [DH97] Drieling, D. and Howe, A. Experience with Selecting search engine using metasearch, (1997)
- [EO97] E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. IEEE Expert (January-February): Page no.11-14, (1997)
- [F99] Fuhr, N. A Decision –Theoretic Approach to Database Selection in Networked IR, ACM Transactions on Information Systems (TOIS), Volume 17, Issue 3, pages: 229 – 249, (July 1999)
- [FG99] Fan, Y. and Gauch, S. Adaptive agent for information gathering from multiple, distributed information source. In Proceeding of the AAAI Symposium on Intelligent Agent in Cyberspace, Page no.40-46, (1999)
- [GGM95] Gravano, L., and Garcia-Molina, H.1995. Generalizing gloss to vector space databases and broker hierarchies' .In proceeding of the International Conferences on Very Large Data Bases, Page no.196-205. (August 1997)
- [HT99] Hawking, D. and Thistlewaite, P. Methods for Information Server Selection, ACM Transaction on Information Systems, Vol.17, pages 40-76, (Jan 1999)
- [KM94] Koster, M. Aliweb: Archie-like indexing in the Web. Computer Network and ISDN Syst. 27, 2, Page no.175–182. (1994)
- [KP+05] Keyhamipoor, A.H., Piroozmand, M., Moshiri, B., Lucas, C. A Multi-Layer/Mutli-Agent Architecture for Metasearch Engines, In AIML 05 Conference CICC, Cairo, Egypt. (Dec-2005)
- [L98] Lui, Bing , Web Data Mining. ACM Computing Classification (1998)
- [LBE] Metasearch Engine <http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/MetaSearch.htm>
- [LL99] Lawrence.S. and Lee Giles, C. Accessibility of information on the wb. Nature 400, Page no.107–109. (1999)

- [LMS] L.Yiyao, M. Weiyi, S. Liangcai, Y. Clement, L. King-Lup. Evaluation of result merging Strategies for metasearch engine
- [LZ+08] L.Chunshuang, Z.Zhiqiang, X.Xiaoqin, L.TingTing Evaluation of Meta-Search Eninge Merge Algorithms. In IEEE, pages 9-14, (2008)
- [MW] Meng, W., Metasearch engine
- [MYL02] Meng, W., Yu. C and Liu K. Building effective and efficient metasearch engine ACM Computing Surveys, Vol.34, No.1, (March 2002.)
- [NF03] Nottelmann, H. and Fuhr, N. Combining CORI and decision theoretic approach for advanced resource selection, (2003)
- [RAS03] Rasolofo, Y., Abbaci, F. and Savoy, J. Approaches to Collection Selection and Results Merging for Distributed Information Retrieval, (2003)
- [RD+86] Rumelhart, D.E, Hinton, G.E., and Williams, R.J. Learning internet presentations by error propagation. In D.E. Rumelhart & J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundation. Cambridge. MA: MIT Press, (1986)
- [S89] Salton, G. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison Wesley, Reading, MA. (1989)
- [SB88] Salton, G. and Buckley, C. Term-Weighting Approaches in Automatic Retrieval. Information Processing and Management, 24(5), Page no. 513–525, (1988)
- [SC03a] Si, L. and Callan, J. A Semi supervised Learning Method to Merge Search engine Results. ACM Transactions on information System, Vol.21, No.4, pages 457-491, (Oct 2003)
- [SC03b] Si, L., Callan, J., Relevant Document Distribution Estimation method for resource selection. In Proceeding of the twenty fifth Annual International ACM SIGIR Conference on research and Development in information Retrieval, (2003).

- [SJ04] Souza, D.Zobel, J.Thom, A, J.Is CORI effective for Collection Selection? An Exploration of Parameter, Queries, and Data, In Proceeding of the 9<sup>th</sup> Australasian Document Computing Symposium, (Dec 2004).
- [SK06] Silberschatz, A., Korth, H.F and Sudershan S.Database Management System Concept, Fifth Edition, McGraw- Hill International Edition.
- [SLH09] Suleyman, C., Luo, S. and Hao, Y.2009. Learning from Past Queries for Resource Selection. In CIKM, pages 1867-1870, (Nov 2009)
- [VGL95a] Voorhees, E.M., Gupta, N.K. and Laird, B.J. The Collection Fusion Problem. In Proceeding of the Third Text Retrieval Conference, (March 1995).
- [VGL95b] Voorhees, E., Gupta,N.K, and Laird, B.L. 1995b. Learning Collection fusion Strategies in Proceeding of the ACM SIGIR Conference (Seattle, WA), Page no,172-179, (July1995)
- [VT97] Voorhees, E. And Tong, R. Multiplesearch engines in database merging. In Proceedings of the Second ACM International Conference on Digital Libraries (Philadelphia, PA), Page no.93–102. (July 1997)
- [YL96] Yuwono, B. and Lee, D. Search and ranking algorithm for locating resource on the World Wide Web. In Proceeding of the 5th International Conference on Data Engineering, Page no.164-177. (Feb. 1996)