1-1-2008

QoS Based Ranking For Web Search

Xiangyi Chen
Ryerson University

Follow this and additional works at: <http://digitalcommons.ryerson.ca/dissertations>

Part of the [Computer Engineering Commons](#)

Recommended Citation

Chen, Xiangyi, "QoS Based Ranking For Web Search" (2008). *Theses and dissertations*. Paper 1081.

This Thesis is brought to you for free and open access by Digital Commons @ Ryerson. It has been accepted for inclusion in Theses and dissertations by an authorized administrator of Digital Commons @ Ryerson. For more information, please contact bcameron@ryerson.ca.

TK
S105.884
C525
2008

QoS BASED RANKING FOR WEB SEARCH

By

Xiangyi Chen

B.Eng. in Computer Science and Technology, Fuzhou University, China, July 2000

A thesis

Presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2008

© Xiangyi Chen 2008

AUTHOR'S DECLARATION

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

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

QoS Based Ranking for Web Search

Master of Applied Science, 2008

Xiangyi Chen

Electrical and Computer Engineering

Ryerson University

ABSTRACT

Text, link and usage information are the most commonly used sources in the ranking algorithm of a web search engine. In this thesis, we argue that the quality of the web pages such as the performance of the page delivery (e.g, reliability and response time) should also play an important role in ranking, especially for users with a slow Internet connection or mobile users. Based on this principle, if two pages have the same level of relevancy to a query, the one with a higher delivery quality (e.g. faster response) should be ranked higher. We define several important attributes for the Quality of Service (QoS) and explain how we rank the web pages based on these attributes. In addition, while combining those QoS attributes, we have tested and compared different aggregation algorithms. The experiment results show that our proposed algorithms can promote the pages with a higher delivery quality to higher positions in the result list, which is beneficial to users to improve their overall experiences of using the search engine and QoS based re-ranking algorithm always gets the best performance.

ACKNOWLEDGEMENTS

Although this thesis represents my individual work, there are various people who, during the past two years, provided me with useful and helpful assistance and support. Without their care and support, this thesis would not have been finished.

Firstly, I would like to express my deepest gratitude to my supervisor Dr. Cherie Ding for her guidance and constant support in helping me to conduct and complete this work. She discussed with me, kindly helped me solve problems I encountered when doing research work, and always gave me encouragement, which inspired me to keep stepping forward. To Cherie, I express my heartfelt thanks.

I want to thank my co-supervisor Dr. Dimitri Androutsos for his kind help and encouragement. He always shows his caring nature whenever we meet. He also invested time to review my thesis carefully and provide valuable comments. Without his support, the study could not be done.

Many thanks to all the people I have come to know in computer engineering department, whose friendship and companionship I will always enjoy and get refreshed. I owe my sincere appreciation to my wife Chaoran Chen, who gave me space and consistently encouraged me while I was studying. These acknowledgements would not be complete without recognizing my parents. I could not do this without their unfailing love, affection and invaluable support in my life and studies. To each of them, my thanks and my love.

TABLE OF CONTENTS

QoS BASED RANKING FOR WEB SEARCH	i
AUTHOR'S DECLARATION	ii
ABSTRACT.....	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS.....	v
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTER 1	1
INTRODUCTION	1
1.1 Background.....	3
1.1.1 Simple Search Engine Retrieval Process	3
1.1.2 Meta Search Engine Retrieval Process	5
1.2 Motivation.....	9
1.3 Research Goals and Approaches.....	12
1.4 Thesis Outline	13
CHAPTER 2	14
LITERATURE REVIEW	14
2.1 Rank Aggregation Algorithms.....	15
2.1.1 Supervised Rank Aggregation Method.....	15
2.1.2 Unsupervised Rank Aggregation Method.....	16
2.1.2.1 Positional Rank Aggregation Method.....	16
2.1.2.2 Majoritarian Rank Aggregation Method.....	17
2.1.2.3 Other Rank Aggregation Methods	19
2.2 Score Combination Algorithm.....	20
2.3 Applying QoS in IR System	21
CHAPTER 3	24
QoS ATTRIBUTES RELATED TO RANKING	24
3.1 Overview	24
3.2 QoS Attribute Definition.....	27
CHAPTER 4	30
COMBINING QoS ATTRIBUTES IN RANKING	30
4.1 System Framework Overview.....	31
4.2 QoS Collection and Evaluation Model	33
4.3 QoS Based Combination and Re-Ranking Model	35
4.3.1 QoS Based Re-Ranking	35
4.3.2 Linear Score Combination with the Original Rank	36
4.3.3 Weighted Condorcet Rank Aggregation with QoS Attributes.....	37
4.3.4 Median-Rank Aggregation with QoS Attributes	40
4.3.5 Weighted Borda Rank Aggregation with QoS Attributes.....	41
4.3.6 Genetic Algorithm Based Weighted Partial Footrule Optimal Rank Aggregation with QoS Attributes	42
CHAPTER 5	50

ACKNOWLEDGEMENTS

Although this thesis represents my individual work, there are various people who, during the past two years, provided me with useful and helpful assistance and support. Without their care and support, this thesis would not have been finished.

Firstly, I would like to express my deepest gratitude to my supervisor Dr. Cherie Ding for her guidance and constant support in helping me to conduct and complete this work. She discussed with me, kindly helped me solve problems I encountered when doing research work, and always gave me encouragement, which inspired me to keep stepping forward. To Cherie, I express my heartfelt thanks.

I want to thank my co-supervisor Dr. Dimitri Androutsos for his kind help and encouragement. He always shows his caring nature whenever we meet. He also invested time to review my thesis carefully and provide valuable comments. Without his support, the study could not be done.

Many thanks to all the people I have come to know in computer engineering department, whose friendship and companionship I will always enjoy and get refreshed. I owe my sincere appreciation to my wife Chaoran Chen, who gave me space and consistently encouraged me while I was studying. These acknowledgements would not be complete without recognizing my parents. I could not do this without their unfailing love, affection and invaluable support in my life and studies. To each of them, my thanks and my love.

TABLE OF CONTENTS

QoS BASED RANKING FOR WEB SEARCH	i
AUTHOR'S DECLARATION	ii
ABSTRACT.....	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS.....	v
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTER 1	1
INTRODUCTION	1
1.1 Background.....	3
1.1.1 Simple Search Engine Retrieval Process	3
1.1.2 Meta Search Engine Retrieval Process	5
1.2 Motivation.....	9
1.3 Research Goals and Approaches.....	12
1.4 Thesis Outline	13
CHAPTER 2	14
LITERATURE REVIEW	14
2.1 Rank Aggregation Algorithms.....	15
2.1.1 Supervised Rank Aggregation Method.....	15
2.1.2 Unsupervised Rank Aggregation Method.....	16
2.1.2.1 Positional Rank Aggregation Method.....	16
2.1.2.2 Majoritarian Rank Aggregation Method.....	17
2.1.2.3 Other Rank Aggregation Methods	19
2.2 Score Combination Algorithm.....	20
2.3 Applying QoS in IR System	21
CHAPTER 3	24
QoS ATTRIBUTES RELATED TO RANKING	24
3.1 Overview	24
3.2 QoS Attribute Definition.....	27
CHAPTER 4	30
COMBINING QoS ATTRIBUTES IN RANKING	30
4.1 System Framework Overview.....	31
4.2 QoS Collection and Evaluation Model	33
4.3 QoS Based Combination and Re-Ranking Model	35
4.3.1 QoS Based Re-Ranking	35
4.3.2 Linear Score Combination with the Original Rank	36
4.3.3 Weighted Condorcet Rank Aggregation with QoS Attributes.....	37
4.3.4 Median-Rank Aggregation with QoS Attributes	40
4.3.5 Weighted Borda Rank Aggregation with QoS Attributes.....	41
4.3.6 Genetic Algorithm Based Weighted Partial Footrule Optimal Rank Aggregation with QoS Attributes	42
CHAPTER 5	50

EXPERIMENT DESIGN AND RESULT ANALYSES.....	50
5.1 Data Set.....	50
5.1.1 Query.....	51
5.1.2 Databases	52
5.2 Experiment Design and Implementation	53
5.3 Evaluation Method.....	57
5.4 Combination Methods and Use Cases	60
5.5 Results and Analyses	61
5.5 Discussion	70
CHAPTER 6	72
CONCLUSIONS.....	72
REFERENCES	75

LIST OF TABLES

Table 5- 1. Query list submitted to retrieval..... 52

Table 5- 2. Database table used in the experiment 53

LIST OF FIGURES

Figure 1. A simple search engine retrieval process	3
Figure 2. The process of meta search engine	6
Figure 3. The process of meta search engine based on QoS.....	32
Figure 4. SimpleRun algorithm in weighted Condorcet rank aggregation	38
Figure 5. Weighted Condorcet rank aggregation using Quick Sort ranking algorithm	39
Figure 6. Partition algorithm for QuickSort.....	39
Figure 7. Weighted Borda rank aggregation algorithm with QoS attributes.....	41
Figure 8. An example for crossover in Genetic Algorithm	46
Figure 9. Simple procedure of genetic algorithm	49
Figure 10. The process of QoS Watcher which collects the data of QoS attributes	54
Figure 11. The process of QoS reranking with QoS attributes	56
Figure 12. ΔQoS for each QoS attribute when only the weight of that attribute is 1	62
Figure 13. Average ΔQoS for each QoS attribute when the weight of this attribute is 1	63
Figure 14. Comparisons of the 6 combination methods on average ΔQoS	64
Figure 15. Comparison of 6 combination methods on average ΔQoS considering 3 QoS attributes only	65
Figure 16. Comparison of T1-T6 combination methods on average ΔQoS	66
Figure 17. Comparison of T1-T6 combination methods on average delta-Top10	67
Figure 18. Comparison of T1-T6 combination methods on effectiveness.....	68
Figure 19. Comparison of T1-T6 combination methods on effectiveness considering 3 QoS attributes only.....	69

CHAPTER 1

INTRODUCTION

In 1945, an American scientist and engineer Dr. Vannevar Bush had first proposed a new idea in his article "As We May Think" [1]. He described how to use a computer to store data and to search relevant information. Nowadays, Internet and the World Wide Web (WWW) have been exploited tremendously as the place to store various kinds of information. In this Internet era, how to search from this huge data storage is a very crucial task for the better use of the Internet. Because of this, information retrieval techniques have been growing explosively, especially the web search engine has become the most popular and foundational way to find useful information on the web. It is hard to imagine that people can search information without the help of search engines because the quantity of information stored on the Internet has been increasing important. Web search engines, such as Google [2], Yahoo [3] and MSN Live Search [4], which serve as gateways to the information repository, have played a crucial role in people's daily life [5]. There are millions of accesses to search engines everyday and the number keeps increasing. Some statistics show that core search engines such as Google, Yahoo, Microsoft's Live Search, AOL, and Ask.com have collectively increased by 15% in December 2007, compared to 2006, serving 9.6 billion searches [6].

People use the search engine to find web pages of their interest. Usually they input some queries, which are composed of a couple of keywords related to their topics of

search. After reviewing the output, people might click on the hyperlinks in the top positions of the output list to browse the most relevant result pages, or they might modify the query in order to get a supposedly improved result according to the relevance of those pages. Although search engines are frequently used, users still suffer from their poor retrieval performance sometimes. When the query is submitted, the user might find some irrelevant URL links lie in the top 10 results or many relevant pages are successfully retrieved but far away from the top 10 positions, which most of the users won't go beyond. Some top 10 documents, the supposedly most relevant results, might not be stably accessible or even not accessible at all. Moreover, the response time of these top 10 pages might be unbearable, especially for some mobile users or customers who experienced slow network connections. Based on all these observations, it is the purpose of this study to devise a solution to make the top 10 retrieved documents have a higher quality of service (QoS).

In this thesis, a QoS based ranking algorithm for web search is proposed and several QoS factors such as reliability, latency, freshness, file size and media richness are defined with an explanation about how to measure and calculate them. Further more, this QoS based ranking algorithm is aggregated with several other ranking algorithms to try achieving a better performance than common ranking algorithms.

1.1 Background

1.1.1 Simple Search Engine Retrieval Process

In order to describe the basic search engine retrieval process, we create a simple architecture shown in Figure 1.

Figure 1. A simple search engine retrieval process

The whole search consists of two parts: the web crawler part and the search retrieval part. In the web crawler part (right of Figure 1), a robot program called “crawler” automatically downloads the web pages from some selected “seed” pages into the local repository. The typical “seed” for the web crawler is an automatically

augmented URL list. After downloading the web pages in the “seed”, it adds all the hyperlinks in the downloaded pages as the new “seed” and continues to collect more web pages. The whole process is iterative. The downloaded web pages will be processed by some text operations such as stop word removing, stemming and index term selection. After the text operations, original web pages are converted to the “*logical view of them*” [7], which are then saved in the local repository. In the mean time, an indexing program builds data structures over the text content of those local web pages so as to speed up the search. This process is worthwhile if the database is large. Those indexes can be updated in a regular basis. Different index structures are used depending on different models, and the most popular one is the inverted file [7] [8].

After index structures are built, the search retrieval part (left of Figure 1) is ready to serve users’ queries. Users can get some help information from the visual interface; input their queries and other requirements such as user preference information, special requirement, etc. For example, Google [2] has an “Advanced Search” option in which users can specify the structures of the input keywords to match their queries, or filter out some web pages, or use other valuable tools that can improve the efficiency and reduce the search time. According to the user’s query and preference options, query operations are executed and transformed to the system representation so that we could easily compare them with those pages which have been converted to the same format of representation.

Finally, the search engine system compares the similarity between the query representation and text representations. By using the previously generated indexes, the comparison process is further accelerated. Before returning the retrieved documents to

the user, a ranking algorithm is applied to the similarity values of the query. Different search engines have their own ranking strategies, but the similarity between the query and the document is always a key factor. Then, the user examines the retrieval result in the order of relevancy and will optionally give the feedback to the search engine system. In the feedback operation, a query expansion is automatically done to help the users getting better keywords which represent their needs more precisely.

1.1.2 Meta Search Engine Retrieval Process

Different search engines have different search strategies, thereby allow one to get different retrieval results as well as different performances. A meta search engines are a special type of search engines, which combines the results from several search engines into a single list for the user. In order to describe the basic meta search engine retrieval process, a simple architecture is shown in Figure 2. In the figure, we can see it is very similar to Figure 1, however, instead of ranking the pages from local repository, meta search engine uses the rank combination method to combine the multiple retrieved results from other search engines and returns the combined result by applying special ranking strategies.

Figure 2. The process of meta search engine

Different search engines have different formats for the input query; however most of the users are not well trained to generate the queries with high quality which can represent what they want. Therefore, the qualities of retrieval results are not satisfactory in many cases. For example, in the search engine *A*, a blank space in the user input query means an “or” relationship among the words, but in another search engine *B*, it is treated as a character in a whole keyword. Definitely, two different search results will be returned.

A meta search engine can help users to automatically generate different query formats suitable for different search engines so as to improve the retrieval performance. In addition, a meta search engine can add specific algorithms to rank those results from different search engines while compiling them into a single list. For example, a meta search engine can remove duplicate URLs from the retrieval lists, or can apply a particular ranking strategy to promote the most relevant documents to the top positions of

the retrieval list. Also different search engines crawl different parts of the web and use different indexing and ranking algorithms. If we could combine them together, ideally, it will cover a wider range of pages and take advantage of different ranking algorithms. Therefore, meta search engine is a promising area to work on in order to get a better search engine. This thesis implements the QoS based ranking into a meta search engine.

1.1.3 Popular Ranking Algorithms for Web Search

Ranking functions are the core parts of information retrieval systems. Different search engines exploit different ranking algorithms to meet their specific targets. The classical ranking strategies are those content-based ranking algorithms such as the vector space model [9], the probabilistic model [10], etc. For example, the most classical algorithm is the TFIDF-based vector space model in which measurements such as *term frequency (tf)* and *inverted document frequency (idf)* are used. Many other ranking algorithms exploit different parts of the content of the pages, but they all compare the similarity values between the query and the document vectors in the collection to find the most relevant documents. The structured text ranking strategy [11] [12] is another example of content-based ranking algorithm that analyses the hypertext structural information of the candidate page, such as title, header and body. If the token from the query appears in the title of a page, that page is likely to be more relevant than the page in which the term appears in the body. Based on positions in which the term appears, different weights are assigned to calculate the term frequency before they are used in the latter ranking process. For example, if the term appears in the title of the web page, the weight will be higher so that it will have a higher term frequency. After calculating the

term frequency in all positions where the term appears, we can get the overall term frequency based on the structure information and then use this data to measure the relevancy of the candidate pages.

Link-based ranking algorithms such as PageRank [5] and HITS [13] are other types of widely used ranking strategies. PageRank is the key ranking algorithm currently used in the popular search engine - Google. It exploits the hyperlink structure of web documents to rank those web pages. The fundamental idea is that, if a web page is linked by many other web pages, the linked web page is more important, thereby a higher rank score is assigned. The linked web page is called authoritative page. However, Google considers not only the value of PageRank, but also the content of the web page. If the page matches the query better, it means that the page is more important in the content, thereby the value of the PageRank from this page will get higher weight during evaluation. Furthermore, PageRank is valuable in past events where many pages were created and well linked, whereas it might not work very well for web pages related to new and real time events due to their high updating frequencies. PageRank can't perfectly measure a page with many in-coming links as well as many out-going links. The HITS algorithm successfully solves this problem. It considers not only the score of the authoritative page, but also the score of the "hub" pages. A "hub" page is a page which points to many authoritative pages. Another difference between PageRank and HITS is that PageRank gives the score to every page in the collection, whereas HITS gives scores based on query.

Instead of analyzing the content of the Web pages or exploiting the link information, click-through ranking strategy is another type of ranking algorithms which

give the score to Web pages by analyzing the user behaviors. DirectHit [14] is a representative example. It ranks the web pages by analyzing the click rates of web pages from the server log files for a specific query. When a user submits a query, the algorithm calculates the page scores based on the previous click-rate of those pages for this query. For example, if a page gets more hits by previous users for the same query, it will get higher rank score because previous users' behavior implies its popularity. However, since the user might browse the web page randomly, or even when the user has a clear idea about what to look for initially, he/she might change his/her mind after a while. Due to these reasons, the result might not be accurate for all the cases. In addition, server log analysis algorithm needs time to collect useful user popularity data, which means that it might not be helpful to track down some news web sites or less popular web sites. [14]

1.2 Motivation

When we look at content-based algorithms, link-based algorithms and click-through algorithms, all of them measure the quality of the web pages to improve the retrieval result. For example, if a web page has a high PageRank score, it means that it is referred by many other documents. In other words, it means that this highly referred web page has a high authority. Similarly, if a web page was clicked frequently, it means it has high popularity. Therefore, this web page should be ranked higher. Either authority or popularity is a kind of measurement on the quality of the web pages.

There are many perspectives when we talk about the Quality of the web pages, such as the quality of its content (for example, authority or hub scores), the quality of its delivery (for example, reliability or latency), or the quality of its reputation (for example,

popularity), etc. Currently many popular search engines are using the quality of contents of web pages as a factor in ranking; however, few of them use the other types of qualities in retrieval. If a user wants to read some news, he might prefer to get the newly updated pages other than pages posted 1 month ago. In this case, we can define the quality of the web page as the freshness of the page. If the value of the **last_modif** field in its HTTP response [4] is very small, it means that the page is quite old, thus poor in its quality. Another example of quality could be the reliability of the Web page. If a Web page keeps returning a “404” error which means this page cannot be found by the browser [15], the page is not stable on its hosting service, thereby having poor *reliability*. In addition, on a larger scale, if the Web pages from a Web site always return “HTTP 404” client errors, we can further believe that this Web site has a low stability, thus poor quality of *reliability* also. When a user submits a query, he/she is willing to get a result with the actual content displayed rather than error messages. Therefore, if two pages have similar content and relevancy, the page with higher reliability should get a higher rank because this attribute can help the user to reach his/her desired result more quickly. Based on these observations, we believe that different quality attributes of web pages will help to improve the retrieval performance and fulfill the user’s satisfaction.

All those quality attributes shown above are part of Quality of Service (QoS). QoS is usually a term used in the telecommunication area to measure the performance of a network and recently, it has been used for web service measurement. In web service, QoS is defined as a set of non-functional attributes such as scalability, performance, availability, cost, etc. [16]. As we mentioned before, relevancy measurements based on a documents’ content can be considered as a type of quality attribute. If a retrieved

document has a higher relevancy, this page will be more desirable by the user, therefore it has higher quality. Moreover, if a page performs well in several qualities, such as a high degree of relevancy to user's queries, a very low error rate in all environments and a fast delivery rate, this page is considered a "high quality page". Therefore, we believe that it is very beneficial to consider multiple QoS attributes when ranking the search results, especially when the users have limitations on network access, such as mobile users.

Combination algorithm research is another hot topic in information retrieval area [17-30]. In the normal search engines, combination algorithms can be used to generate a single result list from the scores of multi-dimensional document representations. In meta search engines, these techniques are also widely employed to aggregate multiple rankings from the underlying search engines and then generate a single list for the users. Depending on the different types of data (i.e. rank or score), the combination method will be different. For example, for pages with relevance scores available, score-based combination method such as linear combination [31] [32] [22] [33] or linear regression [20] might be better than rank-based aggregation method such as Borda-count [17] [18] or Condorcet-fusion [19]. But if the score information is not available, rank aggregation methods might be better. Hereby, it is worth comparing the performance of various combination methods while considering QoS factors. Moreover, in some cases, if the rank-based and score-based representations exist at the same time, we also want to compare the performance of the hybrid combination algorithms.

1.3 Research Goals and Approaches

In this thesis, firstly a QoS based ranking algorithm is proposed for the meta search. Two types of QoS attributes are also defined, one is performance-related, such as delivery of a web page, and the other one is personal preference related, such as the preference on how fresh the content of a page is, or whether a page access is subscription-based. In order to improve the ranking algorithm for mobile users or users with slow Internet connection, in the performance-related attributes, those attributes are included to measure the performance of the web page delivery such as reliability, response time, etc. In the user-preferred attributes, some user optional factors are added such as freshness, media richness, etc. In one word, the basic idea is that if two pages are similar to the query with the same level of relevancy, the page with a lower response time will get higher rank than the other one. The same principle can be applied to other QoS attributes.

To test and support the concepts, in the experimentation, a meta search engine was built based on the three most popular search engines – Google [2], Yahoo [3] and MSN [4]. After the ranking results from three of them were combined, QoS based ranking algorithm was used to generate a new retrieval result. The experiment result shows the gorgeous improvement in the top 10 results. Several rank aggregation methods were compared, and it was found that the QoS-based re-ranking algorithm achieves the best performance on the QoS values of the top 10 results.

The objectives and contributions of this thesis are:

- Define two sets of QoS attributes: performance-based and personal preference related

- Develop and implement several QoS based ranking algorithms
- Compare the results among these combination methods with different user preference parameter settings

1.4 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2 we review some literatures related to QoS, the combination methods used in meta search engines, and how to use QoS factors in Information Retrieval System (IRS). We also discuss the difference and similarity between our approach and those methods. In Chapter 3 we define some QoS attributes which we feel most useful to the web search. In Chapter 4, we describe our approach on how to combine QoS attributes in the ranking process. Experimental design and result analysis will be covered in Chapter 5, where the methods of evaluation of those algorithms are also discussed. Finally in Chapter 6 we conclude with a summary of our results and sketch future research directions.

CHAPTER 2

LITERATURE REVIEW

As indicated in a number of research works ([17] [18] [8] [34] and the references therein), different search engines crawl different parts of the web, have different web page collections, as well as have different indexing and ranking strategies. For example, different seeds for the “*crawler*” will create different web reflections in database. Different representations of web pages will also generate different reflections of those documents. As a consequence, meta search engines become popular because they are supposed to have a wider coverage of the web and be able to provide a better performance than any single search engine. When a user submits a query to a meta search engine, it will issue a set of reformatted queries based on the original query to n search engines, and get back n ranked lists, and then generate a single ranking by applying some specific combining and sorting methods. Many models for the problem of meta search have been proposed [17-30]. There are two types of combination algorithms. The first one is the rank-based aggregation algorithm [17-22] which only requires the ranks from each search engine. The second one is the score-based combination algorithm [23] [24] [25] which requires the use of relevant scores for each returned result.

QoS is another important topic for the thesis. It is first proposed in telecommunication networks. There are few papers which studied how to use the QoS information in the Information Retrieval System (IR system) [39-44]. They are mainly

used in digital libraries or other distributed information systems such as distributed search engine systems. We will review those papers after the combination part.

In the sequel, we provide more details about researches which are closely related to our work: meta search rank aggregation and their ranking strategies, including score-based combination and rank-based aggregation algorithms, and how to use QoS in distributed IR systems.

2.1 Rank Aggregation Algorithms

There are two categories of rank aggregation algorithms: (1) supervised rank aggregation algorithm [26], which needs the training data, and (2) unsupervised rank algorithm, which are the widely used algorithms with no training data.

2.1.1 Supervised Rank Aggregation Method

The basic idea of supervised rank aggregation is that, based on the unsupervised ranking algorithms, the supervised training is applied to get the optimal ranking result. Suppose that n search engines generate n rankings for a query. The overall retrieved pages are noted as A . The training sets are the pair-wise preferences, denoted by H . The score set S holds the scores for all the pages in A , which are generated by the rank aggregation method with a stochastically chosen parameter set. The supervised rank aggregation tries to find the best parameter set by means of training so as to minimize the disagreement between the training set and the result of this function by this parameter set. For example, assume that only 4 pages are retrieved. $A = \{ p_1, p_2, p_3, p_4 \}$ and

$S=\{s_1, s_2, s_3, s_4\}$ by a specific parameter set. In the training data, $p_1 > p_2$, and $p_4 > p_3$,

thereby the training set T can be noted as a metric $T=\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$. The

disagreement d can be formulated as: $d = T*S$ and $d < 0$. In fact, in order to bear the errors, a variable t is used, such that, $d = T*S$ and $d < t$. Liu et al [26] proved that the supervised rank aggregation methods outperform of the unsupervised rank aggregation methods, however, collecting training data is costly. In addition, although supervised rank aggregation can use those existing training data such as the one from TREC [35], we do not use this algorithm in this thesis. This is due to the fact that the main focus of this thesis is using QoS to improve the user experience, however, existing training data mainly focuses on the relevancy of documents.

2.1.2 Unsupervised Rank Aggregation Method

2.1.2.1 Positional Rank Aggregation Method

In the unsupervised rank aggregations, there are two types of rank aggregation algorithms: positional methods such as Borda-fuse [17], which assign scores to those elements in the input ranked list according to their positions in the list and majoritarian methods such as Condorcet-fuse [18] which are based on pair-wise comparisons. The latter considers the winner who wins the whole is that one who wins in all the pair-wise comparisons.

Borda-fuse Rank Aggregation

Positional rank aggregation methods consider the positions from the input rank lists as scores which are used to do the score combination. The Borda-fuse [17] [18] (or

Borda Count) algorithm was proposed to solve the problem of voting. It works as follows. Suppose there are n search engines, given a query, each of them will return a set of relevant pages. Now suppose that altogether there are c pages in the result set. For each search engine, the top ranked page is given c points; the second ranked page is given $c-1$ points, and so on. If one page is not ranked by a search engine, the remaining points are divided evenly among all the unranked pages. In such case, we could calculate the total points earned by each page, and rank all the c pages in the descending order. It is a simple procedure, which has been proved to be both efficient and effective. The weighted Borda-fuse algorithm was also proposed [17]. This algorithm assigns a weight to each search engine based on its average precision, and it has been shown [17] that its performance is better compared to the Borda-fuse algorithm. However, the hypothesis for positional methods is somehow questionable because the position cannot represent the difference between different ranks and also cannot represent the difference of the same rank in different rankings, resulting to a loss of precision. For example, in real applications, input ranked lists from normal search engines are usually partial lists. Dwork et al [18] proved in the experiments that, among all the top 100 documents from 7 input lists, 67% of the aggregation results were present in only one input rank list and less than two documents were listed in all the seven search engines.

2.1.2.2 Majoritarian Rank Aggregation Method

Majoritarian rank aggregation methods are another type of major approaches to the voting problem. They use pair-wise comparison to measure the value of a *candidate document* [18]. Condorcet-fuse is a representative approach in majoritarian rank aggregation methods.

Condorcet-fuse Rank Aggregation

The basic idea is that in the Condorcet, the winner should win (or ties in) every possible pair-wise majority contest. If one need to compare two pages A and B , the initial value of a counter is set to zero, then for each search engine, if it ranks A is higher than B or if B is the only missing document [18], the counter is incremented, otherwise, it is decremented or kept unchanged. After iterations over all candidate search engines, if the counter is a positive value, A is ranked better than B , otherwise B is ranked better than A . By going through all pages in the result set, a ranked order is obtained. In the weighted Condorcet-fuse, a weight can be assigned to each search engine. The pair-wise comparison algorithm has a *tie* problem. For example, in case that among three search engines α , β and γ , we compare three pages A, B and C in three search engines, in search engine α , A wins B but is beaten by C ; In search engine β , B wins C but is beaten by A ; in search engine γ , C wins A but is beaten by B . It is a *tie* when there is a cycle, i.e., we cannot assign a rank to any of the pairs (A, B) , (A, C) , or (B, C) . In order to break the *tie*, the author treated the *tie* as a set and once the set beats the other pages outside, it wins. However, this solution ignores the majorities in some pair-wise matching. Therefore, Condorcet-fuse uses unstable rank information, but is more robust than Borda-fuse.

2.1.2.3 Other Rank Aggregation Methods

Footrule Optimal Rank Aggregation

Footrule optimal rank aggregation [36] (*FOA*) is a consensus ranking algorithm that minimizes the *Spearman Footrule Distance (SFD)* of the input rank lists. The SFD between two permutations of a full list l_1 and l_2 is obtained as:

$$F(l_1, l_2) = \sum_{\forall i} |r_i^{l_1} - r_i^{l_2}| \quad (2-1)$$

where i is the page index which is from 1 to the size of the full list. l_k is the k -th permutation of a full list. $r_i^{l_k}$ is the position value of page i in the full list l_k . It is worth noting that the sizes of l_1 and l_2 are the same. Since l_1 and l_2 are from different search engines, $|l_1|$ and $|l_2|$ might be different, thus requiring to convert l_i into the full list

$$l'_i = \begin{cases} \text{unchanged} & , i \leq |l_i| \\ x | x \in U \vee x \notin l_i, i > |l_i| \end{cases} \quad (2-2)$$

where i is also the page index which is from 1 to the size of the full list. x is the page in a full list. l_i is the permutation of a partial list. l'_i is the converted permutation of a full list.

It extends to multiple input rank lists as well. Beg [21] used a genetic algorithm (GA) to optimize the Spearman Footrule Distance. Here, the GA randomly generates a ranking, and then calculates the SFD with those sources' rankings to find an optimal ranking with minimum SFD. The author found the performance of the GA based footrule distance method to be much better than the Borda count. However, the runtime GA

without training data takes too much time for optimization. Therefore, he used parallel techniques, but still the GA is slower than Borda count.

Median Rank Aggregation

Median rank aggregation is another type of positional algorithm which picks up the documents ordered by their median ranks. Fagin *et al* [20] proved that median rank aggregation method is an efficient and useful form which combines multiple factors. In median rank aggregation, supposed that given n documents and m lists of values that are assigned to the n elements, the final rank is generated by following steps. First of all, the n documents are sorted based on m lists of values, thus getting m ranked lists. Then, the final rank value is calculated as the median of the positional values of each document. The final ranked list is also generated by sorting the median value of each document. Median rank aggregation method can break *ties* which are the main problem in Condorcet-fuse.

2.2 Score Combination Algorithm

Linear Score Combination Algorithm

The simplest and most commonly used score combination model is the linear combination model. In this kind of algorithm, the scores or performances of multiple factors of each page are combined by aggregation operations such as weighted sum so as to get the overall aggregation values, and then generate the ranked list based on the combined score. For instance, Fox and Shaw [23] tested CombSUM, CombMNZ and the other three methods, and found that CombSUM performs the best. Lee [24] tested the

same five combining methods, and found that CombMNZ performs better than others. Bartell *et al* [22] added scalars in the linear model. They used numerical optimization techniques to determine the optimal scalars for a linear combination of results. A detailed review of different score combination IR systems can be found in [33].

The score combination method requires the score of every web page, and it is usually unavailable in meta search engines. There are some algorithms proposed [37] [38] [20] to estimate the score based on the rank, mostly by a linear mapping [37], or by using a logistic function to do the mapping [38], or a linear regression [20].

2.3 Applying QoS in IR System

QoS has been an active research topic in several areas. Rocha *et al* [39] tried to analyze several performance factors of a wireless mobile network and found that the user mobility and the cell size can directly impact the QoS of the whole communication network. Hereby, adding these two factors in QoS control can greatly help to simulate and design a wireless network. Canonico *et al* [40] developed an API for common distributed systems and it included the QoS control features that can be easily deployed in the distributed application system. In the paper, he claimed that QoS-based script language is good to program for the real-time and multimedia distributed applications. Xiong *et al* [41] used QoS attributes such as packet loss rate and frame rate to control the traffic in multimedia communication under heterogeneous environments such as wired and wireless hybrid networks. And the experiment shows that QoS control is useful and efficient in such network environment.

Recently, QoS also becomes a very important issue in web service area. Our review focuses on the web service area because how they use QoS in web services is similar to how we can use QoS for web search. In [16], Ran described a list of QoS attributes and how they can be used in web service discovery. The QoS attributes are categorized into four types: runtime related, transaction support related, configuration management and cost related, and security related. It is quite a complete list, and we define the search-related QoS attributes based on this list. We mainly focus on the first type because other types are not of direct concerns to search engine users. Usually, QoS is a matter of personal preference or restrictions from the user device. Different users have different expectations on QoS and their ratings on the service will be closely related to their expectations [42]. So in this work, we only consider QoS attributes that users prefer.

There are a limited number of papers studying how to use the QoS information in IR systems, and they mainly use it in digital libraries or other distributed information systems. Kapidakis et al [43] provided the resource allocation and the distributed searching and retrieval based on QoS attributes, such as the server load, network load and reliability, etc. They focused on how to provision, monitor and manage QoS in such an environment. Barker [44] tried to help users of the X.500 directory to understand the unevenness of QoS with a widely distributed directory, especially the highly variant response times and availability of information for different parts of the directory.

In this chapter, we reviewed some research closely related to this thesis, such as score combination and rank aggregation algorithms including supervised rank aggregation, Borda count algorithm, Condorcet fuse algorithm, Median rank aggregation

and GA-based rank aggregation etc. We also review the developments of QoS in Web service and distributed IR systems. In the next chapter, we will give the definitions of some important QoS attributes and details on how to collect and normalize the QoS values.

CHAPTER 3

QoS ATTRIBUTES RELATED TO RANKING

In this chapter, we will define some QoS attributes that are related to page ranking after we give an overview of the QoS attributes.

3.1 Overview

Some common QoS attributes include throughput, response time, security, reliability, availability, cost, reputation, accuracy, etc. [16]. In a web search engine, accuracy and reputation are widely used to measure the relevancy and the quality of a result whereas most of the performance-related attributes are seldom used in information retrieval systems, such as reliability, stability or response time. Therefore, we want to define several new performance-related QoS attributes that might be useful in web search.

Many QoS attributes have been proposed in the communication or web service areas, but not all of them are applicable to the web search. Based on our observation and understanding over the common user behaviors in the web searching process, we identify a list of QoS attributes that we believe are quite related to the web search and then categorize them into several types.

Generally there are three types of QoS attributes that might be of concern to the web users. The first type is *content-related*, which could be used to measure whether the

content of a web page is relevant to the user's query or with a high quality. The traditional similarity score calculation, link connectivity analysis or the click rate can be used to measure this type of QoS attributes. Another type of QoS attributes is *performance-related*, which usually measures how well the web page is delivered to the end users. Users normally want a web page with a high delivery speed, a satisfying reliability and sustained availability, and also from a web site that can handle large volumes of simultaneous user requests. The requirement for this type of QoS is usually the same for most of the users. The third type is *preference-related*, which include various measurements based on personal preferences. Different users have concern for different QoS attributes, and even for the same attribute, they have different requirements on the value ranges. For instance, for the same query one user might prefer a web page with a lot of multimedia objects, whereas another might prefer a pure text page.

Most of search engines or web IR systems deal with the first type of quality, and also the third type to a certain extent. For example, in most of the search engines, top 10 results are always the best relevant results. Moreover, most of the search engines such as Google have an advanced search option which allows users to input their profiles so as to designate the language of results or the file type to be retrieved, etc. However, very few consider the type2 or the type3 of QoS attributes, which indeed is the focus of this thesis. We are not going to give an exhaustive list of QoS attributes belonging to type2 and type3, and instead, we choose several most important and representative quality attributes. For the type2, we choose reliability and response time. The ability to handle a big volume of requests is not considered because it would be hard to measure it without the cooperation from the web site owner. For the type3, we choose the file size, the number

of embedded multimedia objects, and the data freshness. The file size and multimedia objects can be reflected in the response time to a certain extent. The reasons we consider them as separate QoS attributes are twofold: firstly, response time is determined not only by the file size, but also by the network speed, the volume of traffic, the stability of the web server, etc; secondly, the file size and multimedia objects can contribute to other quality attributes too, such as the cost of the Internet access. Also compared with the response time, they are more of a personal preference.

There are many more attributes which are preference-related, e.g. whether a web page can be freely accessed or its access requires a registration or subscription, or whether a subscription requires a fee to access the content or only the personal information, or whether a web page access is SSL-based or password-required. As mentioned, we would not be able to consider all of them, however our system framework is flexible and expandable so as to easily include other new QoS attributes. The granularity of these attributes can be adjusted if necessary. For example, if a user does not like any flash file (*swf* as the suffix file name) in a web page, the multimedia objects can be divided into several types depending on the media types. Preferences such as these can be easily controlled by a configuration file. In addition, the data types of these QoS attributes can also be adjusted. Currently, all of them are continuous numeric values, but they could also be a discrete or categorical value. For instance, instead of being a numeric value, the reliability can be categorized into three ranges: low, medium, and high.

3.2 QoS Attribute Definition

In the following subsection we explain the five QoS attributes we choose, and how we measure and calculate them.

Reliability – it measures how reliable a web site is and whether it could deliver a web page without an error. It is defined as the percentage of successful page delivery rate among all the page requests. In order to measure it, we make a number of requests to the same web page during different times of a day and different days of a week. Each time, if the HTTP response code is 2** [15], it is considered as a successful delivery, or otherwise, if it is a server error, a client error, or a page move, it means a failure. Since the reliability of a web page may not be constant depending on the network traffic, in order to have a more accurate measurement, we calculate the reliability by getting the average value of the requests to the same page.

Response time – it measures how quickly a user can see the web page after a request has been submitted to the web server. It is defined as the difference between the time when a user starts to send a request and the time when the user receives the response from the web site. Since both the network and the web server cannot have a stable speed all the time because of the change of the incoming and derivative traffic and the internal status of the server itself, the response time may vary for a page when it is accessed at different times. In order to measure the response time more accurately, we will submit the HTTP request to the same page several times, and generate the average value of the access time as the response time of this page.

File size – it measures the pure text size of the web page excluding all the embedded objects because these embedded objects are considered in another separate QoS attribute. This attribute can be obtained from the HTTP header or by downloading the page. Since some web servers might not fill the file size field in the page's HTTP header, we always use the size of the downloaded page. If the HTTP request is timed out, a negative value such as -1 will be assigned to this QoS attribute and will be specially treated later in the rank aggregation part. Details will be provided in the next chapter.

Media richness – it measures the number of embedded multimedia objects within a web page. The reason to calculate this factor is that the more the multimedia objects are embedded in a web page, the more the downloading flux will happen. Media richness implies the speed of delivering the whole web page and further the cost of downloading time and flux. Hereby, this factor could be helpful especially for the mobile users or users with slow network connections.

Data freshness – it measures how fresh the web page is. A recently updated page is considered fresher than a page that was created years ago. It is defined as the time difference between the last modified date and the time when the page is accessed. In case that there is no last modified date specified for a page in the HTTP response, a predefined value will be used. If the last modified date is the same as the one in the previous access, the value of data freshness factor will not be changed. For the freshness, we also send the

same request several times to calculate the average value, and in this way, we could more accurately measure a frequently updated or newly updated page.

As we have mentioned, one of the objectives we want to achieve in this thesis is to improve the ranking algorithm for mobile users or users with slow Internet connection. For these types of users, the first four QoS attributes are especially important. The first two directly affect the satisfaction level of their Internet experiences, and the latter two determine the cost of their Internet access. Although file size and media richness are both preference related attributes, for this particular type of users, we assume that they prefer a smaller file size and a smaller number of multimedia objects, which we believe is a reasonable assumption. The data freshness attribute is more for general Internet users, and by including it, we could illustrate the possibility of including different preference-related QoS attributes in our system.

In summary, in this chapter, we discussed the three categories of QoS attributes which were used (such as relevancy, popularity) or should be used in IR systems and why we want to define some new attributes for the web search. In addition, we give a definition for five important QoS attributes which will be used in the experiment.

CHAPTER 4

COMBINING QoS ATTRIBUTES IN RANKING

After we define the search-related QoS attributes, we need to decide how we can integrate them into the ranking procedure. The content of a page, whether it is relevant to a query, and whether it has a high quality based on its linkage connectivity or its click rate, are still the main sources in the page ranking process. Since most of the search engines consider all of them, by building a meta search engine based on these existing search engines, we can deploy them directly, and then QoS attributes are used to re-rank the search results.

There are six ways to re-rank the original search results. The first two are score-based combination methods. The last four are rank-based rank aggregation methods.

The first one is to re-rank the search results from the meta search engine based on the five QoS attributes. The second way is to calculate the QoS values for all the pages and then do the linear score combination with the normalized rank scores converted from the ranks. The normalization details will be provided in the following subsection. The third one is to apply Condorcet-fuse [18] to do the rank aggregation on the QoS-based rankings and the original ranking from the meta search engine. The fourth method is to do Median rank aggregation [20]. And the fifth algorithm is to do Borda-fuse [17] rank aggregation. The last one is to do Partial Footrule Optimal Distance rank aggregation based on Genetic Algorithm [21]. All the details of these combination methods are

discussed in the rest of this chapter. It is worth noting that from the second algorithm to the last one, the data sets in the rank aggregations are the same, which include the five rankings from those QoS attributes and the original ranking from the meta search engine.

4.1 System Framework Overview

For this thesis, a meta search engine is built from the most popular search engines. The queries in the query set are submitted to this meta search engine, and we calculate the QoS values of the top ranked pages returned from the meta search engine, and then use these collected QoS values to re-rank those pages so as to generate a new ranking which is hoped to be able to better satisfy users. In order to describe our meta search engine, a simple architecture is shown in Figure 3.

Figure 3. The process of meta search engine based on QoS

In the meta search engine shown above, there are two additional processes which are different from the normal meta search engines: QoS collection and evaluation, and QoS re-ranking. QoS collection is to automatically measure and calculate the values of QoS attributes for the pages returned from the original meta search. After the QoS collection, QoS attributes of those pages are evaluated and the values are saved in the database. This process can be automatically executed in the background. The other process is QoS re-ranking which is to generate another ranking based on the QoS attributes for those pages and the original ranking. In the following sections, we will describe the details about these two processes.

4.2 QoS Collection and Evaluation Model

The first step is to build the meta search engine. We choose the three most popular search engines – Google [2], Yahoo [3] and Live Search [4] as the underlying search engines. In the first step, meta search engine will combine the results from these three search engines and generate the first ranking result for the latter steps. Since only the rank information is available from each of these search engines, we decide to use the rank aggregation method to combine the three ranked lists. Condorcet-fuse and Borda-fuse are two commonly used rank aggregation methods. In [19], it has been proved that Condorcet-fuse can achieve a better performance than Borda-fuse because of the limitation of this positional rank aggregation method (see details in 2.1.1). So we just choose Condorcet-fuse as the rank aggregation method for the meta search engine.

Usually users won't go beyond the first two pages while browsing the search results. [45] [46] Therefore, we only focus on the results of top 20 pages from those three search engines instead of all the results. Only for these top-ranked pages, we collect their QoS values. In the second step, for each page in top n positions, we calculate all the five QoS attributes based on the previous QoS definitions in section 3.2, and normalize the values to (0, 1) range using the simple normalization method (please refer to Equation 4-1 and 4-2) and save in the database tables. The reliability value is in (0, 1) range already, and thus we don't have to do the normalization for this attribute.

For the response time, because the page with a lower response time should be ranked higher, the normalization formula is as below:

$$NQoS_{ij} = \frac{Max_j(QoS_{ij}) - QoS_{ij}}{Max_j(QoS_{ij}) - Min_j(QoS_{ij})} \quad (4-1)$$

where i is the page index which is between 1 and size of retrieved results. j is the index of QoS attributes. QoS_{ij} is the value of the j -th QoS attributes for page i , $NQoS_{ij}$ is the normalized value of QoS_{ij} , Min and Max are operations to get the minimum and maximum values. In the current system, j is a value from 1 to 5 because we have only 5 QoS attributes, and it could be expanded later.

For the other three attributes (data freshness, file size, and media richness), because they are preference related, the page with a lower value could be ranked higher or lower depending on the user's preference. If we take media richness as an example, if the user prefers a lower value on media richness, the normalization formula is the same as the previous one, otherwise, if the user prefers a higher value, we would use a different formula as shown below:

$$NQoS_{ij} = \frac{QoS_{ij} - Min_j(QoS_{ij})}{Max_j(QoS_{ij}) - Min_j(QoS_{ij})} \quad (4-2)$$

where i is the page index which is between 1 and size of retrieved results. j is the index of QoS attributes. QoS_{ij} , $NQoS_{ij}$, Min and Max in Equation 4-2 are the same definition as those variables in Equation 4-1. In our current experiment, we assume the lower value is preferred on these three attributes.

4.3 QoS Based Combination and Re-Ranking Model

After QoS collection, all the values of the QoS attributes are saved in the database and ready to be used in the QoS re-ranking. We have implemented six types of re-ranking strategies which are QoS based re-ranking method, linear score combination with original rank, Condorcet rank aggregation, Median rank aggregation, Borda rank aggregation and Genetic Algorithm based Partial Footrule Optimal Distance rank aggregation (GA based PFOD rank aggregation). These algorithms can be divided into two types: QoS score combination and QoS rank aggregation. The first two are QoS score combination and the last four are QoS rank aggregation.

4.3.1 QoS Based Re-Ranking

Since the main purpose of this study is to test if and how the QoS attributes can be integrated into the ranking process, and if so, how can they be integrated. In this QoS based re-ranking method, we choose the simplest and the most common score combination method – the linear combination.

After the normalization step, we do the simple linear combination on the five normalized values and then each page will have an overall QoS value.

$$QoS_i = \sum_j w_j \cdot NQoS_{ij} \quad (4-3)$$

where i is the page index which is between 1 and size of retrieved results. j is the index of QoS attributes. QoS_i is the overall QoS value of page i , $NQoS_{ij}$ is the QoS value of page i on the j -th QoS attribute and w_j is the weight of the j -th QoS attribute. The values of

the weights are between 0 and 1. The value of 0 means that the user does not care about this QoS attribute, and on the contrary, 1 means the QoS attribute is desired by the user. Any value in-between could indicate the relative importance of this QoS attribute. Since normally users are reluctant to set up their personal preferences in details, currently we only consider two borderline cases – 0 and 1, to simulate the user's opinion (important or unimportant).

In QoS based re-ranking method, when a user submits a query to the meta search engine, it calculates the QoS values for the top n ranked pages, and then these n pages will be re-ranked based on their overall QoS values.

4.3.2 Linear Score Combination with the Original Rank

In this method, since we don't have the scores returned from the meta search engine, we convert the rank to the score using the linear mapping [37]. Then we could combine it with the other five QoS values,

$$S_i = \alpha \cdot OS_i + \beta \cdot \sum_j w_j \cdot NQoS_{ij} \quad (4-4)$$

where i is the page index which is between 1 and size of retrieved results. j is the index of QoS attributes. S_i is the overall score of page i , $NQoS_{ij}$ and w_j are the same meaning as those in Equation 4-3. OS_i is the converted score of page i from the original ranking, and α and β are parameters to measure the weight of each part of the score. In the current stage, we set α and β as 1, which means two parts of the score are equally important.

4.3.3 Weighted Condorcet Rank Aggregation with QoS Attributes

Similar to the previous method, we also get the top n pages from the meta search engine and calculate the normalized five QoS values for each of them. Next step is different. We rank these n pages based on the values of each QoS attribute, and afterwards we get five rankings. With the original ranking from the meta search engine, altogether we have six rankings. Based on users' preferences, each QoS-based ranking has a weight, and we use the weighted Condorcet-fuse [18] to aggregate these six rankings. The basic idea of the Condorcet aggregation algorithm is to compare the positions of the pages pair-wise. Since all the pair-wise comparisons are available, we choose Quick Sort algorithm to rank the n pages. The one who wins is the one who wins in all the pair-wise comparisons. A "SimpleRun" function is shown in Figure 4 to compare the relationship between two pages with the weights of QoS attributes. And another Figure 4-3 shows the process of Quick Sort algorithm for the ranking.

Assume that two pages d_1 and d_2 are compared as the input parameters

1. **for each** ranking R_i of QoS attributes plus original ranking
2. **if** $d_1 \in R_i$
 if $d_2 \notin R_i$
 $Sort_{d_1} += Weight_i$
 else if $Rank_i(d_1) > Rank_i(d_2)$
 $Sort_{d_1} += Weight_i$
 else if $Rank_i(d_1) < Rank_i(d_2)$
 $Sort_{d_1} -= Weight_i$
 end if
 end if
3. **else if** $d_2 \in R_i$
 $Sort_{d_1} += Weight_i$
4. **end if**
5. **end for**
6. **if** $Sort_{d_1} > 0$ d_1 wins
7. **else** d_1 loses

Figure 4. SimpleRun algorithm in weighted Condorcet rank aggregation

where i is the index of QoS attributes. R_i is the ranking for the i -th QoS attribute; $Weight_i$ is the weight of this i -th QoS attribute; $Sort_{d_1}$ is the variable that stores the value of the page d_1 ; $Rank_i(d_j)$ is the function that returns the position value of d_j in the ranking for the i -th QoS attribute. j is the page index which is between 1 and size of retrieved results.

QuickSort(A, p, r) :

Assume that two input parameters p and r are the indexes of lower bound and upper bound of the input ranking A

1. if $p < r$
 2. $i = \text{Partition}(A, p, r)$
 3. QuickSort($A, p, i-1$)
 4. QuickSort($A, i+1, r$)
 5. end if
-

Figure 5. Weighted Condorcet rank aggregation using Quick Sort ranking algorithm

Assume that two input parameters p and r are the indexes of lower bound and upper bound of the input ranking A

1. $i = p - 1$
 2. for j from p to r
 3. **if** $A_j > A_r$
 $i++$;
 $\text{exchange}(A_i, A_j)$
 4. **end if**
 5. **end for**
 6. $\text{exchange}(A_{i+1}, A_r)$
 7. **return** $i+1$
-

Figure 6. Partition algorithm for QuickSort

In Figure 5, A is the input ranking which need to be re-ranked by QuickSort. $\text{Partition}(A, p, r)$ is a function which swaps all the data in A from p to r when the data in this scope are less than A_r , and finally returns a new index of A_r . QuickSort is a recursive function which firstly invokes the Partition function to find the i -th position for the pivot A_r and also re-rank A so as to make members before r are less than A_i and the

members after r are bigger than A_i ; then QuickSort recursively invokes itself twice to re-rank the members between 1st to the i -th and the members after the i -th; eventually all the members in the ranking A will be rank based on pair-wise comparisons. In Figure 6, $exchange(A_i, A_j)$ is a function which exchanges the data in the i -th and j -th positions in the ranking A . A_i means the value of i -th position in the ranking A .

4.3.4 Median-Rank Aggregation with QoS Attributes

In this method, we go through the same process as the previous one to get five rankings for the five QoS attributes. And also with the original ranking from the meta search engine, we have six rankings. Based on users' preferences, each QoS-based ranking has a weight, and then we use the weighted Median rank aggregation [20] to aggregate these six rankings.

The median value for each page is calculated by the following formula 4-5,

$$M_i = \begin{cases} middle(PosRanking), \text{when } |PosRanking| \text{ is odd} \\ \frac{PosRanking_i + PosRanking_{i+1}}{2}, \text{when } |PosRanking| \text{ is even and } i = \frac{|PosRanking|}{2} \end{cases} \quad (4-5)$$

where i is the page index which is between 1 and size of retrieved results. $PosRanking$ is the ranked list of the positions of each page in the six rankings described above. In each one of the six rankings, a page will get a position in the ranking. Altogether, one page has six positions. Considering users' preferences, each QoS-based ranking has an associated weight. Currently we only consider two borderline cases of weight – 0 and 1. Therefore, if the weight of a QoS attribute is 0, this position will be removed from the set of positions. $|PosRanking|$ is the size of the ranked list of the positions after removing

those positions whose weight of the associated ranking is 0. *middle* is the function which can get the value of the middle element from a ranked list. It is worth noticing that the *middle* function is only called when the size of the ranked list is odd. The even condition is considered as the second case in the Equation 4-5. And a new ranked list will be generated by sorting the median position of each page.

4.3.5 Weighted Borda Rank Aggregation with QoS Attributes

Assume that c is the initial parameter that represents the number of retrieved pages

1. **for each** page d_i in the ranking A
 2. **for each** ranking R_j of QoS attributes plus original ranking
 3. **if** $d_i \in R_j$
 - $tmp = c - Rank_j(d_i) + 1$
 - if** $tmp > 0$
 - $Sort_{d_i} += Weight_j \times tmp$
 - end if**
 - end if**
 - end for**
 4. **end for**
 5. QuickSort($A, 0, sizeof(A)$) //use QuickSort to rank the d_i in the ranking
 // A based on the value of $Sort_{d_i}$
-

Figure 7. Weighted Borda rank aggregation algorithm with QoS attributes.

Similar to the Condorcet aggregation method, we get six rankings. Based on users' preferences, each QoS-based ranking has a weight, and we use the weighted Borda-fuse [17] to aggregate these six rankings. In Figure 7, i is the page index which is

between 1 and size of retrieved results. j is the index of QoS attributes. R_i is the ranking for the i -th QoS attribute; $Rank_j(d_i)$ is the function that returns the position value of d_i in the ranking for the j -th QoS attribute. $Weight_j$ is the weight of this j -th QoS attribute; $Sort_{d_i}$ is the variable that stores the value of the page d_i .

4.3.6 Genetic Algorithm Based Weighted Partial Footrule Optimal Rank Aggregation with QoS Attributes

The basic idea of this rank aggregation algorithm is to find an output ranked list which has an overall minimum *Spearman Footrule Distance (SFD)* between the output rank list and all the candidate rank lists. *Spearman Footrule Distance (SFD)* is a measurement of the difference between two ranked lists l_1 and l_2 . In order to get the best performance of all QoS attributes, the best ranking need to consider all the QoS attributes, hereby the total *SFD* of the result rank list will be a sum of the *SFDs* with all the six rank lists. Assume that two ranked lists l_1 and l_2 are full lists, which means all the elements in l_1 are also in l_2 , vice versa, and the size of l_1 , $|l_1|$, is the same as $|l_2|$. The *SFD* between l_1 and l_2 is

$$SFD(l_1, l_2) = \frac{\sum_{i=1}^{|l_1|} |l_1(d_i) - l_2(d_i)|}{\left[\frac{|l_1|^2}{2} \right]} \quad (4-6)$$

where i is the page index which is between 1 and size of the full list. $SFD(l_1, l_2)$ is the *SFD* between two ranked lists l_1 and l_2 . d_i is the i -th page in the ranked list. $l_1(d_i)$ and $l_2(d_i)$ are the functions to find the position value of d_i in the ranked list l_1 and l_2 .

Assume that there are a set of k ranked lists $L(l_1, l_2, \dots, l_k)$, then the normalized SFD between the ranked lists l_1 and L is

$$SFD(l, L) = \frac{\sum_{i=1}^k SFD(l, l_i)}{k} \quad (4-7)$$

where i is the index which is between 1 and k . k is the number of permutations of the full list. $SFD(l, L)$ is the normalized SFD between the ranked lists l_1 and L . $SFD(l, l_i)$ is the SFD between two ranked lists l and l_i .

As for our understanding, an optimal result with QoS attributes should be able to improve the QoS performance while keeping the original level of relevancy. Therefore in this thesis, L is the six ranked lists as in the previous algorithm. We apply Genetic Algorithm to search the optimal result to minimize the SFD between the output result ranking l and the six ranked lists L in this NP-hard problem.

Genetic Algorithm (GA)

Genetic Algorithm (GA) is a type of heuristic optimization algorithms which simulate Darwin's nature selection and evolutionary theory. It has been widely used in finding the best or approximately best results to some problems which don't have or hardly find clear solutions. GA exploits the probability theory to recursively find the optimal result by random search, which reflects the process of mutation and selection. The possible solutions are the individuals (reflecting to *chromosomes* or *gene* in Biology) from the entire collection which is called as population (reflecting to *gene pool* in Biology). By applying the concept "*Survival of the fittest*", there is a fitness value to reflect the level of a individual that adapts to its environment, in other words, fitness

means the adaptation of solutions to the problems. The gene or individual can be represented by a fixed-length binary string or a fixed-length integer or real array. Genetic programming (GP) is relatively close to GA. However, GP can handle more complex data structures such as trees, linked lists or stacks. [11]

Genetic Operations

Genetic Algorithm imitates three major operations in nature selection: selection, crossover, and mutation.

Selection is a genetic operation which chooses relatively adaptive individuals in the population to reproduce the next generation. Therefore, in some papers, this operation is called *Reproduction*. [7] Since choosing to reproduce is probabilistically related to the fitness value which individuals adapt to the outside environment (or parameter), sometimes we also call this operation as *differential reproduction*.

The steps of Selection are: first, normalize the fitness value for each individual, and use the normalized result as the relative fitness value. The formula to normalize the fitness value is

$$Rfitness_i = \left(\frac{fitness_i}{\sum_{i=1}^n fitness_i} \right) \times 100 \% \quad (4-8)$$

where i is an integer from 1 to n , and n is the size of population. Then, the cumulative fitness value is calculated by the following formula:

$$Cfitness_i = \begin{cases} cfitness_i + cfitness_{i-1} & (2 < i \leq n) \\ Rfitness_1 & (i = 1) \end{cases} \quad (4-9)$$

where i is an integer from 1 to n , and n is the size of population. By given a random value as the probability p for fitness, the individuals in the population will be selected by this probability factor. This procedure mimics the reproduction in nature life. By using the randomly selected probability p , GA simulates the randomly chosen event of reproduction in nature. Each reproduction has a specific random probability p . If the probability $p < Cfitness_1$, it means the first gene is lucky to be selected to next generation; else GA will find an index i where $Cfitness_j \leq p < Cfitness_{j+1}$, which means the cumulative fitness value of the j_{th} gene is good enough to be selected. From the formula, we can observe that those individuals with higher fitness value can get higher probability to reproduce themselves to next generation; on the other hand, those individuals which has lower adaptation will get less chance to reproduce themselves and might be eliminated through selection from the population in several generations. However, since those are probability events, all the individuals do have chance to be reproduced.

Crossover is to exchange the gene from the same part of two different individuals so as to create two new individuals for the new generation. The position of exchanging the gene will be randomly decided. In order to simulate the crossover events in nature, there is also a concept “*probability of crossover*” which reflects the probability that crossover happens.

Figure 8. An example for crossover in Genetic Algorithm

For example, for each individual in the population, GA will generate a random value which is the position to crossover the two candidates. Assume two fixed-length individuals are $G1 = 1,2,3,4$ and $G2 = 5,6,7,8$. The probability of crossover is p_{cross} . And the position to crossover is 0.5. For each individual in the population, generate a random value p_i , if $p_i < p_{cross}$, then this i_{th} individual will be one of the parents whose gene will be involved in the crossover to generate the new individuals for next generation. Normally the value of p_{cross} is from 0.25 to 0.75.

In *SFD* algorithm, since two ranked lists l_1 and l_2 are full lists, which means after crossover operation, the new ranked lists l_1 and l_2 should be full lists as well. Therefore, we apply multiplication of permutations when doing the crossover operation. [21] [47] Assume $l_1 = \{e_{1,1}, e_{1,2}, \dots, e_{1,n}\}$ and $l_2 = \{e_{2,1}, e_{2,2}, \dots, e_{2,n}\}$. In the j -th position the crossover happened, we have the new l'_1 and l'_2 :

$$l'_1 = \{e_{2,1}, \dots, e_{2,j-1}, e'_{1,j}, \dots, e'_{1,n}\} \quad (4-10)$$

and

$$l'_2 = \{e_{1,1}, \dots, e_{1,j-1}, e'_{2,j}, \dots, e'_{2,n}\} \quad (4-11)$$

, where j is the value between 1 and n . n is the size of this permutation.

$$\{e'_{1,j}, \dots, e'_{1,n}\} = \{e_{2,j}, \dots, e_{2,n}\} \times \{e_{1,j}, \dots, e_{1,n}\} \quad (4-12)$$

and

$$\{e'_{2,j}, \dots, e'_{2,n}\} = \{e_{1,j}, \dots, e_{1,n}\} \times \{e_{2,j}, \dots, e_{2,n}\} \quad (4-13)$$

where $e'_{1,j}$ is the new element in permutation1 after crossover. $e'_{2,j}$ is the new element in permutation2 after crossover. For the multiplication of two permutations, in general notation, if $e_1 = \{e_{1,1}, \dots, e_{1,n}\}$ and $e_2 = \{e_{2,1}, \dots, e_{2,n}\}$ are two permutations, the product of multiplication $e_1 \times e_2 = \{e_{1,e_{2,1}}, \dots, e_{1,e_{2,n}}\}$. For example, if $e_1 = \{4,3,1,2\}$ and $e_2 = \{1,4,3,2\}$, then $e_1 \times e_2 = \{4,2,1,3\}$.

Mutation is a mechanism that provides the new individuals with new traits by the nature selection. Those individuals with advantage traits will have higher adaptation to survive and reproduce more offspring with the adaptive traits, however those individual with disadvantageous traits will have lower adaptation so as to die out from the population. Therefore, adaptive traits will remain in the population; on the contrary, traits that are not adaptive will be eliminated from the population by nature selection. A selected trait or gene variable is selected randomly by the probability of mutation $P_{mutation}$. For each gene in every individual in the population, a real value between 0 and 1 is randomly generated. It is the current probability of mutation for this gene p_i . Only if $p_i < P_{mutation}$, the mutation for this gene will happen. And a randomly generated value will replace this gene. In most cases mutations are harmful to individuals and too

many mutations might make those traits with higher adaptation unstable in the population. Thereby, the value of $P_{mutation}$ is small and is usually between 0.01 and 0.2.

Elitist is a mechanism that guarantees that the individual with best fitness remains in the population. It plays an important role in nature selection. Sometimes, we can call it "*Elimination*" mechanism. First, if the best individual of current generation is worse than the one of previous generation, the individual with worst fitness in the current generation will be replaced by the one with best fitness in the previous generation. Consequently, the trait with the lowest adaptation is eliminated from the population and the trait with high adaptation is multiplied into the population. If the best fitness of current generation is higher than that of previous generation, it means the traits with the best adaptation have a large probability in current population. Therefore, we only need to record the best individual with the highest fitness value and go to the next generation.

Evaluation is the measurement to calculate the adaptation for each individual in the population. The value of evaluation is called the fitness value for the individual of the population. A simple procedure of GA is demonstrated in Figure 9.

1. Generate an initial population of random values under a certain constraint
2. Perform the following steps for n_{gen} generations.
Calculate the fitness value of each individual in current population
Create a new population by:
Selection (Reproduction)
Crossover
Mutation
Check convergence to the global optimum
3. Select the best performing individual as the discovery output.

Figure 9. Simple procedure of genetic algorithm

After those GA operations for n_{gen} generations, an optimal ranking result will finally be gained which has the minimum *SFD*.

To sum up, in this chapter, six combination methods are described in details and also how to re-rank the search results using QoS attributes defined in the previous chapter is shown. The next chapter will further talk about the environment, procedures and evaluation methods in the experiment and also include the result analysis.

CHAPTER 5

EXPERIMENT DESIGN AND RESULT ANALYSES

Two programs are implemented: QoS Watcher program is to collect the values of QoS attributes and QoS Analysis program is to do QoS re-ranking. In the QoS Analysis, six combination methods are employed and tested on thirty-six queries with thirty-one different sets of user preference parameters. Because currently we only consider the two boundary cases, 0 and 1, as the value of weight associated to a QoS attribute, for the five QoS attributes, we have 32 parameter settings. (e.g, "1 1 1 1 1"). However, the parameter setting "0 0 0 0 0" does not affect the result, therefore this parameter setting is removed from user cases, thus for each query, there are 31 user parameter setting to test. The results are evaluated from the data which are collected by QoS Watcher running for ten times. These evaluations show how much the QoS performance can be improved by using the new re-ranking algorithm and compare the difference when using different combination methods. The following sections will describe more details on data source preparation, experiment design, retrieval process and analysis of result analyses.

5.1 Data Set

Before giving the experimental details, the test queries and the database table description are given in this section.

5.1.1 Query

We chose thirty-six queries as our query set. In order to better represent the actual user queries, the topics are selected from a wide variety of subjects such as computer science, biology, tourism, etc. Although usually *a typical search query is short*, [48] some users might be used to use a short phrase to present their interest, therefore, there are one-word queries, two-word queries, and multi-word queries. In addition, in order to test the different response time and reliability, some topics are country related because the web sites hosted in different countries may have different values on these two QoS attributes. These 36 queries are listed in Table 1.

ID	Query	ID	Query
1	Disney	19	Cheap hotel china
2	Trichloroethylene	20	Cubic crystal system
3	Zygosporangium	21	Education in Malaysia
4	Bharatanatyam	22	Gene expression programming
5	Biology polymorphism	23	machu picchu peru
6	Cartoon guyver	24	Persepolis Iranian site

7	Chinese knot	25	Sea monster video
8	Frontal-striatal dysfunction	26	socio-economic impact study
9	Image processing	27	Blue Danube waltz, Strauss
10	Information retrieval	28	Children life in northwest china
11	Jet airways	29	Cultural life in Venezuela
12	pasupati temple	30	Flexible AC Transmission System
13	schizophrenia research	31	Network Security and Cryptography
14	Service-oriented architecture	32	Object oriented programming language
15	sternocleidomastoid muscle	33	Solar eclipse Egypt video
16	Travel Canada	34	Temple of Heaven Tiantan Echo Wall
17	Beijing Olympic torch	35	taiwan president election 2008
18	Buenos Aires travel	36	taj mahal mausoleum of mumtaz mahal

Table 1. Query list submitted to retrieval

5.1.2 Databases

We use oracle 9i for the database system. All the database tables used in the experiment are listed in Table 2.

ID	Database Table	Description
1	URL_his	Save the historical QoS data of each access of a web page
2	QoS	Save the statistical QoS data of a web page for retrieval purpose

Table 2. Database table used in the experiment

5.2 Experiment Design and Implementation

In the experiment, the meta search engine system and the internal combination algorithms are developed on J2EE version 1.5. BEA Weblogic Application Server version 8.2 is chosen as the application server and web server. The meta search engine is built to get the retrieval pages from three popular search engines and to combine those pages into a single ranked list using QoS factors.

According to the targets of the meta search engine, there are two models, QoS collection and evaluation model and the QoS re-ranking model, working together to complete the retrieval process. The QoS collection and evaluation model, called *QoS Watcher*, is a service which automatically runs to collect all the data of QoS attribute of those retrieved pages and save the evaluated QoS data into database tables. Given a query, Meta search engine will send the result list to QoS watcher to calculate the values of the QoS attributes, and then continue the re-ranking process with these QoS attributes. A simple architecture about the process of the QoS watcher Model is shown in Figure 10.

Another model is the QoS re-ranking Model. When users submit the query, meta search engine firstly combines the results from three search engines and send the combined result to QoS Watcher; in the mean time, meta search engine evokes the QoS analysis re-ranking procedure. The scores of the input pages from the first combination are calculated again according to the QoS values from the QoS table and a new ranked list is generated and returned to the user. This procedure is demonstrated in Figure 11.

Since users normally look at the first one or two pages from the search engine, and by default, there are 10 results in one page, we assume that the top 20 results are the most important pages for users and they would be viewed in most situations. Keeping this in mind, we only downloaded the top 20 pages for each query and calculated the QoS values for each of the page.

Figure 10. The process of QoS Watcher which collects the data of QoS attributes

In order to calculate the QoS value more accurately, we downloaded each page ten times and then we used the average value for each QoS attribute. The process is that, meta search engine first downloads the result lists from three popular search engines and then uses Condorcet-fuse rank aggregation method to generate a ranked list. This ranked list is then transferred to the QoS Watcher program. The QoS Watcher then automatically accesses those web pages in the input list, analyzes the HTTP Response messages and the content of the web pages, and then computes the values of QoS attributes for each page. Finally those values are saved into the local repository after calculation. It is worth noting that QoS watcher is like a service that can keep downloading the web pages in the background, therefore parallel computing and scalable distributed computing are possible.

After QoS Watcher collecting and analyzing those web pages, all the data of QoS attributes has been updated into the database tables. Another process is demonstrated in Figure 11, which is QoS re-ranking. The meta search engine will use the data directly from local database to re-rank the page lists so as to make the result list have a higher QoS value. By re-ranking only the top 20 pages, we believe that we wouldn't sacrifice the relevancy of the results too much and in the mean time we could improve QoS of the results. Altogether six combination methods and several use cases are tested, which are listed in Table 3.

Figure 11. The process of QoS reranking with QoS attributes

ID	Type of Combination Methods	Description
1	T1	QoS based re-ranking
2	T2	linear score combination with original rank
3	T3	Weighted Concorcet rank aggregation with QoS attributes
4	T4	Weighted Median rank aggregation with QoS attributes
5	T5	Weighted Borda rank aggregation with QoS attributes
6	T6	Genetic Algorithm based Weighted Partial Footrule Optimal rank aggregation with QoS attributes (PFOD)

Table 3. Type of combination methods used in the experiment

5.3 Evaluation Method

Since the main purpose of our proposed algorithms is to improve QoS of the search results, we use the top 10 QoS improvement to measure the effectiveness of the algorithm. The top 10 QoS improvement is defined as the sum of the improvement on the average QoS value of the top 10 results for all the QoS attributes based on user's preference. The formula is listed below:

$$\Delta QoS = \sum_{i \in QoS \text{ attributes}} W_i \times \left(\frac{Avg_{new-top10}(QoS_i) - Avg_{old-top10}(QoS_i)}{Avg_{old-top10}(QoS_i)} \right) \quad (5-1)$$

where i is the index of QoS attributes which between 1 and 5. QoS_i is the QoS value for i -th QoS attribute. ΔQoS represents the overall improvement on the average QoS value for a specific query, i is the index of QoS attributes, in the experiment there are five QoS attributes in total. W_i is the weight of user preference for a specific QoS attributes. Right now we use discrete integer value, 0 and 1 to present the user interest.

$Avg_{new-top10}(QoS_i)$ and $Avg_{old-top10}(QoS_i)$ are the average QoS values of a ranking list for i th QoS attribute. The formula to calculate these two values are shown below in Equation 5-2 and 5-3:

$$Avg_{\substack{j \in new-top10 \\ i \in QoS \text{ attributes}}} (QoS_i) = \frac{\sum_{j \in new-top10} (QoS_{ij})}{Size(Ranking_{new-top10})} \quad (5-2)$$

where i is the index of QoS attributes which between 1 and 5, j is the page index of new Top10 results, which is between 1 and 10. QoS_{ij} is the QoS value of j -th page for i -th QoS attribute. Similar to the formula (5-2) which represents the average QoS value of a

specific QoS attribute for the new top10 ranking, the average QoS value for the old top 10 is described in the equation (5-3) .

$$Avg_{\substack{j \in \text{old-top } 10 \\ i \in \text{QoS attributes}}} (QoS_i) = \frac{\sum_{j \in \text{old-top } 10} (QoS_{ij})}{Size(Ranking_{\text{old-top } 10})} \quad (5-3)$$

where i is the index of QoS attributes which between 1 and 5. j is the page index of new Top10 results, which is between 1 and 10. QoS_{ij} is the QoS value of j -th page for i -th QoS attribute. In these two formulas, it is worth noting that the size is not exactly the same as the size of the input ranking. When the QoS value of a specific QoS attribute for a particular page, QoS_{ij} is invalid data, e.g. response time is a negative value, in order to keep the data accurate, we didn't count those kinds of invalid values into average calculation, therefore the size of the ranking should be the size of the elements of the input ranking which has valid values.

Relevancy is another important performance measurement in IR system. There are two basic metrics to measure the relevancy of retrieved documents: recall and precision. *“Recall is the ratio of retrieved relevant documents to total relevant documents. Precision is the ratio of retrieved relevant documents to retrieved documents.”*[17]

$$R = \frac{|D_r|}{|D|} \quad (5-4)$$

where R means the value of recall, $|D_r|$ is the number of relevant documents in the retrieved result, $|D|$ is the number of relevant documents in the collection.

Similarly,

$$P = \frac{|D_r|}{|D_R|} \quad (5-5)$$

where P means the value of precision, $|D_r|$ is the number of relevant documents in the retrieved result, $|D_R|$ is the number of the documents in the retrieved result.

Both of the measurements can represent the document relevancy to some extent. However, they can't entirely represent the document relevancy because we cannot get the exact number of relevant documents in the collections from three popular search engines, and the value of precision $\frac{|D_r|}{|D_R|}$, might change for different $|D_R|$. Therefore, other measurement of relevancy was proposed. For example, P_Avg performance measurement is used in [11], which combines the precision and recall.

We also define another performance measurement for document relevancy in the experiment, $\Delta Top10$, which is the percentage of the pages of Top 10 re-ranked pages lying in the original Top 10 result. For example, assume that there are 100 pages retrieved from the meta search engine, since users normally only view the top 20 results, and the top 10 results from original result list have the similar relevancy or accuracy, the $\Delta Top10$ measurement is related to the Top 10 precision to some extent. Hereby, after QoS re-ranking, we measure how many Top 10 pages in the new ranked list also lie in the original Top 10 pages.

By the $\Delta Top10$ definition, we define a new measurement of performance: *Effectiveness*, which measures the QoS improvement with the $\Delta Top10$ relevancy. In this

measurement, we combine $\Delta Top10$ with the QoS performance. The formula is listed below:

$$\Delta Effectiveness = \frac{\sum_{i \in queries} \Delta QoS_i \times \Delta Top10_i}{|queries|} \quad (5-6)$$

where $\Delta Effectiveness$ means the average combined QoS improvement, i is the index for the query set, in the experiment there are 36 queries in total. ΔQoS_i is the QoS improvement of the Top 10 result for i -th query. $\Delta Top10_i$ is the precision measurement defined above. $|queries|$ is the size of the query set in the experiment.

5.4 Combination Methods and Use Cases

In the experiment, we tested the proposed six combination methods which are listed in Table 5 – 1 (please refer to chapter 5.2). The weight of the QoS attribute could be either 0 or 1 as we explained before. If we consider all the possible combinations of five weights, there are altogether 31 cases of them, without considering the all zero case. Therefore, for each query, each method, there are 31 groups of results.

It is worth noting that our T6 combination method is using Genetic Algorithm (GA) to find an aggregated ranking which has a minimum footrule distance with the given ranked lists obtained from three search engines for a given query. We took 3000 chromosomes in the gene pool for the GA. Each chromosome in the gene pool is a individual which represents a permutation of a full list. Crossover between individuals is implemented by multiplication of two permutations. The crossover rate is 0.75 and mutation is at a rate of 0.015. After 20 generations, we generated the final result.

5.5 Results and Analyses

In the Figure 12 to Figure 19, T1 to T6 are those combination methods which are tested in our experiment (Please refer to the Table 3). T1 means the QoS linear combination. It uses linear combination to calculate the final score for the pages in the rankings so as to get the new rank order. T2 is similar to T1 whereas it also includes the normalized position data on top of the QoS scores, because in this way it could retain more original top 10 results so as to keep the level of accuracy attained by the original search engines. All the rest of the combination methods include the original position data. T3 implements Concorcet-fuse [18] to combine the results. T4 applies Median aggregation method and T5 uses Borda-fuse [17] rank aggregation. T6 is using Genetic Algorithm to find the optimal ranking which has the minimum footrule distance with the original ranking and the QoS rankings. All those combination methods are based on user preferences. Therefore, a weight for each QoS attribute is defined in the configuration file and when it is set as zero, it means that the corresponding QoS attribute is not included in the ranking. Q1 to Q36 refer to queries in our query set (please refer to the Table 1), and Average means the value averaged on all 36 queries.

Figure 12. ΔQoS for each QoS attribute when only the weight of that attribute is 1

Figure 12 shows ΔQoS on each QoS attribute for all queries when the weight of that particular QoS attribute is set to 1 and all other weights are set to 0. And, this figure is generated from T1. We will compare the performances of 6 combination methods in the latter part of this chapter. From this figure, we can see that the improvement on reliability is very little, and most of queries have negligible values. The reason is that the original reliability is very high already, usually above 90%. On the other hand, since the queries are composed by only English words, most of the web sites are from English-speaking countries that usually have well-developed network infrastructure. In other words, if the web site comes from a developing country, the connection to this web site might not be so stable, thus reliability will be an important attribute. The improvement on the other four QoS attributes are very good, on average, both response time and file size have over 60% increase on QoS values, media richness could achieve almost 80% increase, and freshness is the lowest among the four, while its improvement is also over 40%.

Figure 13. Average ΔQoS for each QoS attribute when the weight of this attribute is 1

In Figure 12, only one QoS attribute is considered at a time. However sometimes users may want to consider several QoS attributes. The weight of the user preference could control whether a QoS attribute will be included in the calculation. For example, if the weight vector for all 5 attributes is (1 0 0 0 1), it means that the reliability and media richness factors are included in the QoS based ranking procedure. For each attribute, considering all possible weight combinations (e.g. for reliability, it would be (1 * * * *)), * means it is either 0 or 1), there are 16 of them. Figure 5 - 4 shows the average ΔQoS for each QoS attribute when the weight of that particular QoS is set to 1 and all 16 weight combinations are considered and it only shows the average result for all the 36 queries. We could see that from the results, although the improvement on each QoS attribute is less than that in Figure 12, the pattern is quite consistent and the degree of increase is not affected much by including more QoS attributes.

Figure 14. Comparisons of the 6 combination methods on average Δ QoS

Figure 14 shows the comparison of the six combination methods. Besides the improvement on five individual QoS attributes, the overall QoS improvement is also displayed. The value in Figure 14 is the average value of each attribute for all the queries and all the user preference. We could see that overall speaking, T1 is the best performing method regarding the improvement of overall QoS in the top 10 search results, and it is also consistently the best for 4 attributes: response time, freshness, file size and media richness. T2 is the second best one, but it is 22.6% worse than T1 and 17% better than T4. T4 is the third best one and it is also the best one among all the rank aggregation algorithms. It is only 0.5% better than T6, but, since T6 uses GA for computing, the computing time is much more than T4. In addition, T4 is 18% and 11% better than T3 and T5 respectively. For four QoS attributes – response time, freshness, file size and media richness, T4 performs better than T3 and T5.

When we look at more detailed results for each query, there are 20 out of 36 queries in which T4 performs better than T6 regarding the overall QoS, whereas T1 consistently performs the best for all the queries.

Figure 15. Comparison of 6 combination methods on average ΔQoS considering 3 QoS attributes only

As we mentioned before, one of our target user groups in mind is the group of mobile users and users with slow Internet connection. For them, response time, file size and media richness are three most important QoS attributes for those users. We didn't consider reliability because it is very good already. In Figure 15, we show the comparison of six methods when only three attributes are considered, i.e. the value in Figure 15 is the average value of these three attributes for all the 36 queries with the weight vector (0 1 0 1 1).

If we only look at the overall QoS, T1 is still the best performing method, but it is only 4% better than T4, and T4 is the second best one, which is 6.7% better than the third best one T5. The other three combination methods, T2, T3 and T6, are slightly worse than T5. Here we are interested in the T4 Median rank aggregation. Because in the previous figure 14 when considering all the QoS attributes, T4 is the third best one which is only 16% worse than T2, however, if we only consider three QoS attributes which are important to mobile users, the result is quite different. T4 median aggregation gets better

performance than previous result shown in figure 14. This observation implies that for mobile user or those users with band limitation, T1 is still good in performance but T4 might be another solution if we apply other method to optimize the Median rank aggregation. Moreover, T4 is also the best one in rank aggregation methods. It is very important for the meta search engine because it can only get the ranks from other search engines.

Figure 16. Comparison of T1-T6 combination methods on average ΔQoS

Figure 16 shows the comparison of the average QoS improvement of six combination methods for all queries. Besides the average performance of ΔQoS improvement for all the cases of user preference on all the 36 queries, the overall average ΔQoS is also displayed. In Figure 14, we could see that T1 is always the best performing method regarding the improvement of overall QoS in the top 10 search results. T2 is continuously the second best one, but it is 22.6% worse than T1 and 17% better than T4. T4 is the third best one and it is also the best one among all the rank aggregation

algorithms. It is only 0.5% better than T6, but, because the same reason listed in previous Figure 15, we believe T4 outperforms T6. In addition, T4 is 18% and 11% better than T3 and T5 respectively. And for four QoS attributes – response time, freshness, files size and media richness, T4 also performs better than T3 and T5.

Figure 17. Comparison of T1-T6 combination methods on average delta-Top10

Figure 17 shows another comparison of the average $\Delta Top10$ improvements of the six combination methods for all queries. Like Figure 16, the average $\Delta Top10$ is also displayed. All the values in Figure 17 are the average $\Delta Top10$ for all the cases of user preference on all the 36 queries. The figure demonstrates that on average, T5 is the best method that keeps the most 77.5% original Top 10 documents in Top 10 position in the new ranking. If we take the original ranking as the baseline, we could say that T5 also has the highest Top 10 precision. T4 is slightly worse than T5, only 1.8% less. And T2 and T3 is 1% and 2% worse than T4 respectively. In the result, T1 is the worst one, which is 40.9% worse than the best one T5. The reason why T1 QoS re-ranking method is worst

is that T1 only considers the values of the five QoS attributes, excluding the original ranking, but on the contrary, the other combination methods include the original ranking, thus keep the Top10 precision better. For example, T2 gets 75% $\Delta Top10$ performance and T3 keeps 74.7% $\Delta Top10$. However, our target is to help users have a better experience not only in relevancy but also in other QoS attributes. Therefore, in Figure 18, we will show an overall measurement *Effectiveness* which integrates relevancy and QoS attributes.

Figure 18. Comparison of T1-T6 combination methods on effectiveness

Figure 18 shows the effectiveness comparison of the six combination methods for the cases of user preference on all the 36 queries. And, the average effectiveness over 36 queries is also displayed. Effectiveness is to combine the QoS improvement and $\Delta Top10$ of the re-ranked result.

We could see from the result that on average, T1 is not the best method and that is 11.7% less than T2. T4 is the third best one which is 3.7% less than T1. In the rank

aggregation methods, T3, T5, and T6 are 20.4%, 9.1% and 27.1% worse than T4 respectively. And when we also look at more detailed results for each query, there are 20 out of 36 queries in which T4 outperforms T1 regarding the overall effectiveness.

Figure 19. Comparison of T1-T6 combination methods on effectiveness considering 3 QoS attributes only

Figure 19 shows the effectiveness comparison of the six combination methods for all the 36 queries when only considering 3 QoS attributes. And, the average effectiveness is also displayed. Because of the consideration of mobile users and users with slow Internet connection, we also test the effectiveness over the three important QoS attributes: response time, file size and media richness. We could see from the result that on average, T2 is the best method which is similar in Figure 18. T4 is the second best one which is 5.7% better than T1. In the rank aggregation methods, T3, T5, and T6 are 9.1%, 2.9% and 31.4% worse than T4 respectively. And when we also look at more detailed results for each query, there are 9 out of 36 queries in which T4 outperforms T1 regarding the overall effectiveness. This observation implies that for mobile user or those

users with band limitation, T4 might be a good solution by also considering the result in Figure 15.

5.5 Discussion

Usually precision is an important measurement to decide the effectiveness of a search engine. In this study, we assume users normally view the top 20 results, since our algorithm only changes their order, it doesn't affect the top-20 precision. We also assume that web pages in top 20 positions have similar level of accuracy. Based on these assumptions, we only measure the improvement on QoS without considering the possible change of the precision value. In the future, we could further study the impact of our QoS based algorithm on the precision value. As shown in the experiment, the QoS based re-ranking algorithm can achieve the best performance on QoS improvement. However, it totally ignores the original rank, which might affect its precision value. So the other rank aggregation or score combination methods might be a better choice to have both improved QoS and a good precision.

Another observation is that Median rank aggregation always outperforms other rank aggregation methods in either QoS improvement or Effectiveness measurement which combines the QoS improvement and relevancy. Even though the average precision cannot be over 70% [49], we still believe that Median rank aggregation will be a good solution for meta search engine. In addition, it works well for those users with slow Internet connection or band limitation. Therefore, we believe that Median rank aggregation method has a promising future if we apply other algorithm to optimize it.

In this chapter, we analyze the performance of six combination algorithms in different view point. From the result, we see that QoS-based re-ranking algorithm can improve the QoS performance well and Median rank aggregation method works well when considering the QoS and relevancy together.

CHAPTER 6

CONCLUSIONS

In this thesis, we defined five QoS attributes that are mainly performance related and are especially important to Internet users with slow connection or mobile users. By including them in the ranking procedure to re-rank the top positioned pages, users could have chance to see pages with a high delivery quality earlier than pages with a low quality, and as a consequence, it could improve users' searching experiences. The experiment showed the promising results. From the results, each combination algorithm has at least 40% QoS improvement. Comparing the six algorithms of combining QoS attributes, QoS re-ranking algorithm can always achieve a big improvement not only for a single QoS attribute but also for multiple QoS attributes.

However, for mobile users or Internet users with slow connection, Median rank aggregation method is better than QoS re-ranking algorithm. And it is very promising and effective especially for meta search engines which can only get the rank information. In all the cases, Median rank aggregation method is always the best one among all rank aggregation methods. In addition, when considering the *effectiveness* results in Figure 19 and the QoS improvement in Figure 15 together, although Median rank aggregation method is not the best one in these two figures, it is always the second best one and the best one is not the same. It implies that for mobile users, Median rank aggregation

method is a good solution. If we can somehow optimize the Median rank aggregation method, the performance might be further improved.

Contributions of Thesis

The main contributions of this thesis are:

- Demonstrate the effectiveness of web search with the QoS attributes
- Define some important QoS attributes
- Create a QoS Watcher program that collects and computes QoS information
- Create a QoS Analysis program that analyzes and re-ranks web pages by QoS attributes
- Create a meta search engine that using QoS Analysis as the re-ranking model
- Compare the performance of six combination methods using QoS attributes

Future Work

There are several directions of works we would like to work in the future such as the ones we listed in the previous chapter.

Another direction we would like to work on is that we could test our algorithm on a wireless device such as a cell phone or PDA, and on Internet connections with different speed. We would also like to implement a user interface so that users can choose their own preferences on QoS attributes, and a user study on their satisfaction level can be done afterwards. Further study can also include studying the impact of the QoS based algorithm on the precision value. It is worth creating a traditional search engine to apply the QoS attributes in the ranking algorithm to test whether different rank aggregation or score combination methods can improve both QoS and the precision value.

In addition, reliability is an important factor to be further studied. We could test some queries which are written in the languages of some developing countries.

REFERENCES

- [1] V. Bush, "As We May Think", ACM SIGPC Notes, 1(4), April 1979.
- [2] Google : <http://www.google.com>
- [3] Yahoo : <http://www.yahoo.com>
- [4] MSN : <http://www.msn.com>
- [5] S. Brin and L. Page, "The Anatomy of a Large Scale Hypertextual Web Search Engine", In *Proceedings of the 7th International Conference on WWW*, pp. 107-117, 1998.
- [6] <http://searchenginewatch.com/showPage.html?page=3628309>
- [7] R. Baeza-Yates and B. Ribeiro-Neto, "Modern Information Retrieval", Addison-Wesley Publishing Company, 1999.
- [8] J. Zobel, and A. Moffat, "Inverted Files for Text Search Engines", *ACM Computing Surveys*, 38(2):1-56, July 2006.
- [9] G. Salton. "Associative Document Retrieval Techniques Using Bibliographic Information". *Journal of the ACM*, 10(4): 440-457, October 1963.
- [10] S. E. Robertson and K. Sparck Jones. "Relevance Weighting of Search Terms." *Journal of the American Society for Information Sciences*, 27(3): 129-146, 1976.
- [11] W. Fan, M.D. Gordon P. Pathak. "Genetic Programming-Based Discovery of Ranking Functions for Effective Web Search". *Journal of Management Information Systems*, 21(4):37-56, 2005.
- [12] A. Arasu, J.Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan, "Searching the Web", *ACM Transactions on Internet Technology*, 1(1):2-43, 2001.

- [13] J. Kleinberg, "Authoritative Sources in Hyperlinked Environment", in *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithm*, pp. 668-667, 1998.
- [14] <http://www.searchengineshowdown.com/features/directhit/review.html>
- [15] HTTP: Hypertext Transfer Protocol Overview, <http://www.w3.org/Protocols>
- [16] S. P. Ran, "A Model for Web Services Discovery with QoS", *ACM SIGecom Exchanges*, 4(1): 1-10, 2003.
- [17] J. A. Aslam, and M. Montague, "Models for Metasearch", in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 276-284, 2001.
- [18] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank Aggregation Methods for the Web", in *Proceedings of the 10th World Wide Web Conference*, pp. 613-622, 2001.
- [19] M. Montague, and J. A. Aslam, "Condorcet Fusion for Improved Retrieval", in *Proceedings of the 11th International Conference on Information and knowledge management*, pp. 538-548, 2002.
- [20] R. Fagin, R. Kumar, D. Sivakumar, "Efficient Similarity Search and Classification via Rank Aggregation", in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pp. 301-312, 2003.
- [21] S. M. M. Beg, "Parallel Rank Aggregation for the World Wide Web", in *Proceedings of International Conference on Intelligent Sensing and Information Processing*, pp. 385 – 390, 2004.

- [22] B. T. Bartell, G. W. Cottrell, and R. K. Below, "Automatic Combination of Multiple Ranked Retrieval Systems", in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 173-181,1994.
- [23] E. A. Fox and J. A. Shaw, "Combination of Multiple Searches", in *Proceedings of the 2nd Text Retrieval Conference*, pp. 243-252,1994.
- [24] J. H. Lee, "Analyses of Multiple Evidence Combination", in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267-276,1997.
- [25] J. H. Lee. "Combining Multiple Evidence from Different Properties of Weighting Schemes", in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 180-188,1995.
- [26] Y. Liu, T. Liu, T. Qin, Z. Ma, H. Li,"Supervised Rank Aggregation", in *Proceedings of the 16th International Conference on World Wide Web*, pp. 481-490, 2007.
- [27] R. Manmatha, T. Rath and F. Feng. "Modeling Score Distributions for Combining the Outputs of Search Engines". In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267-275, 2001.
- [28] C. C. Vogt. "How Much More is Better? Characterizing the Effects of Adding More IR Systems to a Combination", in *Proceedings of International Conference on Content-Based Multimedia Information Access (RIAO)*, pp. 457-475, 2000.

- [29] R. Manmatha, and H. Sever. "A Formal Approach to Score Normalization for Metasearch", in *Proceedings of HLT*, pp. 88-93, 2002.
- [30] R. Fagin, R. Kumar, and D. Sivakumar. "Comparing Top k Lists", in *Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms*, 17(1):134-160, 2003.
- [31] E. A. Fox and J. A. Shaw, "Combination of Multiple Searches", in *Proceedings of the 2nd Text Retrieval Conference*, pp. 243-252, 1994.
- [32] J. H. Lee, "Analyses of Multiple Evidence Combination", in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267-276, 1997.
- [33] W. B. Croft, "Combining Approaches to Information Retrieval", in *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*, W. B. Croft, Ed. Boston, USA: Kluwer Academic Publishers, pp.1-36, 2000.
- [34] Z. Bar-Yossef, and M. Gurevich, "Efficient Search Engine Measurements", in *Proceedings of the 16th international conference on World Wide Web*, pp. 401-410, 2007.
- [35] TREC: Text Retrieval Conference. <http://trec.nist.gov/>
- [36] Y. Yao, X. Chen, and S. Zhu, "Rank Aggregation Algorithms Based on Voting Model for Meta-search", in *Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pp. 1 – 4, 2006.

- [37] T. Avrahami, L. Yau, S. Luo, and J. Callan. "The Fedlemur Project: Federated Search in the Real World", *Journal of the American Society for Information Science and Technology*, 57(3):347-358, 2006.
- [38] A. L. Calve, and J. Savoy, "Database Merging Strategy Based on Logistic Regression", *Journal of Information Processing and Management*, 36(3):341-359, 2000.
- [39] M. N. Rocha, G. R. Mateus, and S. L. da Silva, "QoS and Simulation Models in Mobile Communication Networks", in *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pp. 119-122, 2000.
- [40] R. Canonico, M. D'Arienzo, B. Fadini, S. P. Romano, and G. Ventre, "On the Introduction of Quality of Service Awareness in Legacy Distributed Applications", in *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pp.659-664, 2002.
- [41] X. Xiong, N. Uchida, K. Hashimoto, and Y. Shibata, "QoS Control for Continuous Media over Heterogeneous Environment by Wired and Wireless Network", in *Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)*, pp. 118, 2006.
- [42] V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian, "A Quality of Service Management Framework Based on User Expectations", in *Proceedings of the 1st SOC Conference*, pp. 104-114, 2003.
- [43] S. Kapidakis, S. Terzis, and J. Sairamesh, "A Management Architecture for Measuring and Monitoring the Behavior of Digital Libraries", in *Proceedings of the*

2nd European Conference on Research and Advanced Technology for Digital Libraries, pp. 95-114, 1998.

- [44] P. Barker, "Providing the X.500 Directory User with QoS Information", *ACM SIGCOMM Computer Communication Review*, 24(3):28-37, 1996.
- [45] <http://www.useit.com/alertbox/20010513.html>
- [46] <http://arnoldit.com/wordpress/2008/04/11/the-importance-of-being-first/>
- [47] G. Cooperman, and X. Ma, "Overcoming the Memory Wall in Symbolic Algebra: a Faster Permutation Multiplication", *ACM SIGSAM Bulletin*, 36(4): 1-4, December 2002.
- [48] http://research.microsoft.com/workshops/ira2008/ira2008_paper3.pdf
- [49] S. K. Bhandari, and B. D. Davison, "Leveraging Search Engine Results for Query Classification", Technical Report LU-CSE-07-013, Dept. of Computer Science and Engineering, Lehigh University, 2007.