THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

Search Engine Selection with Hierarchical Categorization

by Shiu Koon Hang

A thesis submitted in partial fulfillment of the requirements for the Degree of

Master of Philosophy

Department of Computing

The Hong Kong Polytechnic University

February 2003

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Signature : _____

Name of Student : Shiu Koon Hang

Abstract

Search engines are very successful in locating resources on the Internet. However they cover only small and different parts of the vast Internet. While using metasearching to simultaneously engage multiple general search engines can increase the coverage, users encounter difficulties in finding target information in the large set of results returned. In addition, a large amount of valuable resources are only available behind specialty search engines or databases. The problem of covering the Internet has been replaced by the problem of selecting suitable search engines for a given query. This thesis focuses on developing an algorithm to construct a hierarchical category of specialty search engines automatically to assist in solving the coverage problem and the search engine selection problem.

We believe that metasearching specialty search engines can effectively discover large amount of hidden Web resources. In particular, we proposed to categorize specialty search engines automatically by sending probe queries to them, fetch and analyze the returned result documents. By determining the relevancies between the returned documents and search categories, specialty search engines can be associated with nodes in a hierarchical search engine category directory for metasearching. By utilizing the category, search engines that have a high possibility of relevant information and resources can be easily selected by a metasearch engine. In this thesis, we present the concepts, designs, implementation details, and validation of the proposed categorization algorithm and a metasearch engine prototype, which seeks to demonstrate such a search engine category can be beneficial in finding essential Web resources.

Acknowledgement

I would like to express my sincere gratitude to all those who have assisted me, and made this thesis possible.

I am deeply indebted to my supervisor, Dr. Stephen C. F. Chan, for his invaluable guidance, helpful advice and constructive criticisms. Without his understanding and unrestricted support, this thesis cannot be completed.

I would like to extend my thanks to Dr. Korris F. L. Chung for his valuable suggestions and comments on my research. Thanks also go to Dr. Vincent T. Y. Ng. He taught me many courses and supervised my final year project 3 years ago. In my research career, I benefited greatly from the countless things I learnt from him.

I would like to especially thank Ms Lau Chui Yi, for her love, encouragement and limitless support.

Table of Contents

List of Figures

Chapter 1

Introduction

As the size of the Internet is huge, search engines become the most useful tools for the retrieval of Web documents on the Internet. However, studies showed that most of the search engines only cover small and different parts of the whole Internet. As a result, metasearching has been introduced and widely investigated. While metasearching can increase the coverage, users encounter difficulties in finding target information in the large set of results returned. Because the results are returned by different search engines, and each of them may cover a different part on the Internet, the problem of covering the Internet has been replaced by the problem of selecting suitable search engines for a given query. In this research, we developed an algorithm to build a hierarchical category of search engines to assist in solving the search engine selection problem.

In contrast, we believe providing metasearching for accessing specialty resources on the Web can help novice users who are new to finding specialty resources for a particular area. Since novice users do not know where to locate the target specialty information, they commonly have to start with a general search engine and try to discover the specialty resources from the search results. Therefore by providing metasearching for specialty search engines, users can access the target information directly and the efforts needed for the discovering processes can be significantly saved. Moreover, for experienced users who already know where to locate the specialty resources they want, providing metasearching also can help them access numbers of sources simultaneously and search results can be group into categories or

other means that can provide more easily interpretation.

1.1 Motivations

The last years have witnessed the great success of Internet Search Engines. As the most useful and high profile resources of the rapidly expanding World Wide Web, search engines help users to locate information over the Web efficiently. To locate documents and resources on the Internet without knowing their URLs, a search engine is an incomparable tool for users. Examples of successful search engines include Google [Google03], Yahoo! [Yahoo03], Lycos [Lycos03], AltaVista [AltaVista03] and Excite [Excite03]. Excluding Yahoo!, which have human-edited Web directories, all of them are large in scale and based on using software robots to "crawl" and index Web documents. Although this kind of search engines keep on improving their search technology in different ways, their performance is far from completely satisfactory.

There are two major problems with general search engines: the coverage of the Web is not sufficient, and the search results are not accurate. Research shows that major search engines only index small fractions of the total Web pages, no single general search engine indexes more than about 16% of the Web [Lawrence99]. One of the reasons that make it difficult for general search engines to increase their coverage is that many documents available on the Internet are not "crawlable" to their software robots. As robots traverse from one document to another by following the hypertext links in the document, they index the Web by recording every hyperlink in every page that they traversed. Consequently, robots cannot index documents that are encapsulated by a search engine interface on a Web site and generated dynamically by Web servers. In this research, we use the term *specialty search engines* [Dreilinger97] to represent such kind of search engine interfaces that manage

a huge number of "uncrawlable" Web pages on the Web. As the technologies for dynamically serving Web documents improve continuously, the number of Web site managing documents in such a way increases considerably, and therefore it is difficult for the coverage of the general search engine to increase significantly. A study showed that resources that are not indexed by general search engines are hundreds of times greater than those indexed resources [Bergman01]. It has been estimated that there are nearly 550 billion individual documents in the size of 7,500 terabytes which have not been indexed.

To solve the coverage problem, researchers have developed *metasearch engines*, which can automatically and simultaneously query several general search engines, merge the search results and present them in a uniform format. Examples of metasearch engine are MetaCrawler [Selberg97], SavvySearch [Dreilinger97], Ixquick [Ixquick03], Profusion [Gauch96, Profusion03] and Dogpile [Dogpile03]. Although metasearch engines combine the coverage of different general search engines, the coverage problem still exists because the coverage of any single search engine is too small. Moreover, as metasearch engines combine search results from a number of search engines, the problem of inaccurate results may actually get worse because of the increased size of the combined result list.

As a result, to explore the "uncrawlable" resources managed by specialty search engines becomes a standing challenge. One solution is again metasearching. While general metasearch engines exploit general search engines, a number of metasearch engines were developed for exploiting only specialty search engines. For instance, MetaXChem [MetaXChem03] is a metasearch engine that metasearch numbers of chemistry-related specialty search engines. Other examples of such kind of metasearch engines include LawBot[Lawbot00], Ithaki [Ithaki03], Family

Friendly Search [Family03] and SoftCrawler [SoftCrawler03]. Nevertheless, compared to the amount of all the resources on the Web, the resources discovered by each of the metasearch engines above are limited and specialized. It is because each metasearch engine only exploits a small set of specialty search engines that are inside the same domain.

Therefore, a metasearch engine that can exploit specialty search engines across different domains will be a possible solution to access the "uncrawlable" resources on the Web. However, we currently do not observe any metasearch engine having such an ability. We believe the reason why there is no existing metasearch engine which can successfully metasearch a large set of specialty search engines under different domains is that it is too difficult to filter out suitable engines for a given query. For example, if a user wants to search for some academic articles about "Information Retrieval", using a metasearch engine to access different domains of specialty search engines is not efficient. Alternatively, if such metasearch engine is able to select only specialty search engines that are under the academic or library domain, it will greatly improve the search quality. In other words, metasearch engines are now facing the *search engine selection* problem, and once the problem can be solved, the performance of metasearch engines can improve significantly.

1.2 Design Goals

We believe that developing a hierarchical search engine category which groups different specialty search engines is a possible solution to deal with the search engine selection problem. By using a hierarchical category to organize different specialty search engines, we believe that only search engines which contain the target relevant information and resources can easily be selected by metasearch engine. Therefore, the excessive cost of querying unsuitable search engines can be reduced and

metasearch engine can provide access to more valuable data on the Web without degrading the quality of search results.

However, to categorize large number of specialty search engines need large amount of customization efforts. Thus each of the existing specialized metasearch engines only focus on a specific domain and metasearching a small number of specialty search engines. In this research, we proposed, designed and implemented a categorization algorithm that aim to provide systematic methods to categorize specialty search engines. By using only a small amount of customization effort, a given specialty search engine can be categorized by the proposed algorithm automatically. Moreover, we also developed an experimental metasearch engine that exploits the result search engine category, which seeks to demonstrate how such a search engine category can cope with the search engine selection problem.

1.3 Contributions

In this thesis, we analyzed the problems posed by the above challenges. We also designed, implemented and evaluated solutions that resulted in significant improvement for accessing hidden Web resources in a metasearching environment. Because there is tremendous heterogeneity and diversity in the contents of specialty search engines, and because there is enormous variation in finding relevant search engines for a given query in the metasearching environment, there is no single static solution that can address all the observed problems. We recognized this, and developed an algorithm that can automatically categorize specialty search engines into a hierarchical category to help solve these problems. In addition, a search engine selection algorithm was developed to select relevant search engines from the category, and a metasearch engine that implemented the selection algorithm was also developed.

1.4 Structure of Thesis

The rest of the thesis is organized as follows. In Chapter 2, we discuss the related works in the area of metasearching environment, and describe related work in selection of databases, servers and collections.

Chapter 3 to Chapter 6 form the core of this thesis. Chapter 3 describes the categorization algorithm designed to categorize specialty search engines automatically into a hierarchical category structure. Chapter 4 discusses the experiments done for validating the categorization algorithm. Chapter 5 presents a metasearch engine that implements a search engine selection algorithm, exploiting the search engine category structure, as a case example to show how metasearch engines can benefit from such category. We discuss the design and implementation of the metasearch engine and evaluate the search performance.

Finally, in Chapter 6, we present a summary of our work and contributions. We conclude this thesis by outlining some directions for future research.

Chapter 2

Background and Related Work

As traditional general search engines cannot provide satisfactory performance for locating resources in the fast expanding World Wide Web, metasearching mechanism has been introduced and widely investigated. The major standing challenge for the metasearching environment is how to select suitable search engines for metasearching when a particular query is given. Many efforts have been made by researchers on improving different collection, database and server selection algorithms. As those works motivate our research to tackle the *search engine selection* problem, a clear understanding of past work will help us better understand and gain perception of the complexity of the problem.

2.1 Specialty Search Engine

While general search engines are large-scale, robot-based and focused on indexing the Web as much as possible, *specialty search engines* [Dreilinger97] are human-edited Web directories, localized search engines and other search engines only focusing on specific topics. For example, Scirus [Scirus02] concentrates on searching scientific information on the Web; search engines in the RFC Editor Homepage [RfcEdit02] and Manufacturing.net [Manuf02] are only capable of finding documents from their localized databases. Search results returned by them are high-quality since they only focus on a specific set of search space with domain-specific knowledge, heuristics and technologies applied to the search algorithms.

Specialty search engines provide search interface to access documents that cannot be accessed directly through hyperlinks. The documents can only be accessed by dynamically issuing queries to the search interface, and some of them may contain dynamically generated contents. Traditional general search engines cannot index documents covered by specialty search engines because they only use software robots to traverse the Web by following the hypertext links in static documents. Moreover, querying specialty search engines may be the only way to retrieve fresh documents in some domains (e.g. News), since the documents are usually not indexed by general search engine because they are updated frequently, or they are too new. A recent study [Bergman01] highlighted the importance of the documents covered by specialty search engines, believing the size of the covered Web is approximate 550 times greater than those covered.

In this research, we concentrate on discovering the resources indexed by the specialty search engines used for searching HTML Web documents only; those search engines which provide search service for other types of resources like images are excluded. In addition, all the documents returned from the search engines should logically be created, maintained and owned by their originated Website. That is, search engines are also excluded if their search results are scattered over other Websites, since we believe such kind of results is "crawlable", and can be covered by different general search engines.

2.2 Metasearch engine

To increase the coverage of the Internet for searching, metasearch engines are developed which can automatically and simultaneously query several general search engines, construe the search results and present them in a uniform format. Based on the architecture of metasearch engine, much has been done by researchers in

improving the search quality of metasearch engine in different ways.

2.2.1 Metasearch Engine Architecture

Though different metasearch engine have used different approaches in improving the search quality, they have similar architectures. Figure 2.2.1a illustrate the major components generally used by a metasearch engine:

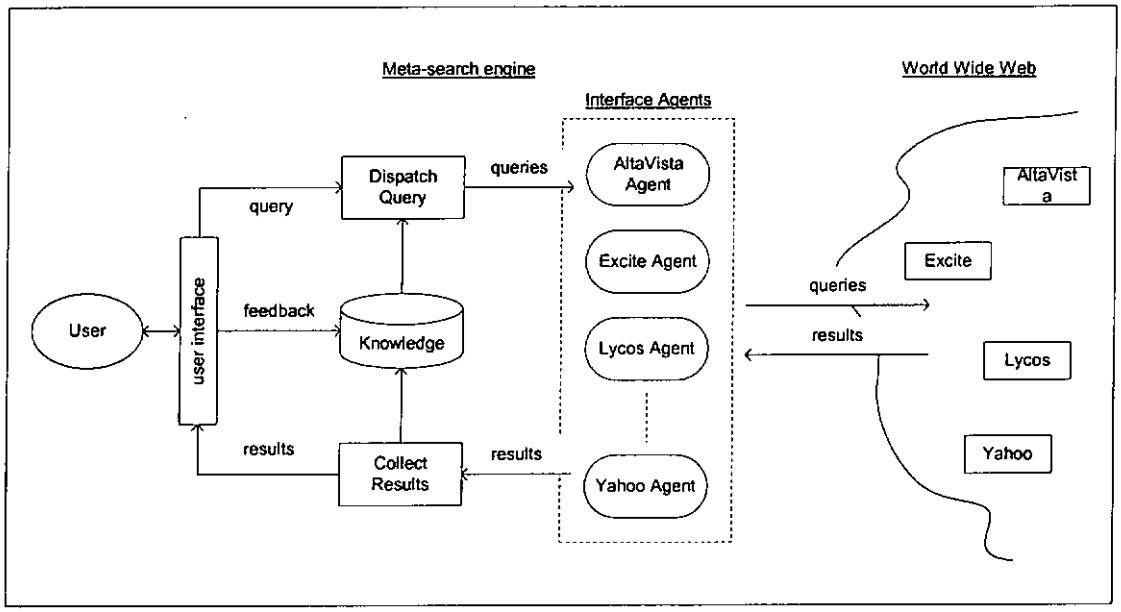

Figure 2.2.1a Architecture of metasearch engine

User Interface: the user interface of the metasearch engine lets the user issue queries in a uniform format. A user does not need to know how the other search engines work. The interface is also responsible for displaying the search results collected from different search engines.

Query Dispatching: when a query is received from the user, the metasearch engine decides which search engines should be used for that particular query. After deciding, the query is sent to those search engines simultaneously through the corresponding interface agents.

Interface Agents: Each agent is responsible for controlling the interactions and communications with a particular search engine. It converts the user's query into a format that can be understood by the interface of the search engine. After sending the query to the search engine, the agent will collect the returned search results and extract target information from them for later processes. Therefore, we can say that each interface agent is tailor-made for a particular search engine.

Results Collection: when all the results are received from the interface agents, the metasearch engine will integrate them all into its presentation format. Duplicated or broken links are usually removed. The integration process here is one of the key challenges in the metasearch engine. Since each search engine uses an independent search model, their ranking mechanisms are generally not comparable with each other. How the search results are combined into a rational list influences the overall performance of a metasearch engine.

Knowledge Base: as shown in the diagram, the knowledge can be used to improve the search quality of the metasearch engine. For example, when a user is assessing a result set, his behavior can be captured as useful information. This information tells the metasearch engine which results are selected by the user, as well as which search engines have returned useful results. The dispatch process can then make use of such knowledge when deciding which search engines are suitable for a given query. However, a number of issues have to be considered when such a mechanism is used. For example, computation and communication cost, user's privacy and, most importantly, developing appropriate algorithms. Currently, most existing metasearch engines do not have this capability.

2.2.2 Research Focuses

Similar to other general search engines, metasearch engines accept queries in the form of a set of keywords. Considering the task to search information from enormous number of documents, the information that can be obtained from the query itself is extremely limited. Since forcing users to specify their information needs more clearly is not practical, improving the search system is the only way to improve the performance of metasearch engines. Major research issues for improving metasearch engine can be classified into the following three areas:

Query Modification. As each search query submitted by users usually contains a few keywords only, it is hard for search engines to understand the information needs of the users. As a result, some researchers proposed to modify or expand the given query by different methods before performing the searching process. Modifications include changing, removing and adding extra terms. The objective is to make the modified query more precise and representative of the original query. A simple example is a user submitting a query "*Java*". The metasearch system may modify the query to "*Java coffee beans*" or "*Java programming*", according to the user's information need, which should be obtained before the modifications. Examples of related works in query modification can be found at [Mitra98], [Budzik99] [Agichtein01] and [Glover01].

Source Selection. If a metasearch engine is able to select only relevant search engines for searching, the unnecessary costs for querying irrelevant search engines can be reduced, and the search results returned are all of high quality. However, it is difficult to comprehend the exact information needs of the users based on few keywords in the query. As a result, researchers continue developing different selection methods aimed at improving the selection performance as much as possible.

As this research focuses on the selection problem, we will review another area of metasearch engine briefly, and then we will review related works in the selection area in more detail.

Collection Fusion. After collecting the search results from different search engines, metasearch engines try to merge the results to a single result list with the goal that the most relevant documents will be placed on the top of the list. It is very difficult to achieve a good fusion since there are heterogeneities between different search engines. Although methods, employed by metasearch engines such as MetaCrawler [Selberg97] that fetch all the result documents for fusion analysis can produce a high-quality merged list, the cost of downloading all the documents is very expensive. ProFusion [Gauch96] used a weighted score merging algorithm that based on the relevance score returned by search engines. When merging the results from different search engines, the factors of the accuracy of the search engines are estimated to improve the quality of the merged results. Voorhees et al. [Voorhees95] examined two fusion methods that based on the performance of past queries. While the number of results returned to every query is equal, the number of results retrieved from each collection is different according to the performance of past queries. Other less expensive fusion methods can also be found in [Gravano97], [Lawrence98] and [Yu01]. As our research focuses on the search engine categorization and selection, only simple fusion method has been used for the implementation of our metasearch engine. We believe in case a fusion method is needed, for example, to increase the quality of the search results, we can simply exploit one of the existing algorithms.

2.3 Selection Methods

Many approaches for the selection of databases, servers and collections have been

proposed in the past. Representative systems that are similar in the basis of their methodology are grouped together and summarized in this section.

2.3.1 Based on Theoretical Model

As the pioneer for searching distributed collections using the probabilistic model, Callan et al. [Callan95] introduced the CORI algorithm for providing collection selection based on the inference network model of information retrieval. The algorithm was widely evaluated in different studies, and the performance is outstanding. Meng et al. [Meng99] also proposed methods for estimating the usefulness of text databases based on the probabilistic model.

Fuhr [Fuhr99] presented a decision-theoretic model for selecting data sources based on the retrieval cost and some typical information retrieval parameters. The model aims at retrieving a maximum number of relevant documents at minimum costs. However, the performance of the model depends on how the broker knows the number of relevant documents in each data sources.

2.3.2 Based on Feedback from Past Queries

As metasearching in the Web environment and metasearching distributed text collections have different characteristics throughout, using only theoretic-based algorithms maybe insufficient. Focusing on metasearch search engines, some selection methods are developed based on past performances of the search engines. SavvySearch [Dreilinger97] is a metasearch engine that categorizes search engines into a hierarchical directory for searching. While searching, users need to specify the category to search, and search engines will be selected based on what category has been chosen. A prototype for selection that considers the predicted recall for a given query is also presented.

MetaSEEK [Benitez98] is another metasearch index that considers past performance. It utilizes the historical information of queries searched to deal with new query, and select the highest performance search engine to do metasearching. However, the search engines' past performance is based on all users' feedback. Hence a new user will obtain poor metasearching performance since his opinion may not be compatible with other users' judgments.

Conversely, the Inquirus2 [Glover99] metasearch engine considers individual user's past feedback. Search engines are grouped into different categories. When submitting queries, users need to specify their information needs by selecting a category in which to search. However, each search engine requires manual categorization and different categories contain different predefined attributes that limited the number of search engines which can be included for metasearching.

In addition, to guarantee the search quality at a certain level when utilizing the past performances of the search engines, the system needs to be executed for some period of time in order to collect enough historical information for the prediction. As a result, there are selection methods developed based on a pre-built centralized index approach.

2.3.3 Based on Cooperative Meta-index or Content Summaries

Most of the traditional selection algorithms use statistical data to characterize the contents of each data source. Statistical data of each data source are combined as a large centralized index that can be referred to as *meta-index* or *content summaries*. It usually includes the document frequencies of words which appear in the data source with other simple statistics. Relevant data sources are selected by evaluating the degree of similarity between the given query and the meta-index.

ProFusion [Gauch96] uses meta-index in terms of a set of hand-built knowledge base. Each search engine has one tailor-made knowledge base. Selection is made by mapping the given query to the knowledge bases for the most relevant search engine to search. Though the performance of this selection method is good, hand-building a knowledge base for each search engine is too subjective and not scalable.

Yuwono et al. [Yuwono97] ranked distributed text server by developing a centralized broker which maintains a document frequencies table for each server. The main disadvantage for this method is that it requires all servers to cooperate with the broker by submitting up-to-date frequencies data regularly.

Another method that also requires the cooperation of databases is GlOSS [Gravano99]. The meta-index is constructed by combining the indices of the databases, which are document frequencies of words. When a query is accepted, databases are ranked according to their appropriateness to the given query. This is done by estimating the number of documents in each database for which query similarity was greater than a predefined threshold. A score will be created for each database through summing up the similarities values, and databases will be chosen according to the scores.

Instead of using the statistical information of words contained to characterize a collection, Xu et al. [Xu98] proposed to use the phase information accompanied by query expansion. It was believed that carrying out a collection selection within such kind of distributed retrieval systems will be effectively poorer comparing to a single centralized retrieval system.

Because most of the databases do not release the required information to the public,

it is sometimes difficult to build the meta-index in the Web environment although meta-index is widely investigated and is one of the promising selection algorithms.

2.3.4 Based on Probe Queries

In the case where meta-index cannot be obtained from databases that do not provide cooperation, metasearch engine has to resort to *probe queries*. The idea of probe queries draws on the nature of search engine (or searchable database) in that it returns a set of results for any given query. By sending a set of sample queries to a search engine and then, downloading and analyzing the documents from the search results, statistical information can be extracted. This kind of information will be considered as a representative of all the documents inside that search engine.

Callan et al. [Callan99] introduced query-based sampling approach to sampling text database contents via the normal process of running queries and retrieving documents. Using this kind of probe queries, indexing or categorizing a database needs no cooperative work, which is important when applying search engine selection in the Web environment. It also points out the problem: the size of the database cannot be easily estimated by using probe queries approach.

Another probe query method introduced by Hawking et al. [Hawking99] is called Lightweight Probes (LWP). A small amount of probe queries will be submitted to each of the databases at querying time. The probe results are non-random, which can greatly improve the efficiency of the probing process. After comparing with different server ranking methods, manually generated LWP achieved results were superior on all measures. However, up-to-date document frequency statistics are needed from the servers, and therefore the cooperation problem surfaces again.

Craswell et al. [Craswell00] compared three server selection methods: CORI [Callan95], gGlOSS [Gravano96] and CVV [Yuwono97], based on statistics extracted from the results of probe queries submitted to each server. A method for estimating server effectiveness when performing probe queries is also introduced. Results show the method needs further improvements, and CORI has the best performance compared to other selection methods.

The advantage of using probe queries to select search engines is that a metasearch engine can address a new search engine without cooperation. On the other hand, since the set of probe queries is usually static, the results returned from a search engine are non-random. Whether a search engine can be characterized correctly highly depends on the quality of the probe queries selected.

2.4 Summary

In this chapter, we illustrated the architecture of common metasearch engines, and defined specialty search engines. To help us gain understanding of the complexity of selecting search engines, we described some of the existing selection algorithms and highlighted some of their features. Nevertheless, applying text collection or database selection methods directly to solve the search engine selection problem may not be sufficient. In order to select the best relevant search engines for a given query, the characteristics of each search engine should be considered. For example, the response time for the search engine, the presentation style of the search results, the search options provided, and the information provided for each result (result summaries, document sizes), do not exist in most database environments, and can be candidates for improvement.

Chapter 3

Search Engine Categorization

3.1 Introduction

For the metasearch engine that only metasearch a small number of search engines, the selection of search engines is not a key issue since the query received by the metasearch engine will be sent to all the search engines. However, in this research we propose to metasearch a large number of specialty search engines. Therefore, to select suitable sources to search is the key issue. For both the user and the metasearch engine, it is not easy to select only relevant search engines to perform a metasearching. On one hand, users usually do not know which search engines will contain the target information, because there are many of them, and users have no experiences with most of these. On the other hand, it is also difficult for the metasearch engine to choose relevant search engines, since the information provided by users regarding the search query is often insufficient.

To solve this *search engine selection problem*, a metasearch engine should find some ways to acquaint the search engines it employs so as to select the most relevant search engines for every user query. We believe that by classifying specialty search engines into a hierarchical structure can help metasearch engine to select relevant search engines to perform metasearching. Hence, the cost of querying unsuitable search engines and processing unsuitable search results can be reduced. Because the results returned to the user will be more promising, leading to a reduction of the total cost of each search.

Instead of constructing a hierarchy from scratch to categorize specialty search engines, we decided to utilize a well-known category hierarchy: the DMOZ Open Directory Project (ODP) [ODP03]. The ODP contains well-defined hierarchical category which is originally designed to categorize the Web pages on the Internet. The approach we proposed here to categorize specialty search engines adopts parts of the hierarchy of the ODP. Document sampling techniques will be used to collect sets of documents contained in different search engines being categorized. After analyzing the statistical data extracted from the documents, each search engine will be ranked for each category in the hierarchy according to how relevant they are to those categories. And finally, by keeping only the most relevant search engines in each category, a hierarchical search engine category can be built.. Compared to other human-edited categorization, the categorization processes we proposed here has the advantage that only small amount of customization efforts needed to categorize specialty search engines, thus creating a more efficient categorization.

3.2 Categorization Algorithm

In this section, we will first describe the Open Directory Project which provides us the hierarchical structure for our category. Next, we will explain step by step the proposed categorization algorithm.

3.2.1 The Open Directory Project

The hierarchical structure we used to categorize specialty search engines is adopted from the ODP. Currently, ODP is the largest and most comprehensive human-edited directory for the Web. Huge number of Web pages and Web sites has been categorized into a predefined hierarchical category. The structure of the category does not change all the time. It only gets expended when new concepts are added in real life whereas updating the structure frequently is not the key

objective of ODP. Tens of thousands of volunteer editors are employed to maintain the directory and this makes the directory more updated compared to other human-edited directories such as Yahoo!. As of 10 May 2002, the ODP consists of 3,407,364 sites, 48,606 editors and 395,824 categories. Another feature of ODP is that all the category data is freely available to all takers under a free use license. As more than one hundred search engines and Web sites are licensed to use the ODP, we believe the quality of the directory is evident.

One assumption has been made here, and that is, the number of specialty search engine in the Internet is fewer than the number of Web pages. Evidence can be obtained by comparing the numbers of special search engine and Web page indexed in the ODP, which can be said to be a synopsis of the Internet. Inside the ODP, the number of specialty search engine indexed is much smaller than the number of Web page indexed. As shown in Figure 3.2.1a, the category " *Top / Computers / Software / Internet / Clients* " in ODP is an example. Including all sub-categories, 1655 Web pages have been indexed, but only 136 of the Web pages indexed contain a specialty search engine. Therefore, we believe that specialty search engines only occupy a small portion of the Internet, and using the complete hierarchy of ODP to categorize search engines is not efficient. Given that the ODP hierarchy was originally designed to categorize all kinds of Web pages, a simpler version of the hierarchy should be used to categorize specialty search engine instead. The process for obtaining such a hierarchy will be described in Chapter 4.

ODP Category = Top/Computers/Software/Internet/Clients	
Number of sub-categories	104
Number of Web page indexed	1655
Number of specialty search engine indexed	136
Approximate number of specialty search engine per category	1

Figure 3.2.1a Number of specialty search engine in ODP as of 9th July 2002

3.2.2 Document Sampling

One of the characteristics of specialty search engine is that the full set of the documents indexed is encapsulated by a search interface. Without the authority granted from the Web site, outside parties can obtain only subsets of the collection. On the other hand, acquiring and analyzing the full collection for every search engine is also not cost-effective, if not impossible. Consequently, we propose to acquire only a small number of documents for each search engine, and use those documents as samples to represent the whole collection indexed by that search engine.

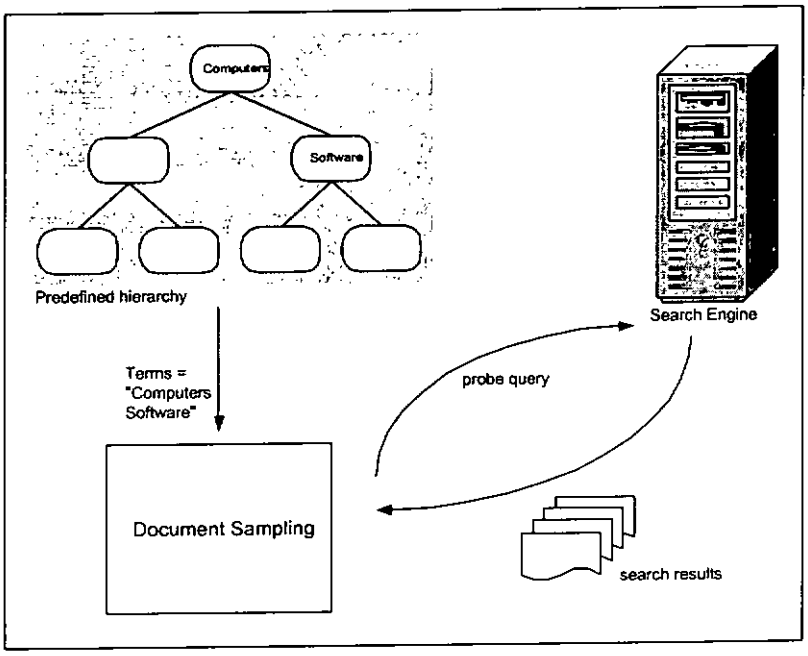

Figure 3.2.2a Sampling documents from search engines

The sampling method we used here is *probe queries*. The idea of probe queries exploit the nature of search engine being that it always returns a set of results considered relevant to a given query. The sampling process, as depicted in Figure 3.2.2a, works as follows: to categorize a search engine, a set of sample queries will be sent to the search engine. The sample queries we used were generated using words from a predefined hierarchical category. After downloading and analyzing

the documents from the search results, statistical information can be extracted. And this kind of information can be used to characterize that search engine.

As mentioned, a categorization structure of the ODP is used for the document sampling process. Similar to other Web directories, the selected hierarchy ***H*** contains a number of categories organized into multiple levels such that the higher level categories have more generic meanings than those at the lower levels. As shown in Figure 3.2.2b, for each search engine, our sampling algorithm collects sample documents for all the categories contained in ***H***. To collect documents for a category ***C***, a probe query ***q*** = { *name of* ***C*** + *name of parent of* ***C*** } is formed. For example, if ***C*** = "*Software*" and the parent of ***C*** = "*Computers*", then the probe query ***q*** for ***C*** is "*Computers Software*". ***q*** is then sent as search queries to the search engines and the documents listed in the search result will be downloaded as document samples.

```
CollectDocumentSample(Hierarchy H, Set of Search Engines S)
    foreach category node Ci in H
        form a probe query q by using the name of Ci
        and the name of the parent of Ci
        foreach search engine Si in S
            send q to Si
            ResultSet = search results returned by Si
            download the first m-top documents in ResultSet
        next Si
    next Ci
return
```

Figure 3.2.2b Steps of collecting document samples

The rationale behind the sampling method is that search engines belonging to a category should be able to return search results that are relevant to that category. As a result, search engines can be categorized by measuring the relevancies of documents returned. For example, a query "C programming" should retrieve a number of relevant documents from a computer-related search engine, while a query "Chicago Bulls" is likely to retrieve few or no document from the same search engine. Compared to other methods like [Callen99] that generates probe queries by randomly choosing words from a general dictionary, and [Gauch96] that constructs probe queries manually, our method raises the effectiveness by binding probes to words relevant to the categories.

For each search engine, a number of search results will be returned and the first m-top documents of the results will be fetched and collected as the document samples. In addition, since a probe may retrieve no documents, each execution will collect from zero to at most m sample documents. In order to collect enough samples, m cannot be too small; however, a large m will make the categorization process expensive. Additionally, because users are usually interested in only the top-ranked search results, say, the results on the first page of the result list, we believe that the results ranked lower are not suitable to be sample documents as they are less important to users. As shown in the later section, we have used $m = 50$ for our experiments, roughly equal to 2 pages of the result lists for many search engines.

When downloading the sample documents, the corresponding relevance scores or values given by search engines are excluded. Since different search engines use different ranking schemes to define how a document is relevant to a given query, the relevance scores are not comparable. Although common ranking schemes use the same basic term-frequency [BaezaYates99] concept, different variations have been

employed, and therefore exploiting the relevance scores directly is not appropriate. Alternatively, the rankings provided by the search engines are more comparable. For each set of rankings, the higher ranked documents are generally more important than those at the lower ranks. Thus the use of the ranking given by a search engine has in a way indirectly used some information provided by the relevance calculation of the search engine. As a result, the rankings (e.g. first, second or third), rather than the scores (e.g. 99% or 80%) of the documents will be recorded.

Furthermore, different search engines provide different search options. Obviously, if such options can be utilized effectively, the search results returned can be improved. However, we believe that to understand and utilize all the search options provided by different search engines is too expensive to be practical. As a result, when sending probe queries to the search engines, we only request to perform a common *all-the-word* match, and do not make use of any advanced options.

3.2.3 Relevancy Calculation

Our final category hierarchy groups different search engines for metasearching. Each sub-category contains a number of search engines, ranked according to their relevancy to the category. To estimate the relevancy of a search engine to a category, a corresponding *relevancy score* will be computed. For each search engine, the *relevancy score* is computed based on the statistical information calculated from the sample documents collected. In short, our approach first calculates the frequencies of terms in each document, then the values will be amended by the weights according to the ranking position of the documents. Finally, the values will be used to compute the *relevancy scores* by including the hierarchical information.

The reason for determining the relevancy using the sample documents is that we believe for a metasearch engine, the most direct and practical way to categorize search engines is to get some actual experiences from the search engines for the users in advance. Based on informal interviews with users and observations on search behavior, we believe that users usually qualify a search engine according to the quality of the past searches they made. Based on past experience such as the *scores* or *number of matches* of previous search results, users develop a personal but fairly definite perception of the *quality* of a search engine with respect to certain types of searches. As a result, our categorization approach collects sample documents from the search engines, and examines the quality of the search results in order to determine the relevancy of search engines to specific categories.

Additionally, compared to other approaches, collecting and examining the search results in the categorization process is more practical and unbiased, although it maybe costly. Other researchers have proposed to employ different factors to categorize searchable databases or document collections; however those factors may not be easily obtained from the search engines being categorized. The following are some examples. [Ipeirotis00] proposed to classify text databases automatically using probe queries. Databases are classified based on two metrics, *Coverage* and *Specificity*. While *Coverage* is the number of documents that match the queries and *Specificity* is the fraction of the matched documents contained in the databases. One major problem with this approach is that search engines normally do not disclose the total number of documents that they contain. As *Specificity* cannot be determined for many of the search engines, the number of search engines that can be classified is limited.

[Dolin99] described another collection classification algorithm. A *Summary Profile*

is constructed for each collection for classification. Nevertheless, the profiles are constructed by analyzing all the individual documents in the collections, and this makes the algorithm not applicable for categorizing search engines. As the indices of the documents are not disclosed, and the documents can only be retrieved though the search interfaces, to obtain the entire set of documents contained in a search engine is not practical.

GlOSS [Gravano94] uses a meta-index to search multiple databases, where the meta-index is constructed by integrating the indices of each of the databases. The problem is that each database must cooperate with GlOSS by supplying up-to-date index information. As a result, this approach is also not feasible for categorizing search engines, since search engines normally do not release such kind of index information.

Furthermore, there are other factors that can be used to qualify a search engine. For example, most search engines not only supply the documents, but also the corresponding scores for each document. However, different search engines have their own searching algorithm and ranking scheme; the scores given by different search engines are not comparable, and thus are not useful when categorizing search engines. Alternatively, as the scores given in the result sets are not comparable, our approach exploits the ranking position of the documents as one of the factors to determine the relevancy of the documents.

3.2.4 Term Frequency

The first step of our algorithm calculates the relevancy of the documents based on the *term-frequency* concept [BaezaYates99]. The concept is based on the idea that the number of occurrences of the search term in a document is an indication of the

relevancy of the document relative to the search term.

Let q be the probe query used, S be the search engine being categorized, and D = { d_1 ... d_m } be the set of documents returned by S. Then $tf_{i,q}$, the normalized term frequency of document d_i using q is determined by:

$$tf_{i,q} = \frac{freq_{i,q}}{\max freq_{l,q}} \qquad (Equation\ 3.1)$$

where $freq_{i,q}$ is the raw frequency (the number of occurrence) of q in d_i, and $\max(freq_{l,q})$ is the largest frequency of q found in D. By dividing the frequency by the maximum frequency, the value of tf is normalized and falls into the range $0 \le tf \le 1$.

Recall that the corresponding ranking positions of the documents are used as a factor for computing the relevance scores. As a result, the value of tf will be amended by w, which is the ranking factor, as follows:

$$w_i = \frac{m - (\alpha \times o_i)}{m} \qquad (Equation\ 3.2)$$

$$tf_{i,q}' = tf_{i,q} \times w_i \qquad (Equation\ 3.3)$$

while m is the total number of documents in D, and o_i is the ranking position of d_i given by S. The value of w is used to weight the tf values and the importance w is controlled by a constant α. The effect of the weightings is set to reflect our belief that a document's relevancy to a query should depend primarily on the number of

occurrence of the term, but only secondarily on the ranking position of the document. By letting m = 10, Figure 3.2.4a show the behaviors of w if α is set to 1.0, 0.3 and 0.1 accordingly. As shown in the table, if α is set to 1.0, the value of w for the document ranked the first (0.9) will be eight times more than the document ranked ninth (0.1), hence the effect of the ranking position is too large. Alternatively, if α is set to 0.1, the different between the first and last documents is too little, and therefore the effect of the ranking position is not sufficient.

	$\alpha = 1.0$	$\alpha = 0.3$	$\alpha = 0.1$
documents	w_i	w_i	w_i
d_1	0.90	0.97	0.99
d_2	0.80	0.94	0.98
d_3	0.70	0.91	0.97
d_4	0.60	0.88	0.96
d_5	0.50	0.85	0.95
d_6	0.40	0.82	0.94
d_7	0.30	0.79	0.93
d_8	0.20	0.76	0.92
d_9	0.10	0.73	0.91
d_{10}	0.00	0.70	0.90

Figure 3.2.4a Behavior of w when α is set to different values

Researchers like [Mowshowitz02] have also including the ranking position as an important factor when calculating the quality of search results. Nevertheless, there is no standard way to define how important the ranking position should be. For our experiments, we have chosen $\alpha = 0.3$. As a result, the corresponding minimum and maximum values for w_i will be approximately 0.7 and 1.0, which may in turn cause a variation of the frequency values of the documents between 1% and 30%.

After calculating the tf values for all the documents in D, the final term frequency value $TF_{s,q}$ for the search engine S using query q is determined by:

$$TF_{S,q} = \frac{\sum_{i=1}^{m} tf'_{i,q}}{\max tf'_{i,q}} \qquad (\textit{Equation 3.4})$$

where $\max(tf'_{i,q})$ is the largest TF value computed using q. Consequently, the computed $TF_{S,q}$ value represents how much the search engine S is relevant to the query q. The following are case studies to study the behavior of TF values under different distributions of term frequency values.

EXAMPLE 1: Consider 3 search engines S_1, S_2 and S_3. 10 documents are collected from each search engine using query q generated for a category C. α in (Equation 3.2) is set to 0.3 and as shown in Figure 3.2.4b, the computed maximum *freq* and Σtf values are 27 and 4.526 accordingly. Consequently, as search engine S_1 contains documents that have superior quality, it obtains the highest TF value (1.000). And it is considered that S_1 is the most relevant search engine for C comparing to S_2 (0.448) and S_3 (0.242).

	S_1		S_2		S_3	
documents	*freq*	*tf'*	*freq*	*tf'*	*freq*	*tf'*
d_1	27	0.970	15	0.539	8	0.287
d_2	22	0.766	12	0.418	8	0.279
d_3	20	0.674	10	0.337	5	0.169
d_4	17	0.554	8	0.261	3	0.098
d_5	15	0.472	5	0.157	2	0.063
d_6	12	0.364	4	0.121	2	0.061

d_7	9	0.263	2	0.059	2	0.059
d_8	7	0.197	2	0.056	1	0.028
d_9	5	0.135	2	0.054	1	0.027
d_{10}	5	0.130	1	0.026	1	0.026
Σtf		4.526		2.028		1.096
TF		1.000		0.448		0.242

Figure 3.2.4b Examples of computing *TF* value

EXAMPLE 2: Consider another set of search engines S_4, S_5 and S_6 with same number of documents collected and the same α value. In this example, the distribution of ***freq*** is different for each search engine. For S_4, the first 3 documents contain very high ***freq*** compared to the last 7 documents; for S_5, ***freq*** decreases gradually; and for S_6, ***freq*** is relatively uniformly distributed in all 10 documents. The test data are setup such that total frequency values for all the 3 search engines are equal. As shown in Figure 3.2.4c, though the total term frequencies found in the top 10 documents are the same for the 3 search engines, the result ***TF*** values computed are *not equal*. This circumstance illustrates one characteristic of our algorithm, based on the design intent behind the algorithm. *We believe the most relevant result set should contain outstanding quality documents of outstanding quality and those documents should be ranked at the top segment of the result set.* As the top 3 documents in S_4 contains the outstanding frequencies, the algorithm gives it the highest ***TF*** value (1.000). Although S_5 and S_6 also contain quality documents, the ***TF*** values computed are relatively smaller (0.954 and 0.905 correspondingly).

	S_4		S_5		S_6	
documents	***freq***	***tf'***	***freq***	***tf'***	***freq***	***tf'***
d_1	17	0.970	10	0.571	5	0.285

d_2	15	0.829	8	0.442	5	0.276
d_3	11	0.589	7	0.375	5	0.268
d_4	1	0.052	6	0.311	5	0.259
d_5	1	0.050	5	0.250	5	0.250
d_6	1	0.048	4	0.193	5	0.241
d_7	1	0.046	4	0.186	5	0.232
d_8	1	0.045	3	0.134	5	0.224
d_9	1	0.043	2	0.086	5	0.215
d_{10}	1	0.041	1	0.041	5	0.206
Σ *freq*	50		50		50	
Σ *tf*		2.714		2.588		2.456
TF		1.000		0.954		0.905

Figure 3.2.4c How the distribution of *freq* affects the *TF* value

3.2.5 Relevancy Score

We believe that the hierarchical structure of the category tree can be exploited as extra information to augment the precision of the categorization process in the computation of the final relevancy score. In other words, the relevancy score of a search engine with respect to a category will be influenced by the search engine's term frequency values in different but related categories.

The rationale behind this decision is the following. Nodes at the lower levels of the hierarchy of categories have more specific meanings than those nodes at higher levels. Hence the relevancy of a search engine S with respect to a category C should be affected by the relevancy of S with respect to the child categories of C. For example, consider two search engines S_A, S_B and a category "*Computer*" which has two child categories "*Software*" and "*Hardware*". The initial relevancy scores for

S_A and S_B in "*Computer*" are nearly equivalent. However, the scores of S_B in "*Software*" and "*Hardware*" are higher than S_A's. We believe that in this case, S_B should be more relevant to the category "*Computer*" than S_A, as reflected by a higher relevancy score.

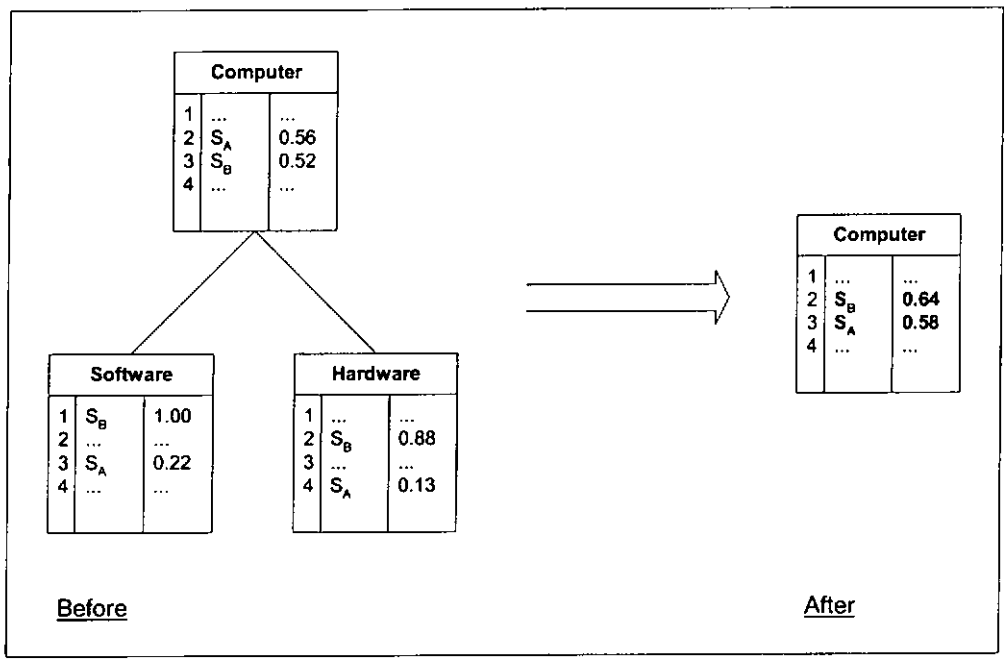

Figure 3.2.5a Relevancy scores in the parent category are affected by the corresponding scores in the child categories

To include the hierarchical structure information, the ODP link information is used to give different weights to different categories. We believe that for a parent category, its child categories should not be treated as equally important. For example, "*Internet*" and "*Shopping*" are two of the child categories of the "*Computer*" category. The relevancy scores in the "*Internet*" category should be able to affect the scores in "*Computer*" more than those in the "*Shopping*" category. As a result, to represent this kind of difference between categories, we assign weights to categories by using the number of Web pages indexed in the ODP as the factor to identify the importance of a category. That is, for a category, the larger the number of Web pages contained,

the higher the weight.

As a search engine's relevancy score for a category will be computed by the frequency values of the child categories, the process for computing all the relevancy scores in the hierarchical category will be done in a bottom-up manner. The first step of the algorithm will assign the relevancy scores for all the search engines in the *leaf node* categories equal to their ***TF*** values. For a search engine S inside a *leaf node* category ***LC***, its relevancy score will be:

$$R_{S,LC} = TF_{S,LC} \qquad (\textit{Equation 3.5})$$

Afterwards, the remaining search engines inside the *non-leaf* node categories will be processed. The relevancy score $R_{S,C}$ of the search engine S for the category C which has nc number of immediate child is given by:

$$R_{S,C} = TF_{S,C} + \sum_{i=1}^{nc} \beta \left(\frac{link_i}{L} \times R_{S,i} \right) \qquad (\textit{Equation 3.6})$$

where $link_i$ is the number of Web pages indexed in ODP for the child category i and L is the total number of pages indexed for all the immediate child of C. β is a constant to control how much the hierarchical information should affect the final relevancy score.

Additionally, to convince the later process that exploits the produced search engine category such as metasearching, the relevancy scores will be normalized by the following formula, and the scores in every category will fall into the range $1 \leq R_{S,C} \leq 0$.

$$R_{S,C}' = \frac{R_{S,C}}{\max R_{I,C}} \qquad (\textit{Equation 3.7})$$

Example 3: Consider two search engines S_7, S_8 and a category C_1 which has two child categories C_2 and C_3. The number of links indexed in C_2 and C_3 are identically equal to 75 and β in (Equation 3.6) is set to 0.2. Initially, inside C_1, the score of S_7 (0.850) is a bit higher than the score of S_8 (0.830). Since the score of S_8 in C_2 is much higher than S_7; after the calculations, the final relevancy score of S_8 (0.940) is higher than S_7 (0.918). This example shows the proposed concepts can be reflected by our algorithms.

C_1		C_2	(75)	C_3	(75)
S_7	(0.850)	S_7	(0.320)	S_7	(0.360)
S_8	(0.830)	S_8	(0.820)	S_8	(0.280)

→

C_1'	
S_7	(0.918)
S_8	(0.940)

Figure 3.2.5b Augmenting relevancy scores by scores in the child categories

Example 4: Consider another example similar to example 3. The only difference is the number of links indexed in C_2 and C_3, which are 30 and 270 respectively. After the calculations, the final relevancy score of S_7 (0.921) is still higher than S_8 (0.897). This is because the large difference of scores in C_2 is reduced since C_2 is relatively less important in terms of the number of links indexed.

C_1		C_2	(30)	C_3	(270)
S_7	(0.850)	S_7	(0.320)	S_7	(0.360)
S8	(0.830)	S8	(0.820)	S8	(0.280)

→

C_1'	
S7	(0.921)
S8	(0.897)

Figure 3.2.5c How the importance of categories affects the calculations

3.2.6 Search Engine Elimination

The final step of the categorization process is to eliminate irrelevant search engines from the categories. After performing the procedures described in the previous sections, each category in the hierarchy contains the same number of search engines. The number of search engines is the same because each search engine has to go through every sub-category using the algorithms. For instance, if 50 search engines are categorized, each category will contain the ranking of those 50 search engines regardless of the scores assigned for the search engines. One reason for eliminating low quality search engines is to reduce the cost of storing useless ranking information in order to increase the efficiency while other processes exploit the produced category.

One method that can be used to eliminate irrelevant search engines is to make every category keep an equally n number of search engines. This method is simple, but it is hard to define n. Also, a number of high score search engines may be eliminated. Alternatively, we can eliminate search engines within categories based on the standard deviation of the relevancy scores.

For each category, a set of search engines is ranked according to their relevancy scores. First, the standard deviation (*stdv*) will be calculated using all the relevancy scores of the search engines within that category. If a search engine has a score smaller than the *stdv*, it will be removed from the category. The following formula is used to calculate the *stdv*:

$$stdv = \sqrt{\frac{n\sum x^2 - \left(\sum x\right)^2}{n^2}} \qquad (\textit{Equation 3.8})$$

After performing the procedures described above, the categorization process is finished and a hierarchical search engine category is produced. The produced category contains numbers of sub-categories in a hierarchical structure. Each sub-category contains numbers of relevant search engines. We believe that metasearch engines will benefit from such kind of search engine category in different ways. The next chapter describes how our experimental metasearch engine utilizes the developed category. The next section also includes experiments for evaluating the explained categorization algorithm.

3.3 Summary

In this chapter, we have presented the categorization algorithm for categorizing search engines. We first discussed why and how we chose the Open Directory Project as the skeleton for hierarchical structure of the final category. Then we discussed the main processes of the categorization algorithm, which are the document sampling and relevancy calculation processes. We explained how the relevancy scores are calculated for the search engines, and how to construct the final search engine category.

In the next chapter, we will discuss experiments for evaluating the categorization algorithm presented in this chapter.

Chapter 4

Experiments

In this chapter, we will first describe how the proposed categorization algorithm is implemented. The setup processes, test bed and test data are explained in details. Second, we describe how we evaluate the performance of the proposed categorization algorithm by comparing the rankings of the search engines produced by the algorithm to those constructed by human judgments.

4.1 Experiment Setup

Recall that our categorization algorithm produces a search engine category by categorizing the number of specialty search engines into an adopted hierarchical structure. In this section, we will describe how the hierarchical structure is selected, what kind of search engines is selected and how the algorithm is implemented to produce the final category.

4.1.1 Selecting the Hierarchical Structure

As mentioned in the previous chapter, only a subset of the ODP hierarchy will be needed. Our intention was to use a hierarchy that contains fewer than 2000 pages indexed and 30 child categories. In additional, the target hierarchy should not have too many or too few levels.

Initially, from the Top category, we have chosen the Computers category as the starting point. Within the Computers category, there are 40 child categories, and the category Software was selected because it is the largest sub-category in Computers.

As of 9th July 2002, it has indexed 39,323 Web pages. Using the same reason, the category Internet, which has indexed 5110 Web pages, was selected within the Software category. Last, the category Clients, which has 1732 pages, was selected within the Internet category.

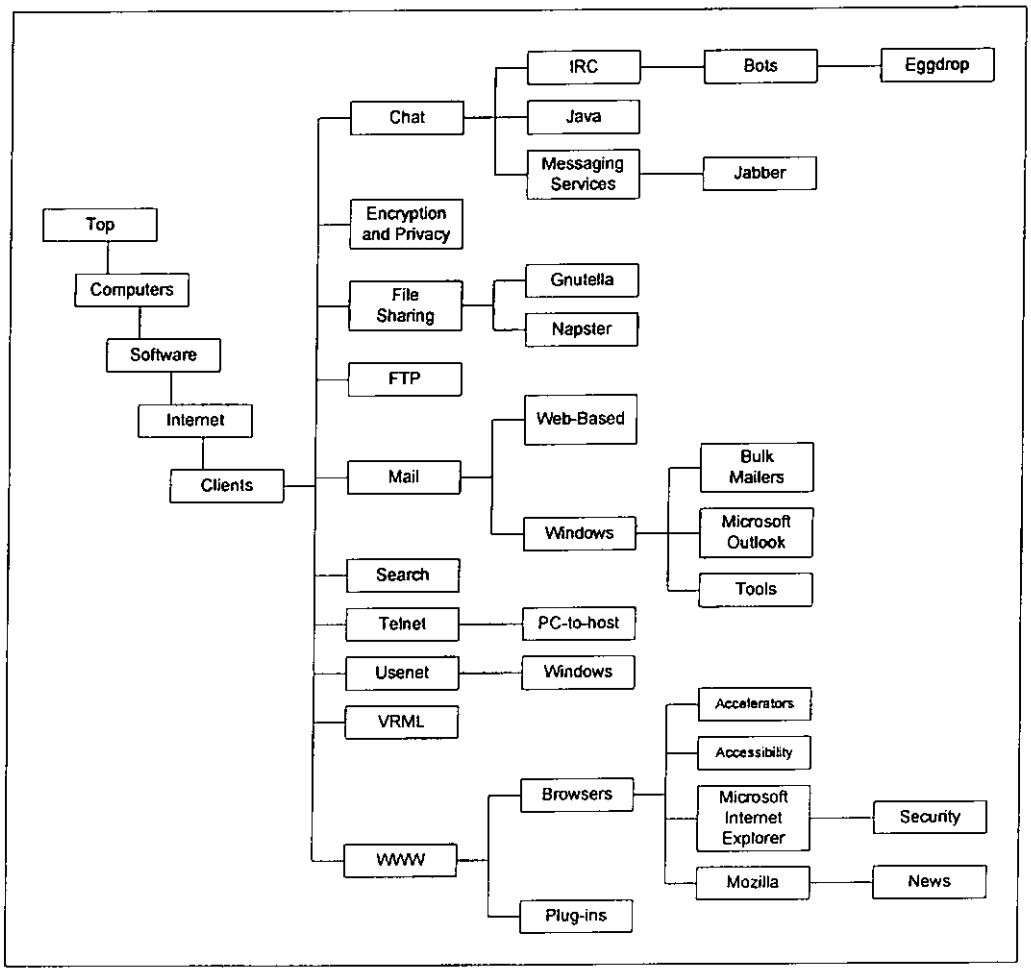

Figure 4.1.1a The hierarchical structure selected

The selected simplified hierarchy, which was used for the experiments in this research, was formed as shown in Figure 4.1.1a. Within the selected " *Top / Computers / Software / Internet / Client* " category, there were 104 child categories. In order to reduce the size of the hierarchy used for the experiments, those child categories which we believe are not significant were excluded. To identify the importance of a category, we consider the number of Web pages it indexed and the

number of specialty search engines it included. All child categories which have fewer than average indexed Web pages were removed and categories that do not index any specialty search engines were also removed.

4.1.2 Selected Specialty Search Engines

As shown in Figure 4.1.2a, 20 specialty search engines were selected for the experiments. In order to demonstrate the effectiveness of the proposed categorization algorithm that it is capable to categorize relevant search engines into the selected hierarchy structure, half of selected search engines are belong to the computer domain and half of them are not. We believe only the computer-related search engines which contain relevant resources will be categorized into the selected categories by the algorithm.

Search Engine (*Computer-related*)	**URL**	**Description**
Apple	www.apple.com	computer company
IBM	www.ibm.com	computer company
Java Sun	java.sun.com	Java official site
Sun Microsystems	www.sun.com	computer company
Tucows	www.tucows.com	software downloads
Macromedia	www.macromedia.com	software company
Borland	www.borland.com	software company
Internet.com	www.internet.com	Internet news
Redhat	www.redhat.com	Linux and software company
RFC Editor Webpage	www.rfc-editor.org	RFC collections
Search Engine (*Non-computer*)	**URL**	**Description**
CBS Sportsline	cbs.sportsline.com/u/cbs/sports/	TV channel
NASA Spacelink	spacelink.nasa.gov	US NASA web site
Science News Online	www.sciencenews.org	science news
Discovery Channel	dsc.discovery.com	TV channel
U.S. Nation Library of Medicine	www.nlm.nih.gov	Library
Prevline	www.health.org	health organization
Manufacturing.net	www.manufacturing.net	manufacturing news and articles

The Whitaker Foundation	www.whitaker.org/news	biomedical engineering news
The Internet Bookshop	www.bookshop.co.uk	online bookshop
ebay	www.ebay.com	online bid company

Figure 4.1.2a Search engines selected for the experiments

4.2 Implementations

The first step of the categorization algorithm was the document sampling process. Probe queries were generated using the selected hierarchy and were sent to all selected search engines. In order to reduce the time required to parse the search results and download the documents, only the top 50 documents were downloaded for each search engine for a probe query. As there were 34 categories in the selected hierarchy, a maximum of 1,700 documents were collected from each search engine.

4.2.1 Document Sampling

Figure 4.2.1a illustrates the implementation of the document sampling processes in detail. First, probe terms were generated using the ODP data, which were downloaded from the Open Directory Website (http://dmoz.org). Besides, tailor-made profiles were created for each search engine, and each of them contained the detail information about the search engine's search interface. Probe queries were then generated by using the generated probe terms and information contained in the profile. After submitting a query to the target search engine, a search result page was returned. If the result page contained one or more search results, it was then passed to the result extraction process. URLs linking to the search results was then extracted. As different search engines had different styles and formats for the result page, customization was needed for the extraction, and such information was stored in the profiles of the search engines. Finally, the documents linked by the extracted URLs were fetched and saved as the sample documents.

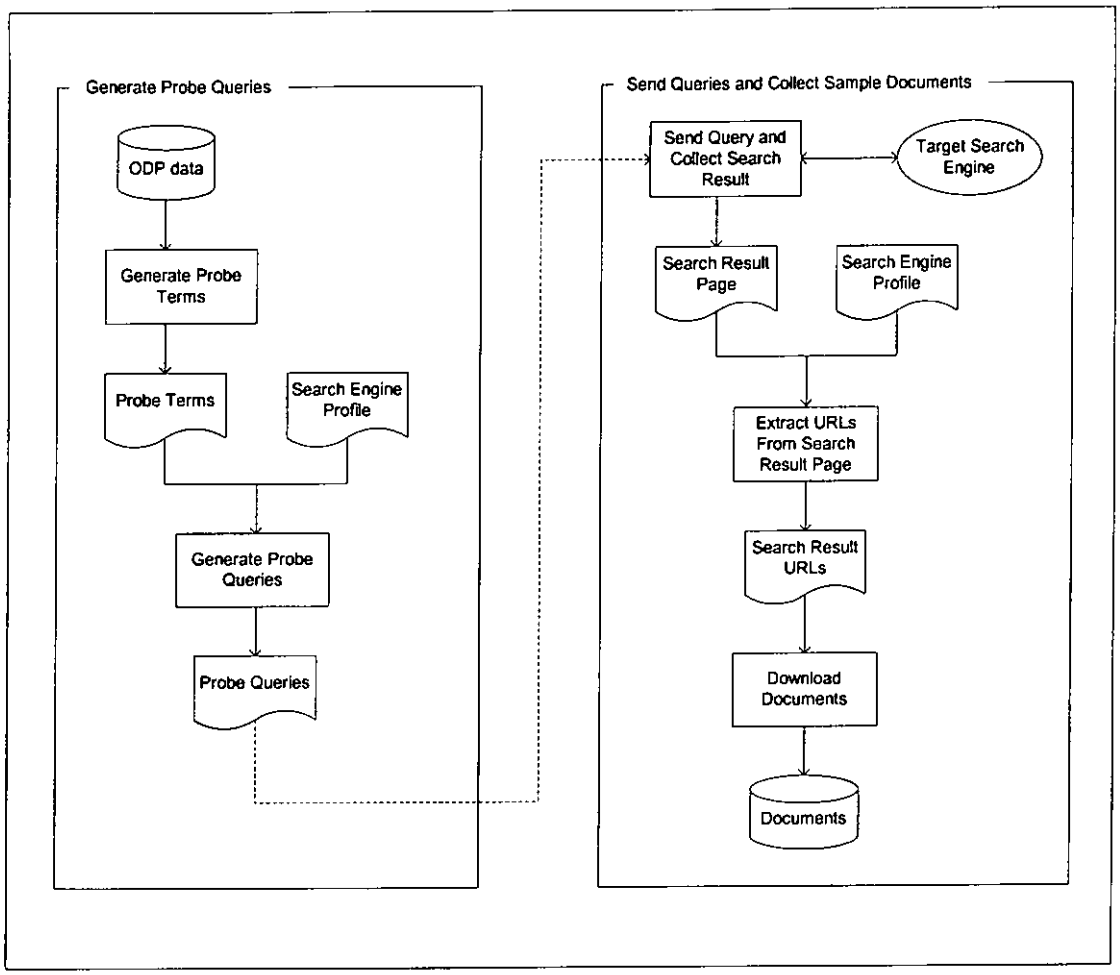

Figure 4.2.1a Processes flow for document sampling

4.2.2 Relevancy Calculation

The first step for calculating the relevancy scores was to count the frequencies of terms in the sample documents. Before doing this, basic stemming method [BaezaYates99] was applied to the documents. Based on a hand-built dictionary, words in the documents were converted to their basic grammatical form. For example, the words *mails*, *mailed* and *mailing* in the documents were all converted to the word "*mail*". The algorithms described in chapter 3 were then applied to calculate the term frequencies for the stemmed sample documents. Finally, each search engine was assigned relevancy scores for different category, and the hierarchical search engine category was produced as shown in Figure 4.2.2a.

The Search Engine Category

Category	*Specialty Search Engines*
Eggdrop	Internet.com
Bots	Internet.com
IRC	IBM, Internet.com
Java	IBM, Internet.com, Java Sun, Sun Microsystems
Jabber	IBM, Internet.com
Messaging Services	IBM, Internet.com
Chat	IBM, Internet.com
Encryption and Privacy	IBM
Gnutella	IBM, Internet.com
Napster	IBM, Internet.com
File Sharing	IBM, Internet.com, Java Sun
FTP	IBM, Internet.com, Redhat
Web-Based	IBM, Internet.com, U.S. Nation Library of Medicine, Redhat
Bulk Mailers	Internet.com
Microsoft Outlook	IBM, Internet.com
Tools	Borland, IBM, Internet.com, Java Sun, Redhat, Sun Microsystems
Windows	IBM, Internet.com, Java Sun, Macromedia, NASA, Redhat, Sun Microsystems, Tucows
Mail	IBM, Internet.com, Sun Microsystems
Search	Apple, IBM, Internet.com, Java Sun, U.S. Nation Library of Medicine, Redhat, Sun Microsystems
PC-to-host	IBM
Telnet	IBM, Internet.com
Windows'	IBM, Redhat
Usenet	IBM, Internet.com, Redhat
VRML	Internet.com
Accelerators	Borland, Internet.com

Accessibility	Borland, IBM, Internet.com, Macromedia
Security	IBM, Internet.com
Microsoft Internet Explorer	IBM, Internet.com, Java Sun, Macromedia
News	IBM
Mozilla	IBM, Internet.com, Redhat
Browsers	Borland, IBM, Internet.com, Macromedia
Plug-ins	Apple, Borland, IBM, Internet.com, Java Sun, Macromedia
WWW	Apple, Borland, IBM, Internet.com, Macromedia
Clients	IBM, Internet.com, Java Sun

Figure 4.2.2a The produced category of specialty search engines

4.3 Analysis of Experimental Results

After the categorization process, specialty search engines were grouped into different categories, and search engines in the same category are ranked according to their relevancy scores. Another set of rankings of those search engines are also obtained by human judgments. The evaluation of the categorization algorithm was made by comparing the rankings produced by the algorithm to those produced manually. The experiment indicates that the rankings using these methods were similar.

4.3.1 Human-Judged Rankings

Manual ranking of Search engines were obtained by the following method. First, 20 computer experts were invited as testers to perform the experiments. Many rounds of experiments were conducted. In each round, the testers were given a subject term (*e.g. "mail tools"*) and a set of Web pages fetched from different specialty search engines. They were required to evaluate the relevance between the subject and the set of pages by assigning a relevancy score to each subject-page pair. The scores ranged from 1 (totally irrelevant) to 6 (highly relevant). As shown in

Figure 4.3.1a, a Website was constructed to present the Web pages and to collect the scores given by the testers:

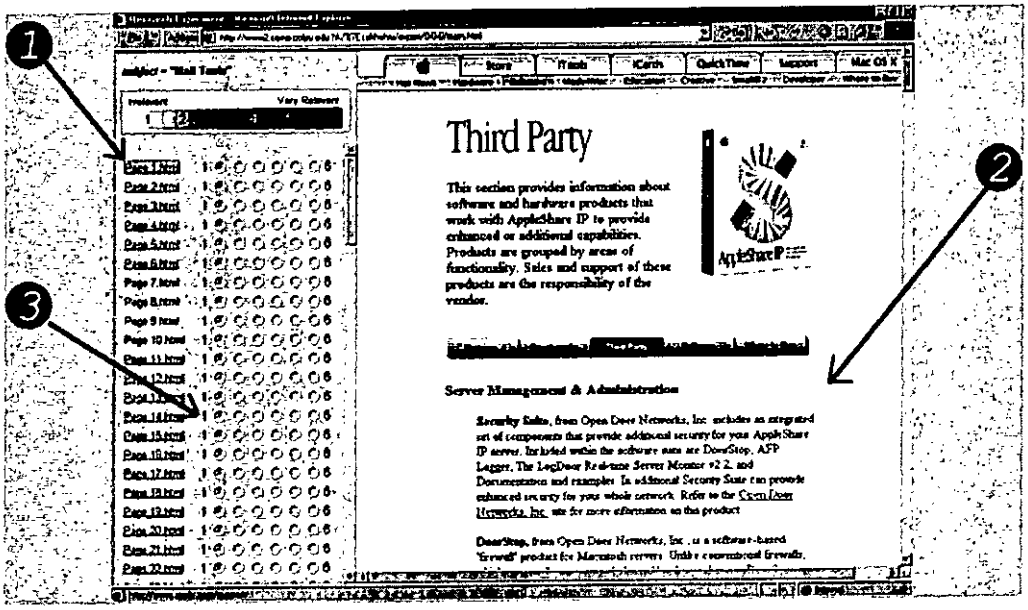

Figure 4.3.1a Obtaining human judgment. (1) links of the pages that need to be evaluated; (2) content of the current page; (3) scores assigned for that page

In fact, each round of the experiment was used to collect the judgments of the proficient for a particular category. The subject given is actually the probe query used for that category, and the pages presented were the sample documents fetched from different specialty search engines. As the proficient had no idea which page corresponded to which search engine, he unknowingly assigned scores for the search results of different search engines. By averaging the relevancy scores from different testers, manual-judged rankings of the search engines were obtained.

4.3.2 Result Analysis

For each sub-category listed in Figure 4.1.1a, both human-judged rankings and the rankings derived from the proposed algorithm were collected and compared. Let us take Figure 4.3.2a here as an example. Figure 4.3.2a presents the rankings for the category " *Top / Computers / Software / Clients / Mail / Windows / Bulk Mailers* ".

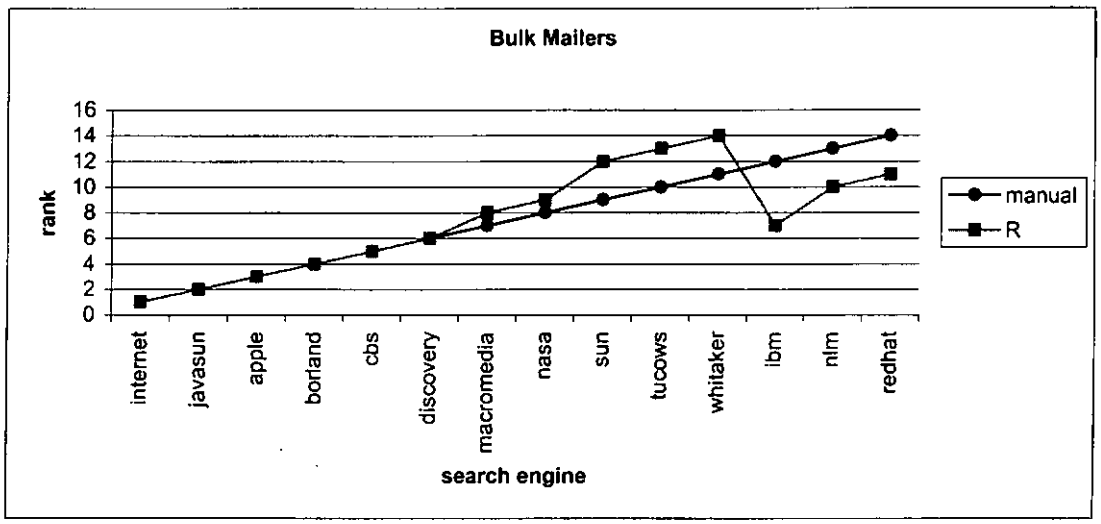

Figure 4.3.2a Category Bulk Mailers

As shown in the figure, the rankings derived by the algorithm (*R*) stayed very close to those judged by humans (*manual*). *R* ranked the top 6 search engines, exactly the same as *manual*; and the 7th and 8th rankings given by *R* only had 1 rank different from those given by *manual*. Although *R* was not able to rank all search engines exactly the same as *manual*, it produced highly similar rankings.

In order to gain insight into the performance of the proposed categorization algorithm, we used the *Spearman's rank correlation coefficient* to measure the correlation between the rankings and used the results to explain the comparison. Figure 4.3.2a presents the definition of the coefficient algorithm.

For each category, we computed the correlation coefficient (r_s) between the human-judged rankings and the rankings given by the categorization algorithm. Figure 4.3.2b listed the computation results. According to Spearman's definition, the value r_s always falls between -1 and +1, the closer r_s is to +1, the greater the correlation between the ranks. In other words, if r_s = 1, both group of rankings are in *perfect positive correlation*. That is, the rankings given by the algorithm is

exactly the same as those given by the humans.

$$r_s = \frac{SS_{uv}}{\sqrt{SS_{uu} SS_{vv}}}$$

where

$$SS_{uv} = \sum u_i v_i - \frac{(\sum u_i)(\sum v_i)}{n}$$

$$SS_{uu} = \sum u_i^2 - \frac{(\sum u_i)^2}{n}$$

$$SS_{vv} = \sum v_i^2 - \frac{(\sum v_i)^2}{n}$$

u_i = Rank of the i-th search engine given by human-judgment

v_i = Rank of the i-th search engine given by the proposed algorithm

n = Number of search engines contained in the sub-category

Figure 4.3.2b Spearman's Rank Correlation Coefficient

We take again the "Bulk Mailers" category mentioned previously as an example.

Search Engine	*Manual*	*R*
internet	1	1
javasun	2	2
apple	3	3
Borland	4	4
cbs	5	5
discovery	6	6
macromedia	7	8
nasa	8	9
sun	9	12
tucows	10	13
whitaker	11	14

ibm	12	7
nlm	13	10
redhat	14	11
r_s	**0.8418**	

Figure 4.3.2b Correlation coefficient of the category Bulk Mailers

Figure 4.3.2b shows the two groups of rankings and the result r_s equals 0.8418, indicating that the rankings given by our algorithm was highly similar to those produced by human judgment. We summarized the correlation results for all the sub-categories in the search engine category in Figure 4.3.2c:

Category	r_s	***Category***	r_s
Eggdrop	0.8626	Mail	0.8393
Bots	0.8788	Search	0.2571
IRC	0.4066	PC-to-host	-0.5636
Java	0.5245	Telnet	0.6484
Jabber	0.4011	Windows	0.3516
Messaging Services	0.6319	Usenet	0.7582
Chat	0.7582	VRML	0.6703
Encryption and Privacy	0.0140	Accelerators	0.3497
Gnutella	0.6364	Accessibility	0.6835
Napster	0.7902	Security	0.8462
File Sharing	0.5209	Microsoft Internet Expl.	0.6967
FTP	0.6044	News	0.4560
Web-Based	0.0462	Mozilla	0.4670
Bulk Mailers	0.8418	Browsers	0.5560
Microsoft Outlook	0.8059	Plug-ins	0.4901
Tools	0.6544	WWW	0.3714
Windows	0.5172	Clients	0.5588
Average r_s		0.5392	
Percentage of having a positive r_s		97%	
Percentage of having a r_s greater then 0.5		65%	

Figure 4.3.2c Correlation coefficients of all categories

As shown in the above figure, many of the categories had a high coefficient score. The average coefficient of all the categories was 0.5392, which indicated a positive correlation. In addition, there was a positive coefficient in 97% of the categories, with 65% of these greater than 0.5. The above results indicated that the proposed categorization algorithm could categorize search engine with a high accuracy, and the quality of the produced search engine category was promising. The complete list of figures of all the comparison results are in the Appendix section.

Figure 4.3.2b presents an example to illustrate how the performance of the proposed algorithm is affected by the hierarchical structure information. Three groups of rankings for the category " *Top / Computers / Software / Internet / Clients / Chat* " are presented. The *"manual"* group is the rankings obtained from human judgments, which served as the optimal rankings; the *"TF"* group ranks the search engines according to their term frequency values only (Equation 3.4), and the *"R"* groups rank the search engines according to their final relevancy scores. (Equation 3.7).

As shown in the figure, both the *TF* and *R* groups rank search engine similar to those by human-judgments. In particular, the rankings given by *R* are closer to the *manual* than those given by *TF*. For example, the rankings given by *manual* for the search engines "*javasun*" and "*sun*" are 2 and 3 respectively, while the rankings given by *R* for the same search engines are 3 and 4; rankings given by *TF* are 5 and 6, which have greater differences than *manual*. Likewise, the rankings given by *R* for the search engines "*cbs*", "*discovery*", "*borland*", "*nasa*" and "*nlm*" all ranked closer to the *manual* rankings than those given by *TF*. In addition, the correlation coefficients of *R* and *TF*, which are 0.7582 and 0.6099 respectively, also indicate that the performance of *R* is closer to the *manual* than those given by *TF*. This result demonstrates that when computing the relevancies of search engines, advanced

performance can be obtained by including the hierarchical structure information.

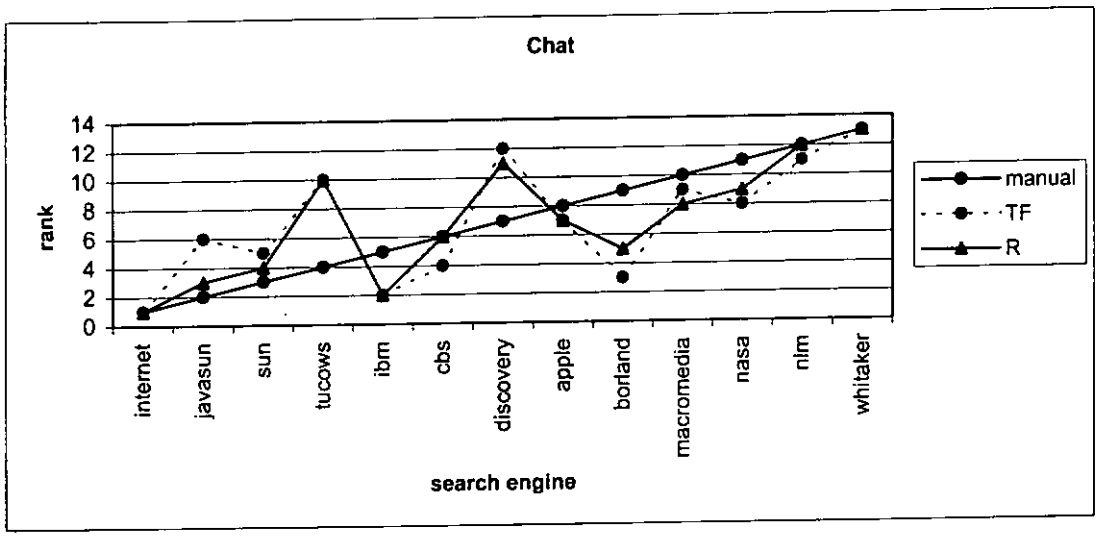

Figure 4.3.2b Ranking produced by (1) human; (2) considering term frequency only; (3) the categorization based on relevancy score

4.4 Summary

In this chapter, we explained the implementation issues in detail, and evaluated the performance of the proposed categorization algorithm by comparing the rankings of the search engines produced by the algorithm to those constructed by human judgment. While the encouraging results show the proposed algorithm is able to construct a sensible search engine category automatically, we would like to show that metasearchers can benefit from the accuracy of automatic search engine selection by exploiting such a directory of search engines. In the next chapter, we will discuss "*Amase*", an experimental metasearch engine that exploits the proposed search engine category presented in this chapter. We will explain how queries are handled, and how to select suitable search engines by utilizing the category.

Chapter 5

Amase – the Experimental Metasearch Engine

In this chapter, we present "*Amase* – *A*ccurate *M*etasearching with *A*utomatic *S*election of *E*ngines", an experimental metasearch engine prototype developed to demonstrate the effectiveness of the search engine category produced by the proposed categorization algorithm. The primitive objectives of Amase are not only to perform metasearching on a search engine category, but also to provide advanced searching capabilities such as personalization and collaboration filtering. However, as those advanced functions are not completed yet, we are focusing on evaluating the performance of the search engine category. At present, Amase only supports the metasearch function. Search results of Amase show that it can select and metasearch relevant specialty search engines for different queries by exploiting the search engine category.

5.1 User Interface

Figure 5.1a presents the user interface of Amase. Amase has four components: "*find*", "*which is*", "*focus-on*" and "*search*". To submit a query to Amase, the user first needs to input some search keywords into the "*find*" section, which is the same as querying traditional search engines. Next, the user has to select a *descriptive term* in the list box inside the "*which is*" section. The *descriptive term* will serve as context to help the user explain the former inputted keywords. For example, if a user wants to discover the function of an Internet chat client named "ICQ", his query may be "*find*: ICQ feature, *which is*: Chat Clients".

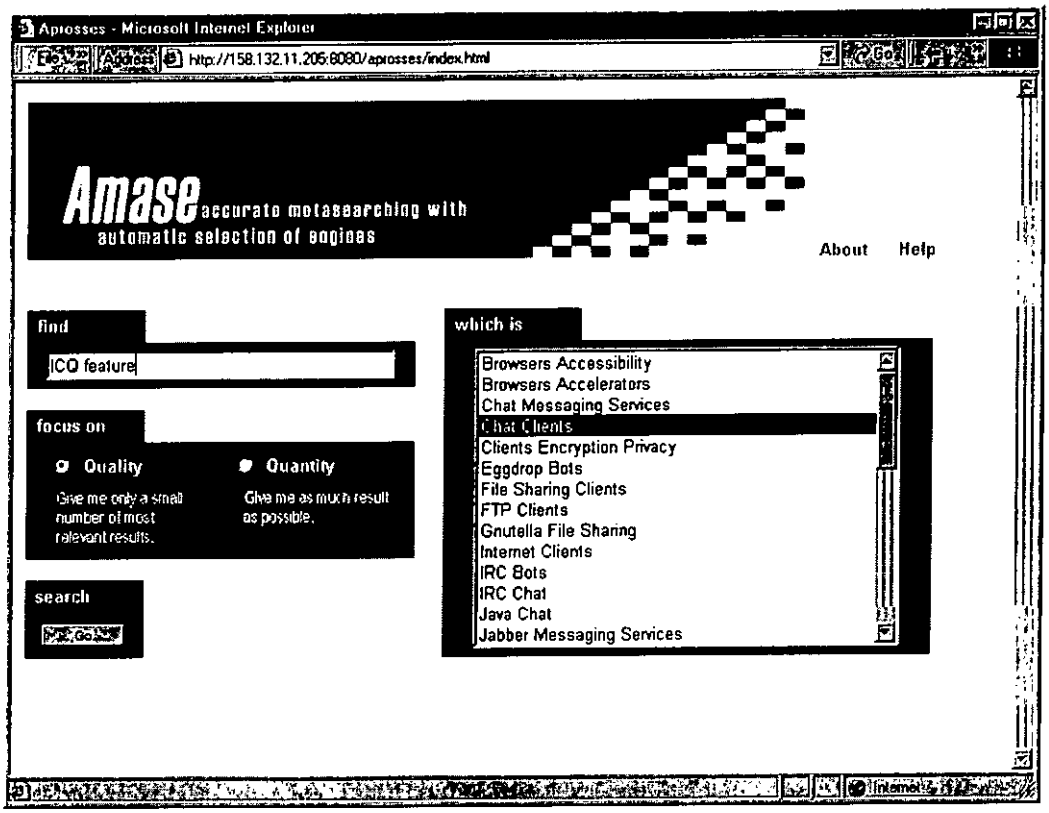

Figure 5.1a User interface of Amase

Figure 5.1b shows another example: if a user wants to find out the system requirements of a Web browser plug-in named Flash, his query may be "*find*: Flash requirements, *which is*: WWW Plug-ins".

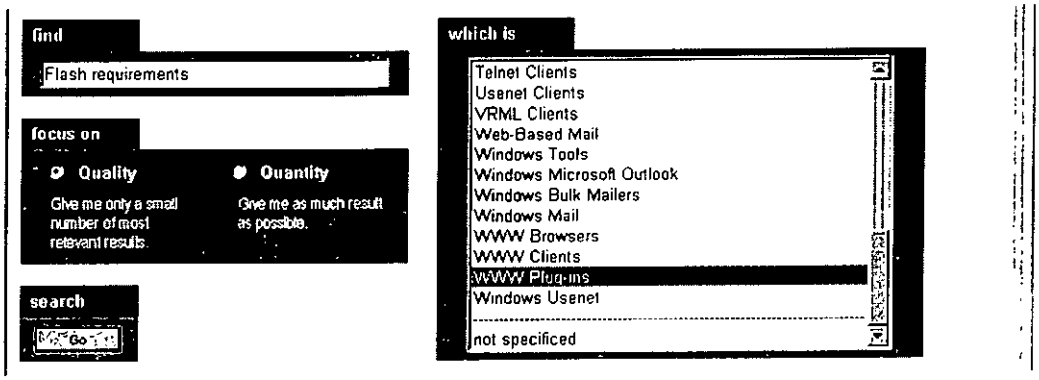

Figure 5.1b Sample search query using Amase

After entering the search keywords and selected suitable description, the user can

choose one of the search options inside the "*focus-on*" section. The "*Quality*" option indicates that the user wants only the most relevant documents. When this option is set, Amase will select only the top results from different specialty search engines for the users. This option can provide users a small set of highly accurate results in a very short process time. On the other hand, the "*Quantity*" option indicates that the user may want to collect all the search results. When this option is set, Amase will return all the search results collected from different specialty search engines to the users. This option can provide support to users when they want to do some broad review on a specific topic.

After selecting suitable options, the user can start the metasearch by pressing the "*Go*" button in the "*Search*" section. Figure 5.1c shows the search result listing the user received for the above example.

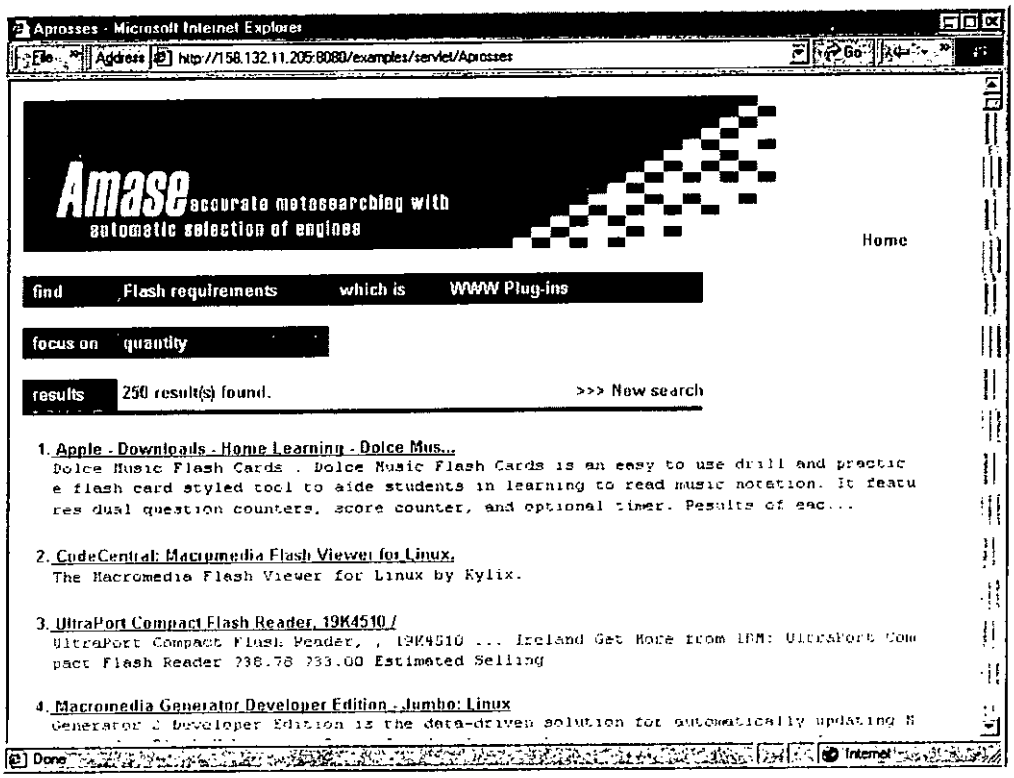

Figure 5.1c Returned search results

5.2 The Metasearching Process

After a search query is received from the user, Amase performs different processes before returning the metasearch results. Figure 5.2 illustrates the metasearching process inside Amase.

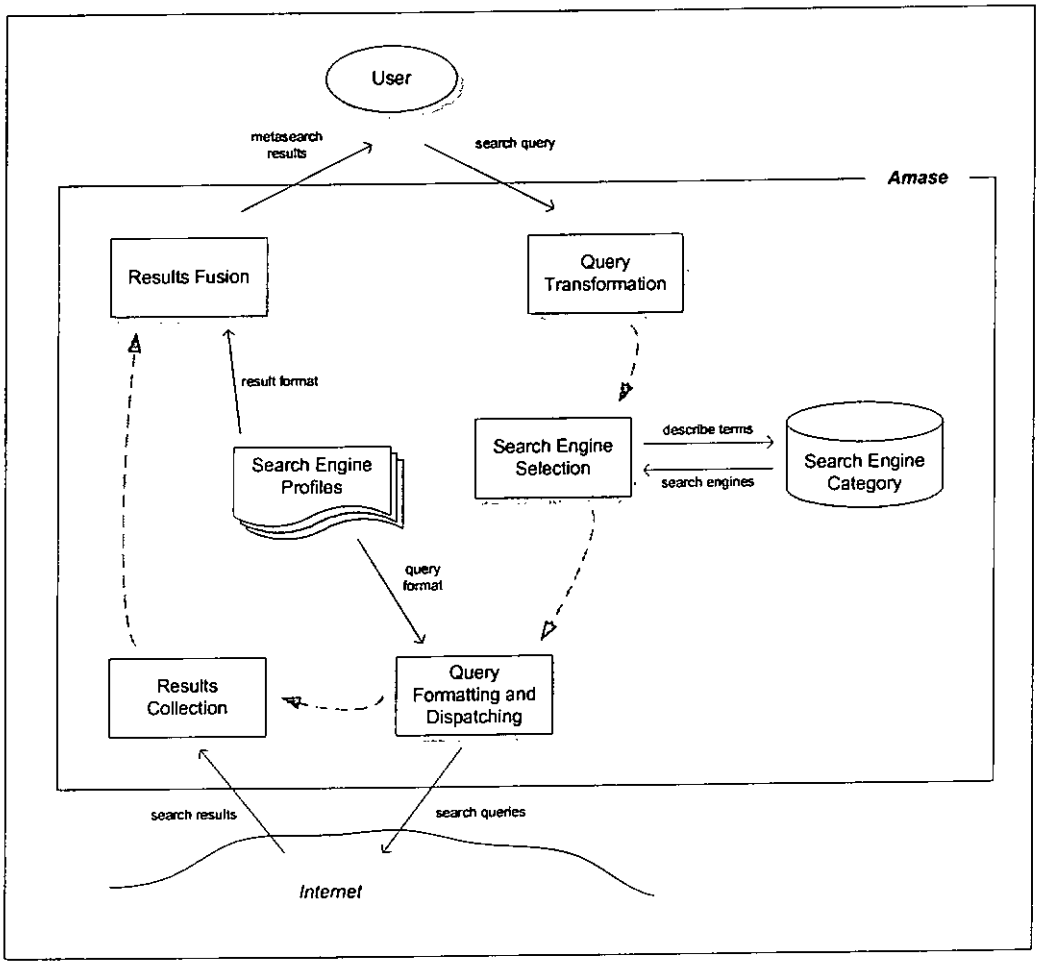

Figure 5.2a The metasearching process

As mentioned, users will submit their query to Amase in terms of a set of search terms plus a set of "descriptive terms". When a query is received, the *Query Transformation* process will first apply simple stopwords removal process to the given search terms. For example, the search query "*feature of Windows*" will be amended to "*feature Windows*": the article "*of*" is removed in order to increase the accuracy of the metasearching process. Afterwards, the *descriptive term* selected

by the user is used for the *Search Engine Selection* process, which will be explained in the next section in detail. After the relevant search engines are found, the original search query will be reformatted based on the search interface of each search engine as described in the corresponding *Search Engine Profile*. The *Query Formatting and Dispatching* process will then submit the reformatted queries to all the target search engines simultaneously. After all the search results are collected by the *Results Collection* process, the *Results Fusion* process will then extract the result data from the search result sets. Since each search engine has its own search result presentation format, the tailor-made procedures for extracting result data from each search engine are created and described in the corresponding profile. Finally, the result data will be combined into a single result list and returned to the user in the presentation format of Amase's.

5.2.1 Search Engine Selection

To select relevant search engines, a new querying approach is introduced. While general search engines require the user to submit only one set of search keywords, Amase requires the user to submit two sets. The first set named "*search terms*" is the search query as submitting to traditional search engines; the second, called "*descriptive terms*", is a set of descriptive keywords explaining what the search terms are. The objective of using two sets of keywords is to help users provide more information of their required information by stating the domain of the query being searched.

Based on the specialty search engine categories, a set of descriptive terms is manually created. Each sub-category is associated with the number of those terms. When performing a search engine selection, Amase tries to match the descriptive terms input by users to the predefined set of descriptive terms. When a match is

found, the sub-category associated with the matched terms will be used for the metasearching. Metasearching is done by sending the search terms to the search engines inside the selected category. Figure 5.2.1a illustrates an example of the selection process.

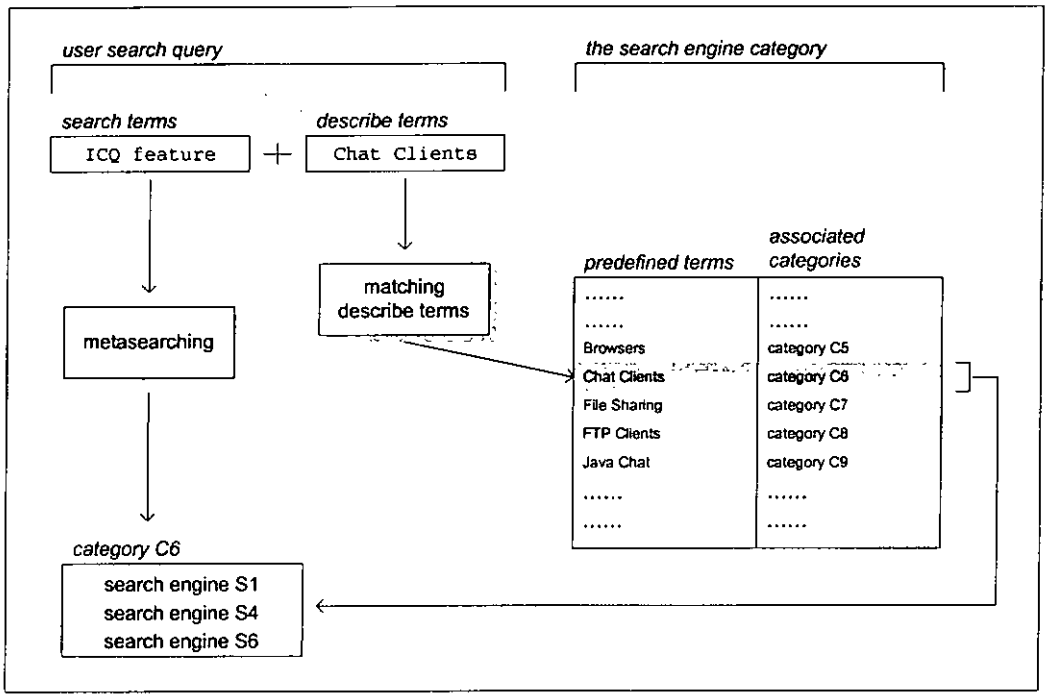

Figure 5.1.2a Selecting search engines based on the descriptive terms

Our initial idea is to let users specify both the search and descriptive terms; however, Amase was developed for experiment at purpose. As stated in the previous chapter, only 20 specialty search engines have been categorized into 34 categories and therefore, only 40 descriptive terms have been predefined. 34 of the terms are defined by combining the name of the selected categories with the name of their corresponding parent category. 6 additional terms are added to increase the understandability of the descriptive terms. For example, the term "*Java Chat*" was added since user may feel difficult to interpret the name of the category "*Chat Java*". When issuing queries, if users are allowed to perform metasearching in all domains, the descriptive terms given by users may hardly match those which have been

predefined. As a result, for experimental purpose, we list all the predefined terms for the users to select instead of asking them to input them so as to restrict the metasearch within the domains covered by the produced search engine category.

5.2.2 Search Result Fusion

As mentioned, the final metasearch result is a combined result list that merges different search results returned by different specialty search engines. Each result list returned by a specialty search engine is actually a list of hyperlinks pointed to the set of target document. Three kinds of data for each hyperlink are extracted and recorded in the *Result Fusion* process: (1) the URL of the hyperlink, (2) the title of the document and (3) the summary of the document, which may usually be the first few lines of the content of the document.

Two types of policy are defined for combining the extracted search results. As explained in previous sections, when submitting query to Amase, one of the two types of options inside the "*focus on*" section can be selected, and each option in fact represents one of the polices. When the "*Quality*" option is set, Amase will make use of only the top 10% of the entire search results and combines them in a round-robin manner. This option provides users the shortest processing time with the highest quality results. Alternatively, when the "*Quantity*" option is chosen, Amase combines all the search results in the same order as are returned from the specialty search engines. This option allows the user to collect all the search results with a trade-off of a lengthy process time. Figure 5.2.2a illustrates an example of the fusion policies.

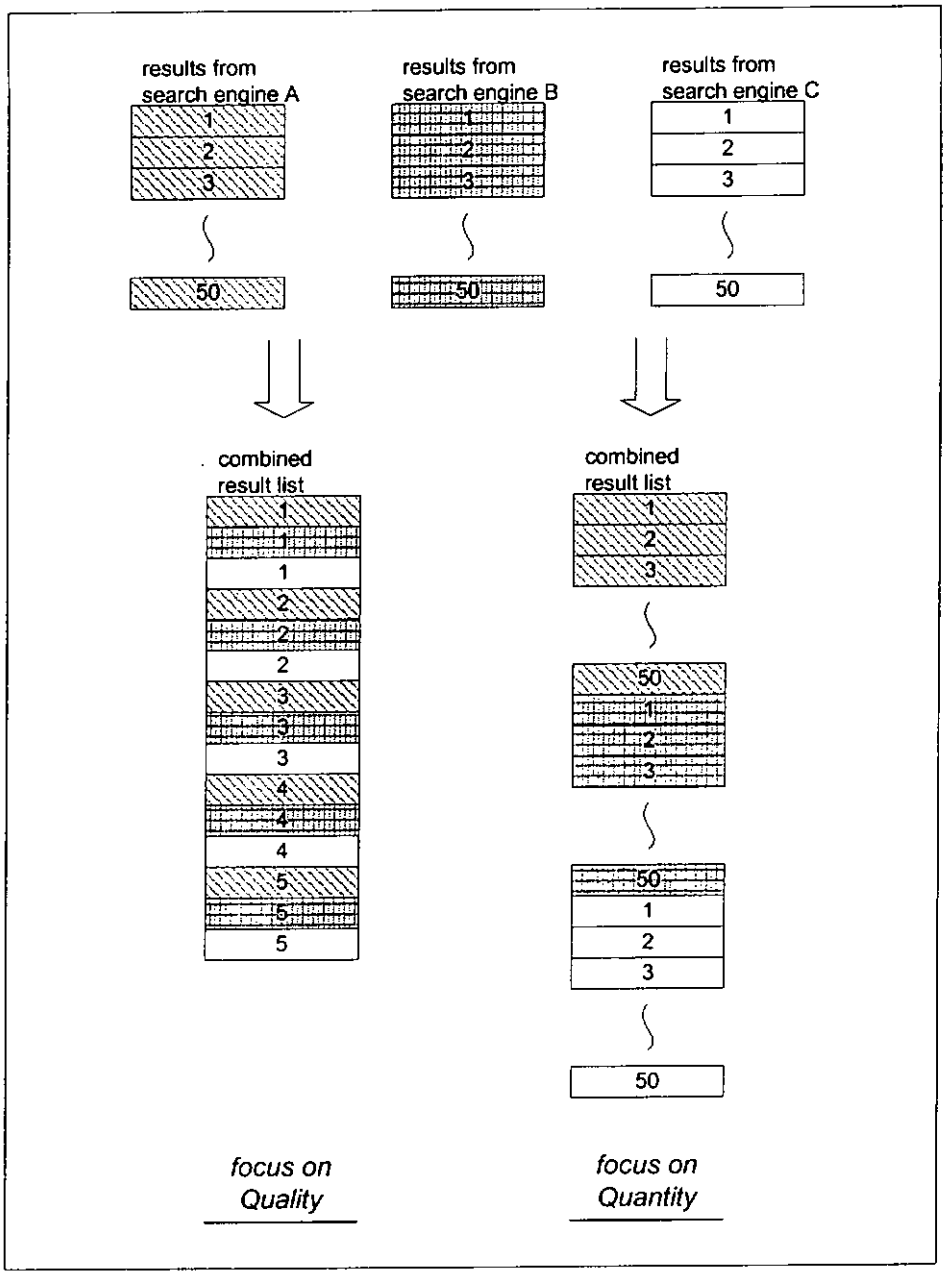

Figure 5.2.2a Two approaches in combining search results

5.3 Experiments

To evaluate the effectiveness of finding specialty information using Amase, sample searches were made using Amase and a number of general search engines. The objectives of this experiment are firstly, to demonstrate that most search results returned by specialty search engines do not overlap with those returned by general search engines and secondly, organizing specialty search engines into a hierarchical

structure can help metasearch engine in selecting suitable source for metasearching. Figure 5.3a listed the queries and search sources selected for the experiment.

Query	Search Terms	Descriptive Terms
1	check mail	mail clients
2	junk mail	mail clients
3	anti spam	mail clients
4	virus	mail clients
5	mail broadcast	mail clients
6	multimedia chat	chat clients
7	voice messages	chat clients
8	web-based	chat clients
9	random chat	chat clients
10	instant messaging	chat clients

	Search Engines
1	Amase
2	Google (restricted)
3	Google (normal)
4	ProFusion
5	Manual Select

Figure 5.3a Search queries and search engines used for the experiment

As shown in the figure, one of the general search engines we used was Google [Google03], which is one of the most successful general search engines on the Internet. Google was chosen not only because of its reputation, but because of its ability to restrict search to a sub-categories of the Open Directory. Amase requires the user to submit a query with one set of search terms and one set of descriptive terms, which is used to implicitly specify the search domains. Using the same set of search terms to query another general search engine may lead to incomparable results, as the search results returned by the general search engine belong to all kinds of domains. Therefore, Google was selected in experiments and two types of search were made. First, we restricted Google to search within the same search domain as Amase and second, we using Google to perform a normal search using the same set of queries.

The second general search engine used is Profusion, which is a metasearch engine that metasearch 13 other general search engines excluding Google. In addition, we have also manually selected one specialty search engine, which was the most relevant one to the queries, out of the 20 engines used by Amase for the experiment.

The queries were sent to the listed search sources and the numbers of search results returned are recorded and listed in Figure 5.3b:

Query	Manual Selection	Amase	Google (restricted)	Google (normal)	ProFusion
1	167,204	488,769	87	1,730,000	235
2	78,436	135,566	34	123,000	208
3	2,089	7,089	52	96,600	204
4	27,442	54,910	26	400,000	193
5	81,279	141,553	7	360,000	208
6	13,446	42,560	4	178,000	203
7	172,963	470,974	10	63,700	183
8	15,597	111,479	19	115,000	203
9	10,797	27,133	22	82,300	205
10	55,811	279,557	56	83,200	212

Figure 5.3b Number of search results returned by search engines

As shown in the above figure, Amase was generally able to return a larger number of search results than the restricted Google, ProFusion and the manually selected specialty search engine. For the first 100 search results of each search source, we discovered that none of the search results returned by Amase overlapped with those returned by Google (both restricted and normal) or ProFusion. Certainly, the

number of search results is not the only factor for analyzing the quality of a searching method. Therefore, we analyzed the search results returned manually and listed the number of relevant results out of the first 100 results of each search sources in Figure 5.3c:

Query	Manual Selection	Amase	Google (restricted)	Google (normal)	ProFusion
1	17	44	83	79	57
2	40	73	34	99	97
3	22	77	51	98	52
4	30	62	22	93	69
5	26	49	5	77	55
6	4	13	2	29	16
7	25	60	4	92	63
8	8	46	11	79	57
9	3	10	0	27	11
10	18	71	50	98	74

Figure 5.3c Number of relevant search results returned by search engines

As shown in the above figure, Amase was able to return more numbers of relevant results than the manual selected specialty search engine and the restricted Google. Moreover, the numbers of relevant results of Amase are similar to that of ProFusion. However, as shown in figure 5.3b, the total number of search results of Amase are a lot larger than that of ProFusion, therefore when counting all the results, we believe that Amase is able to return a larger number of relevant results than ProFusion.

Additionally, the quality of the results returned by Amase is better than that given by the manual selected one. It demonstrated that by using a hierarchical structure to

organize specialty search engines can help metasearch engine in selecting relevant specialty search engines for metasearching.

Although the experimental results show that Google outperforms other searching methods, the results by Amase were not overlap with Google and therefore, we believe metasearching specialty search engines can help access quality specialty information that have not been indexed by general search engines.

As a final point, the search results returned from Amase confirmed that the coverage of general search engines was not sufficient; resources indexed by specialty search engines were not fully indexed by those general search engines. By including metasearching of specialty search engines, the coverage on the Internet would be increased and users could obtain more significant search results. Moreover, the search engine selection problem of metasearch engine could be alleviated by the search engine category automatically built by the proposed algorithm, and resulting in more accurate and effective metasearching.

5.4 Summary

In this chapter, we presented the experimental metasearch engine Amase to demonstrate how metasearch engine can exploit the search engine category to obtain relevant and distinguish search results. We described the user interfaces of Amase, explained how queries could be formulated, and discussed the internal architecture in details. Search results from sample queries showed that Amase was able to discover hidden Web resources effectively, promising higher effectiveness of the proposed search engine category.

Chapter 6

Conclusions and Future Work

To conclude, this thesis presents our researches in two main aspects: we have (1) demonstrated metasearching of specialty search engines can help discover precious resources on the Internet, and (2) developed a search engine categorization algorithm for constructing a hierarchical search engine category to deal with the search engine selection problem of metasearch engine.

Based on careful review of works done by other researchers on the selection methods, we proposed and developed a categorization algorithm which can automatically categorize specialty search engines into a hierarchical structure. We discovered that only a small number of probe queries are needed to acquire enough document samples from specialty search engines. Returned documents from those search engines precisely reflect the context of the documents they contain. Document samples therefore can be used as representatives of the search engines for categorization processes. When determining the relevancies between specialty search engines and sub-categories, we found that including the hierarchical structure information of the sub-categories can significantly improve the categorization accuracy.

To evaluate the category produced by our algorithm, we invited a number of computer experts to help construct a human-judged search engine category. By comparing the former category to the human-judged one, we found that our algorithm is able to categorize specialty search engines, and construct a sensible

search engine category similar to that built by the human.

Lastly, we have developed "Amase", an experimental metasearch engine prototype which exploits the proposed search engine category. The search results returned by Amase are encouraging. They show that general search engine misses most of the resources available in specialty search engines, and the proposed search engine category is valuable in selecting relevant specialty search engines for metasearching.

6.1 Future Work

As the search results from metasearching specialty search engines do not overlap with those from general search engines, we believe general search engines mainly target for casual users who aim to search for generic information. Therefore by complementing general search engines by the function of metasearching for specialty search engines, more effective searching can be made.

Particularly, as one major difficulty on developing the specialty search engines category is the expensive labor cost needed to discover existing specialty search engines, we believe that by applying additional functions to the software robots that are used by general search engines can significantly reduce the cost. Since general search engines use robots to traverse from Web page to Web page and we believe that specialty search engines which embedded inside Web pages have numbers of common elements, methods for identifying whether a page contains a specialty search engine or not can be developed and used to enhance existing robots. In other words, when a robot arrive a particular page, it not only can index that page for generic searching purpose but also identify whether there are specialty search interface, and if there is one exist it can be indexed and categorized for the metasearching purpose.

In addition, general searching and specialty searching can be provided simultaneously by, for example: (1) including general search engines into the proposed search engine category; or (2) sending the metasearch query to both specialty and general search engines. However, a suitable method for combining the search results must be employed or developed in order to have a quality list of search results.

Moreover, as general search engine contains a numbers of processes in the search cycle, the proposed categorization algorithm and the metasearch engine involve different complex processes, and each of them directs different research issues. Therefore, a number of possible improvements and extensions of this research can be made, and we summarize some of them as follows:

- As the proposed categorization algorithm is able to categorize search engines into a hierarchical structure adapted from the ODP directory, we believe that the algorithm may also be used to categorize different text collections or databases into other kinds of hierarchical structures, taxonomies or classifications.

- To identify the relevancy of a Web page, which in the format of HTML file, the proposed approach computes the corresponding relevancy score based on the raw frequencies of different terms occurring in the document. We believe that the relevancy score can be enhanced to reflect the actual relevancy by including the HTML structural information of the document. For example, there are two documents A and B containing an equal number of term frequency, and the terms in A and B are embedded in the "Title" tag and the "P" tag in the file accordingly. We believe that the terms occur in A is more significant since they hold a weightier place in the HTML file, and as a result, document A

should be assigned a higher ranking.

- When querying specialty search engines during the document sampling and metasearch processes, the quality of the search results can be improved by utilizing the advanced search options of each search engine. We believe the more search options utilized, the more search options can be provided by the metasearch engine; and therefore, the flexibility of the search functions provided by the metasearch engine will increase.

- When analyzing different specialty search engines, we recognize that most of them share some common characteristics in terms of their HTML structure. We believe that if more specialty search engines can be analyzed, different classification rules can be derived, and automatic discovery of specialty search engine maybe possible. If specialty search engines can be discovered automatically, using the robot approach for example, the cost of the categorization process and the update process of the produced category can effectively cut.

- As the developed metasearch engine provides only simple search functions, some potential advantages of the search engine category may not have been discovered yet. For instance, we believe applying other advanced search engine abilities, like PageRank [Google03], to the metasearch engine, may improve the search quality. Finally, the metasearch engine can determine different approaches in combining the metasearch results in order to enhance the overall performance of the metasearching.

6.2 Publication

Segments of this research have been presented in the following publication:

Jacky K. H. shiu, Stephen C. F. Chan and Korris F. L. Chung "Accessing Hidden Web Documents by Metasearching a Directory of Specialty Search Engines". To appear in *Proceedings of the 3rd International Workshop on Databases in Networked Information Systems*, Japan, 22-24 September 2003.

Jacky K. H. Shiu, Stephen C. F. Chan and Korris F. L. Chung "Developing a Directory of Search Engines for Metasearching". To appear in *Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning*, Hong Kong, 21-23 March 2003.

Appendix A

Figures A1 – A34 present comparisons between manual-judged rankings (*manual*) and the rankings derived by the proposed algorithm (*R*) for all the selected categories.

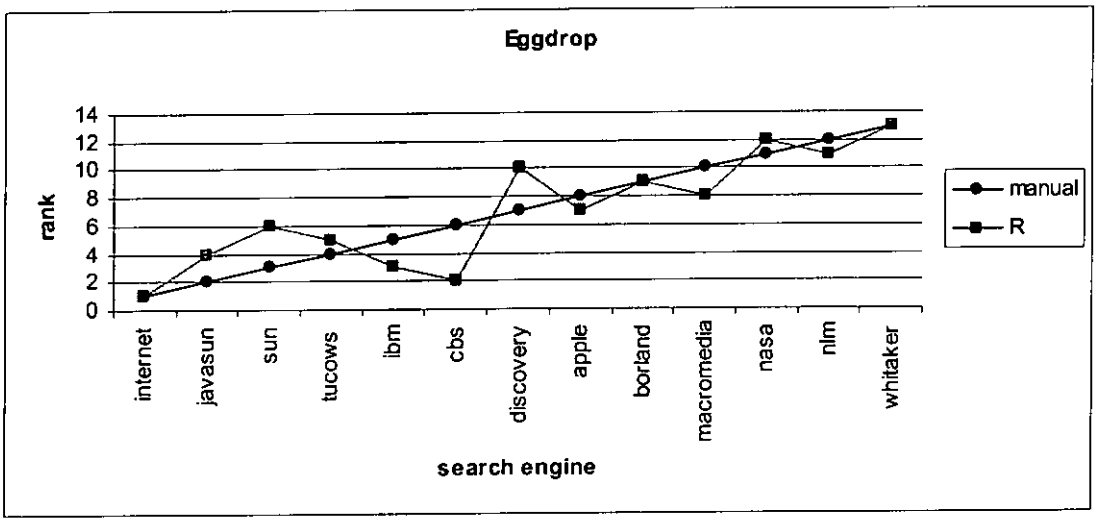

Figure A1 Category Eggdrop

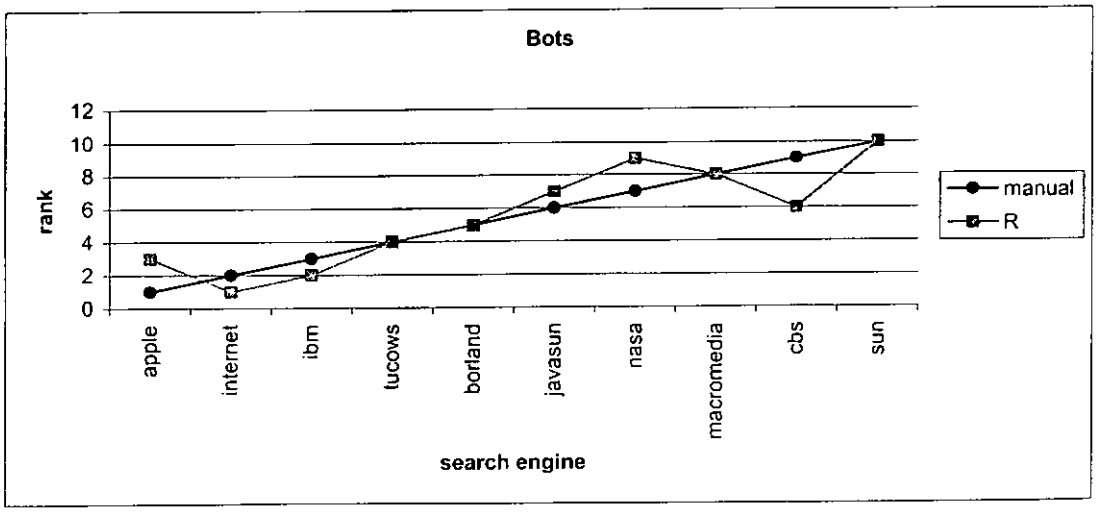

Figure A2 Category Bots

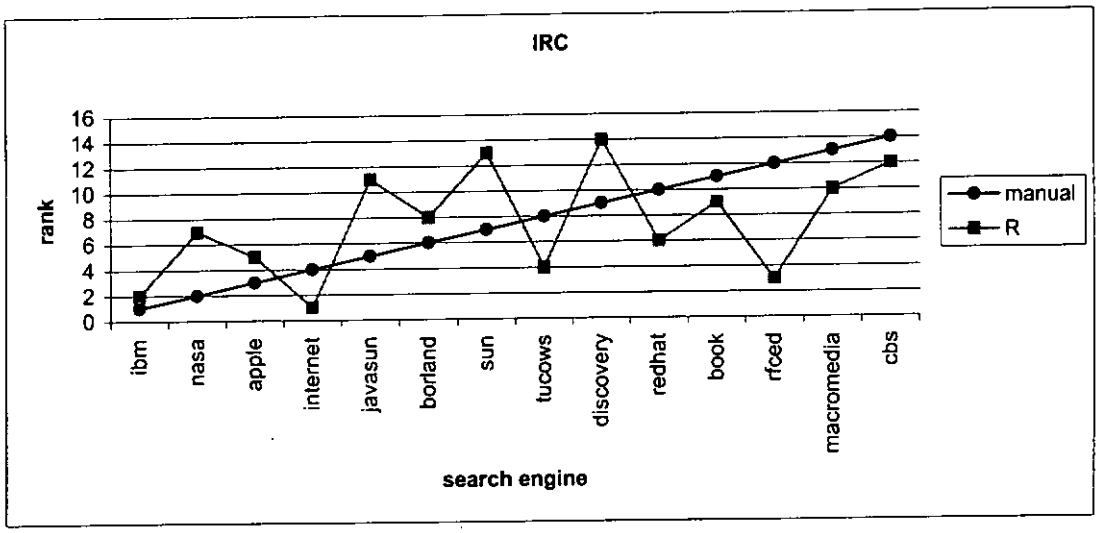

Figure A3 Category IRC

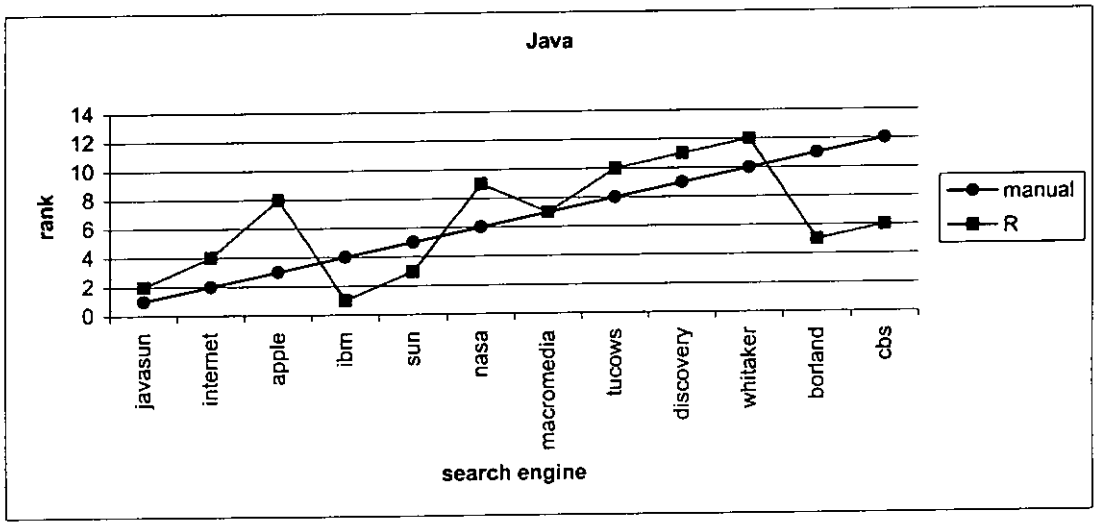

Figure A4 Category Java

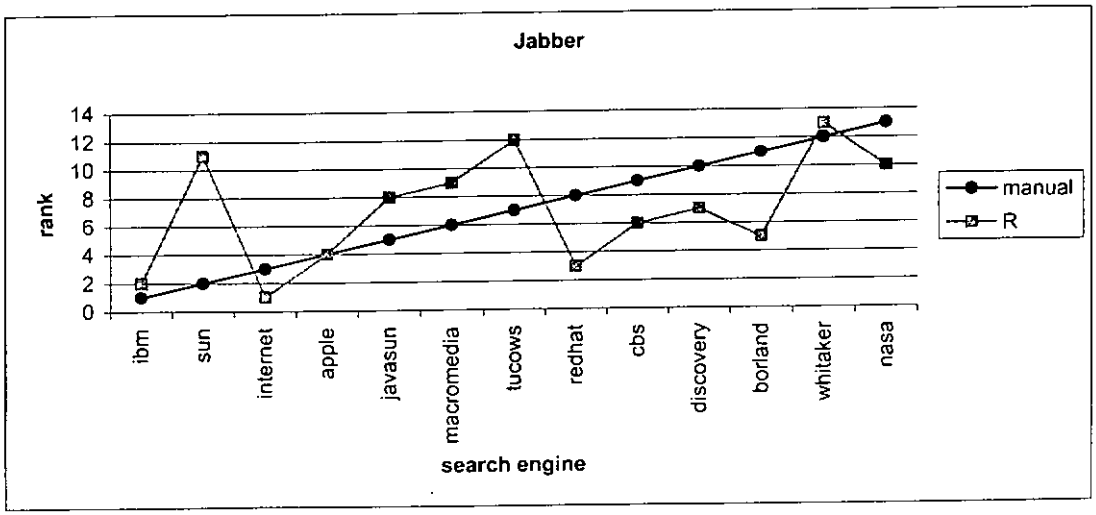

Figure A5 Category Jabber

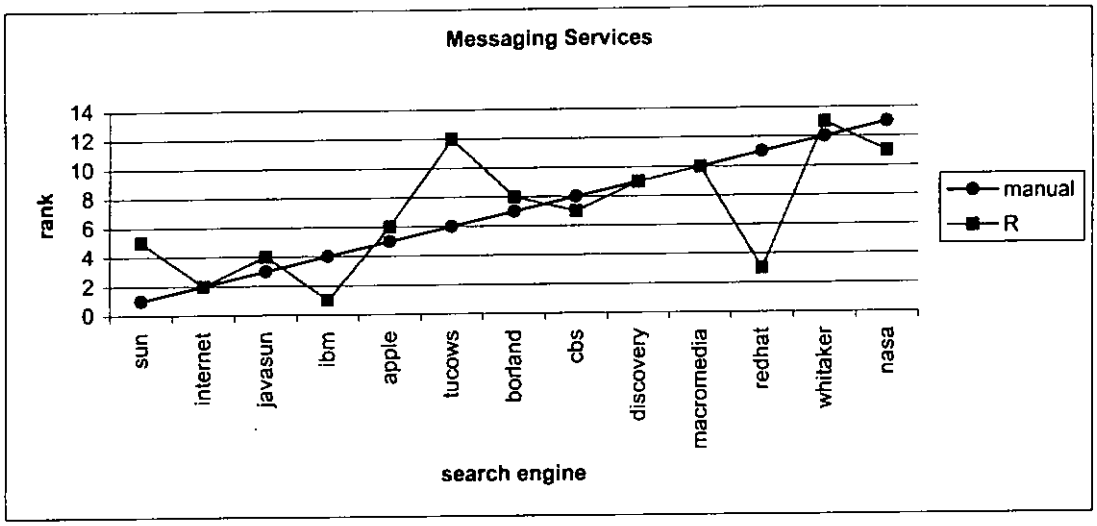

Figure A6 Category Messaging Services

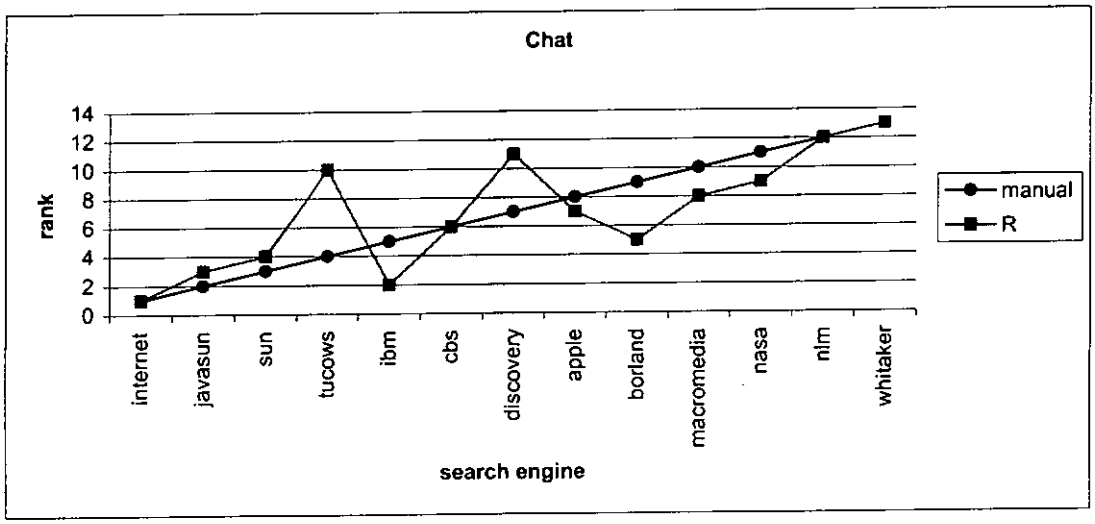

Figure A7 Category Chat

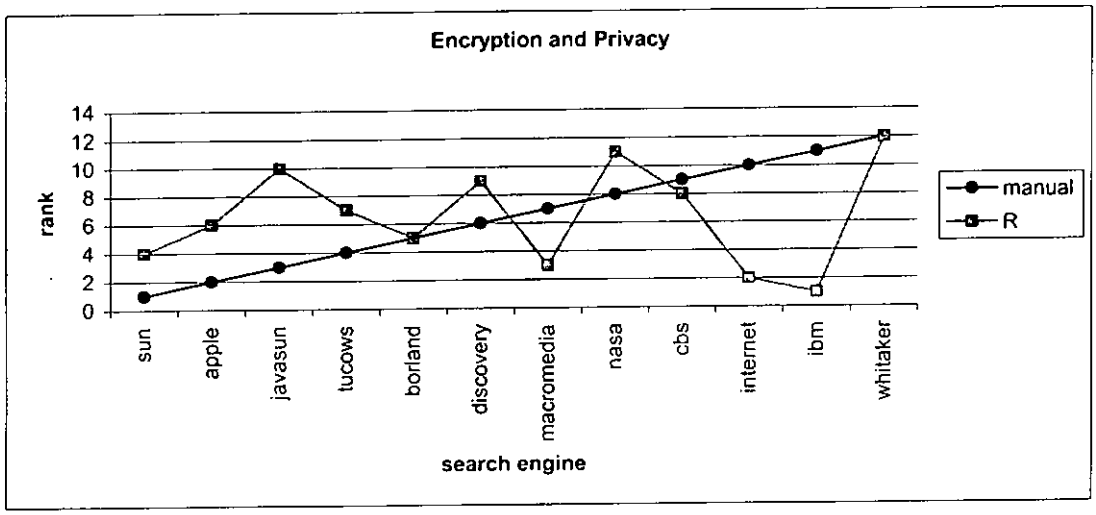

Figure A8 Category Encryption and Privacy

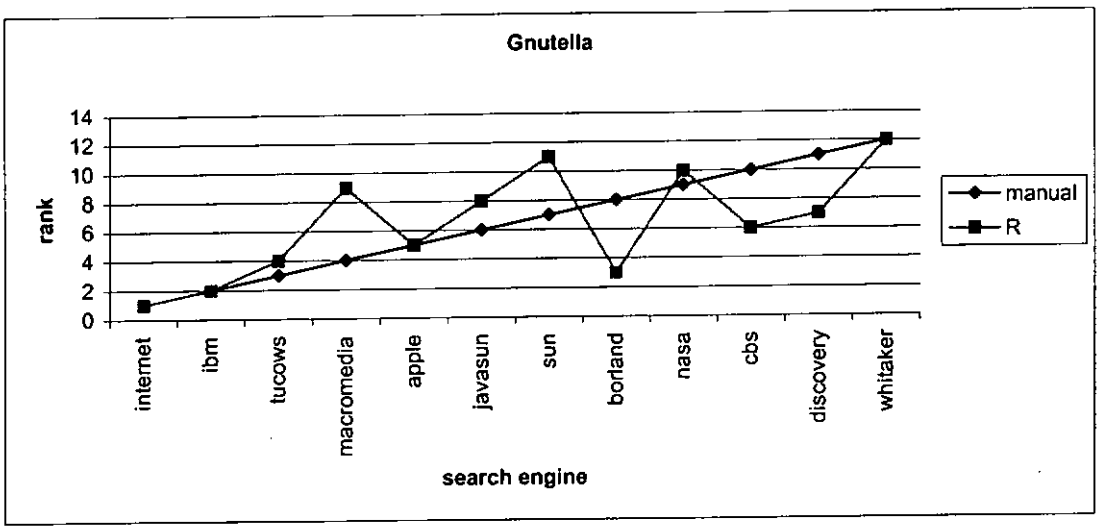

Figure A9 Category Gnutella

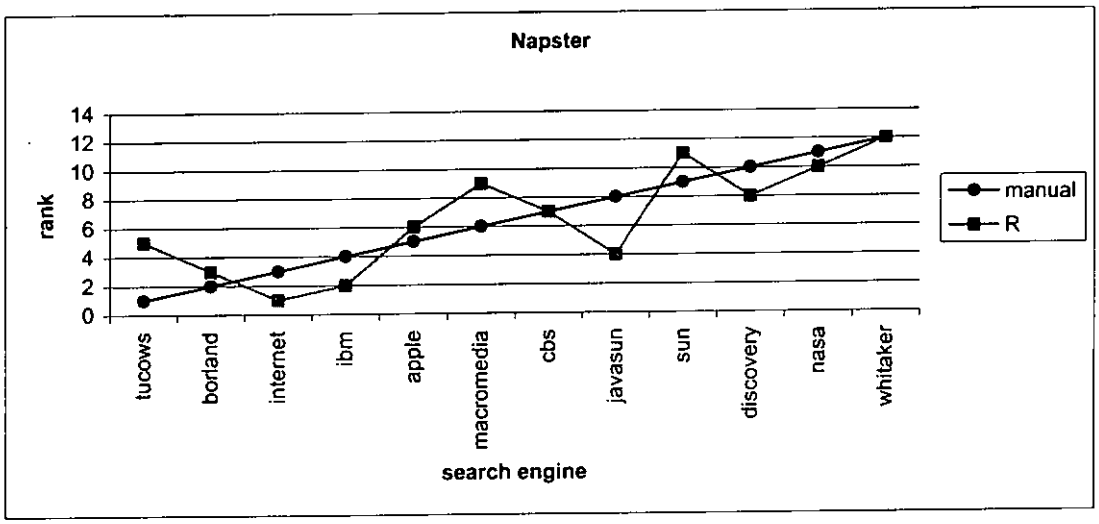

Figure A10 Category Napster

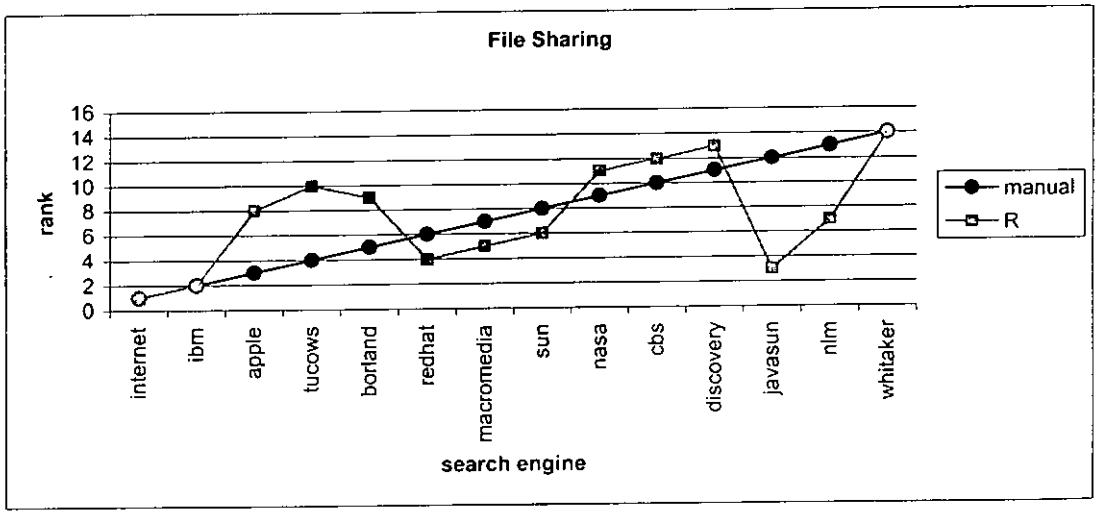

Figure A11 Category File Sharing

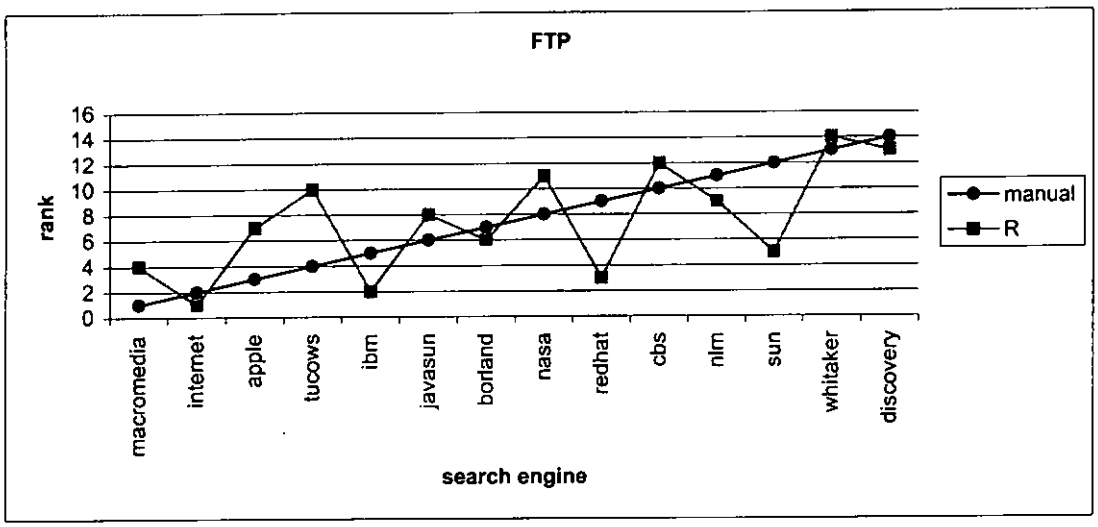

Figure A12 Category FTP

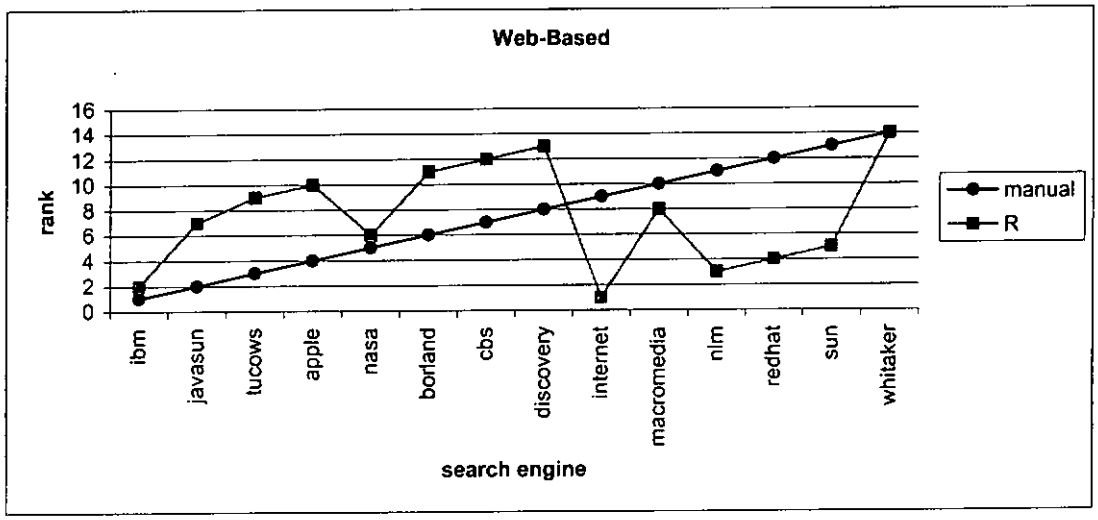

Figure A13 Category Web-Based

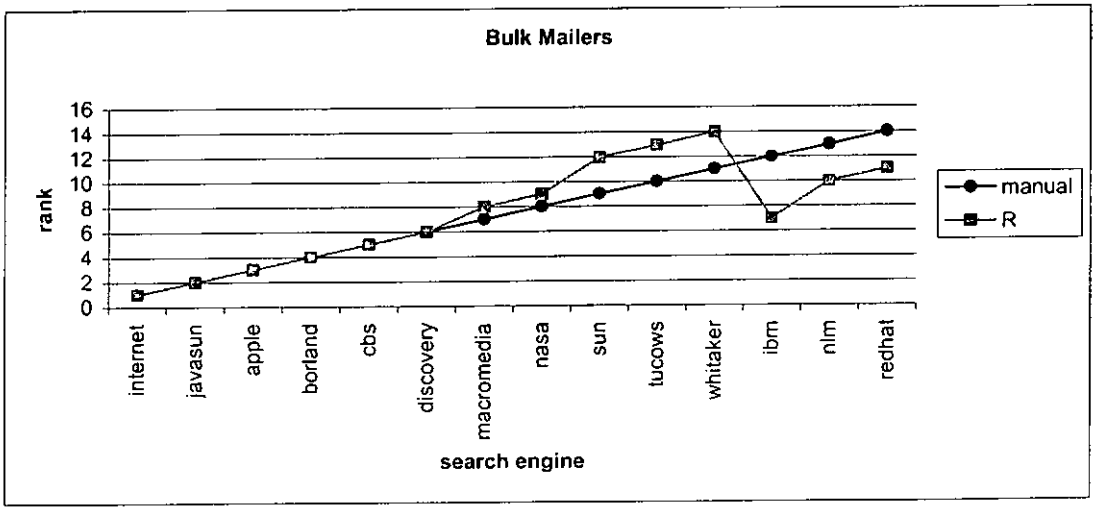

Figure A14 Category Bulk Mailers

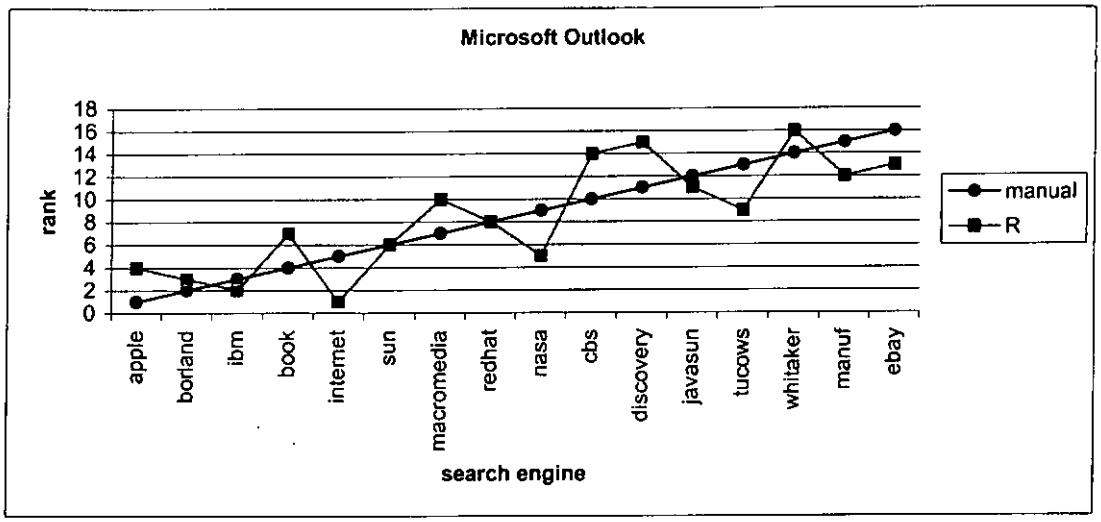

Figure A15 Category Microsoft Outlook

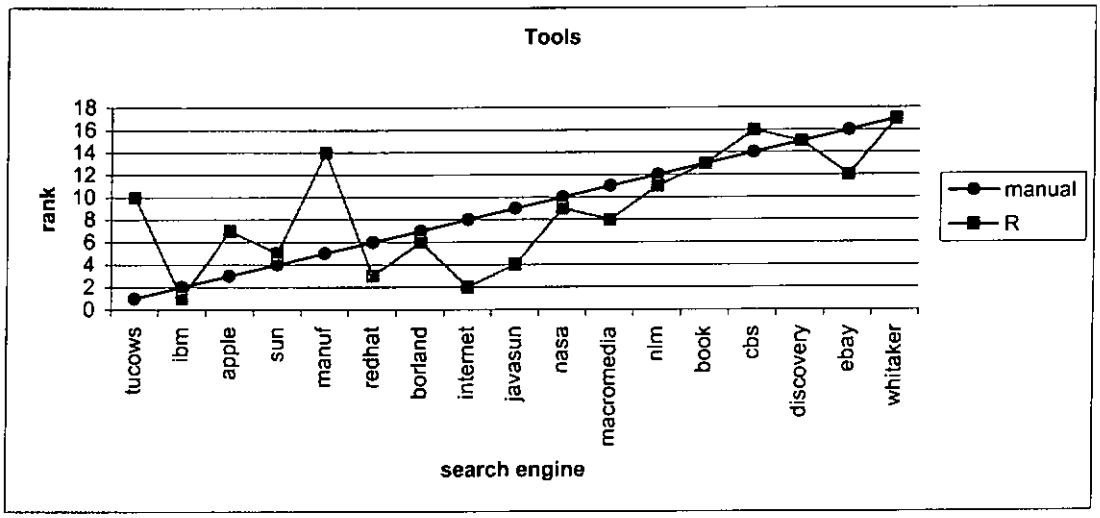

Figure A16 Category Tools

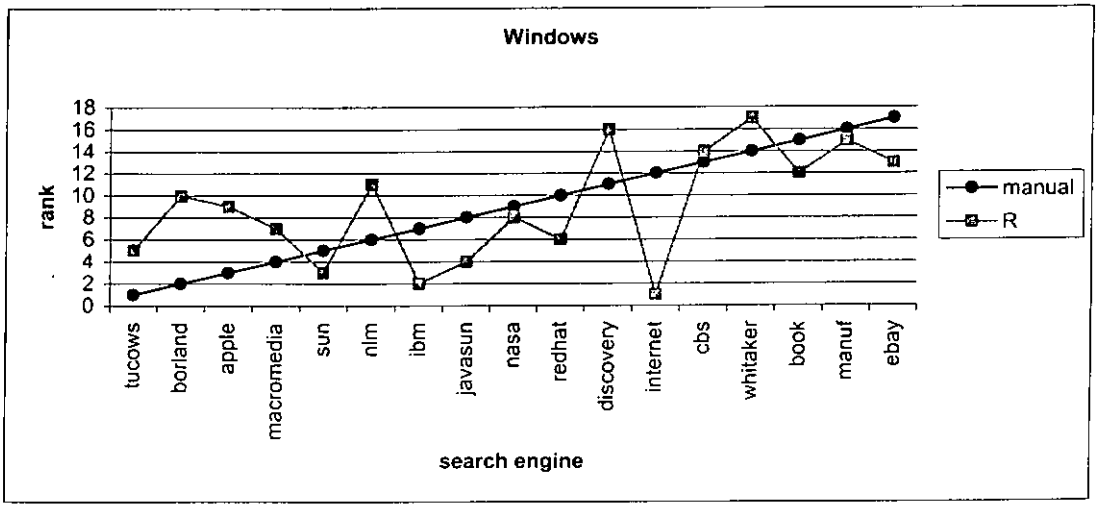

Figure A17 Category Windows

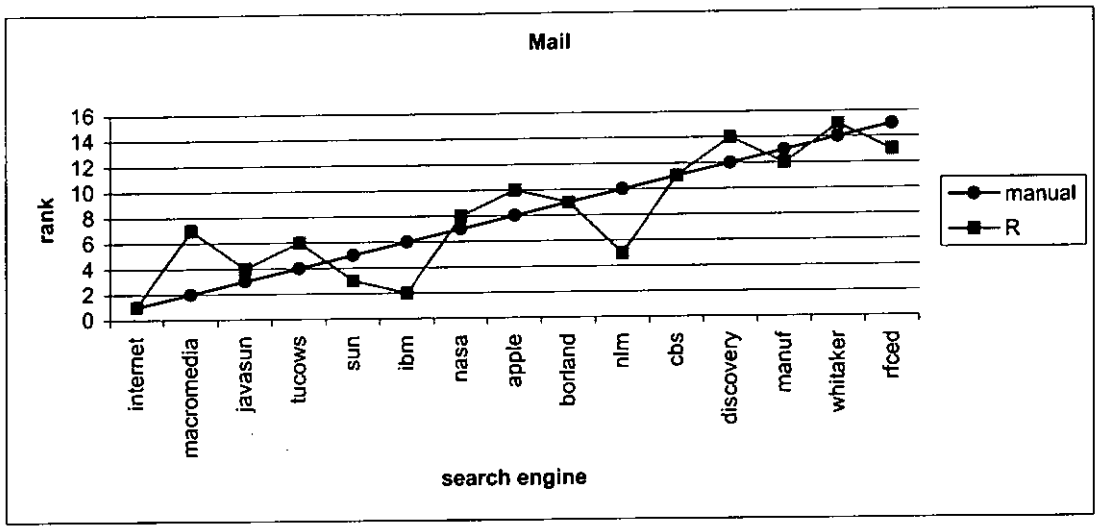

Figure A18 Category Mail

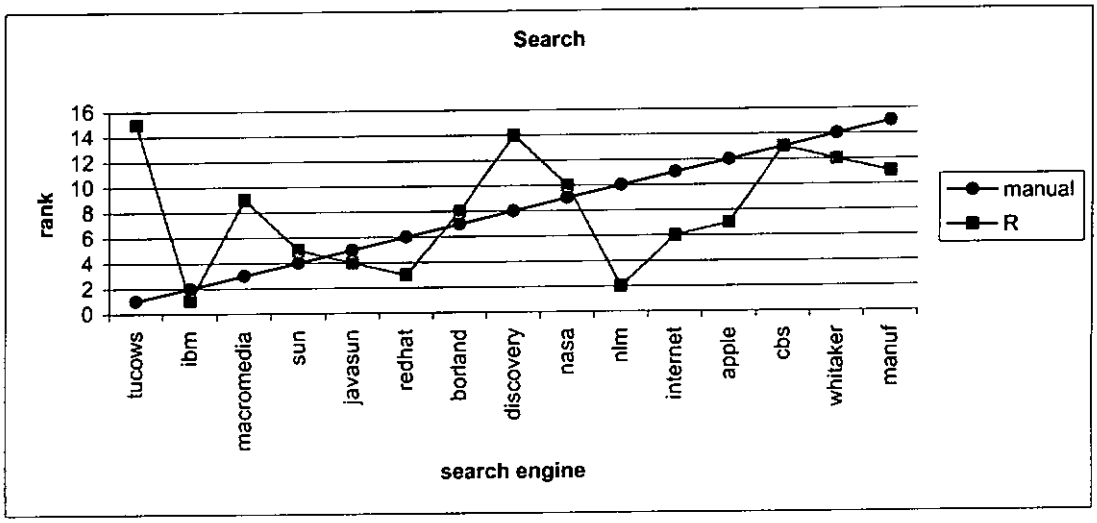

Figure A19 Category Search

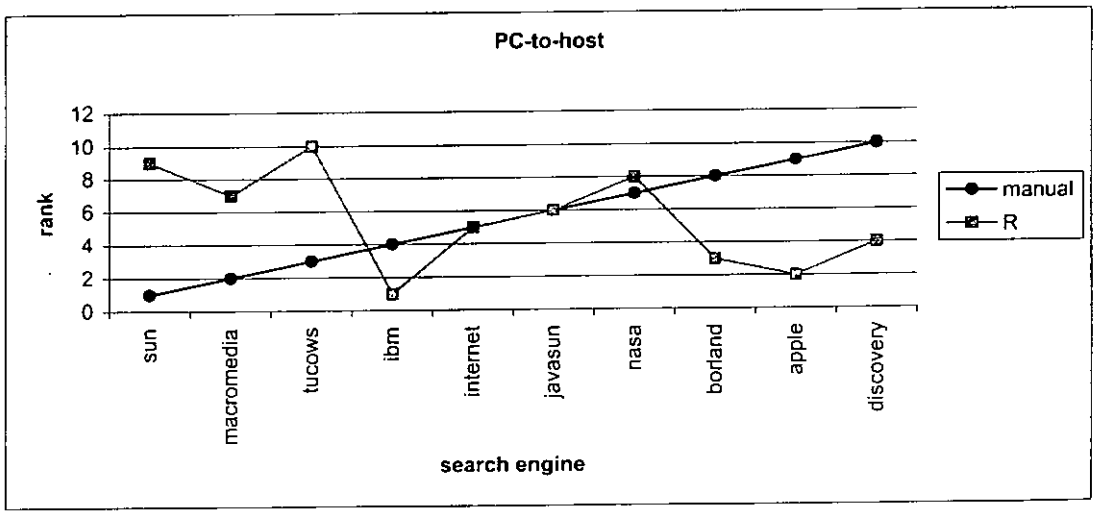

Figure A20 Category PC-to-host

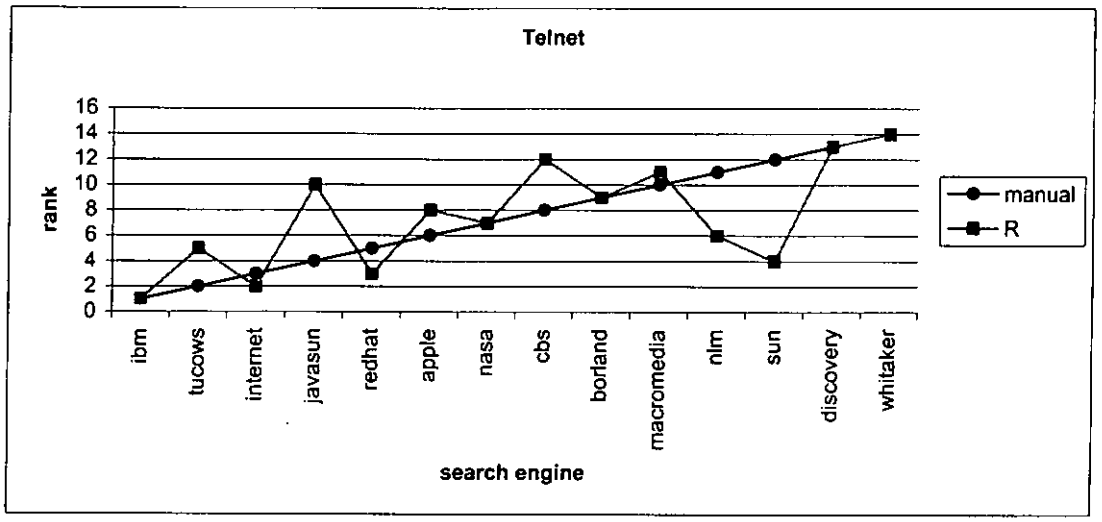

Figure A21 Category Telnet

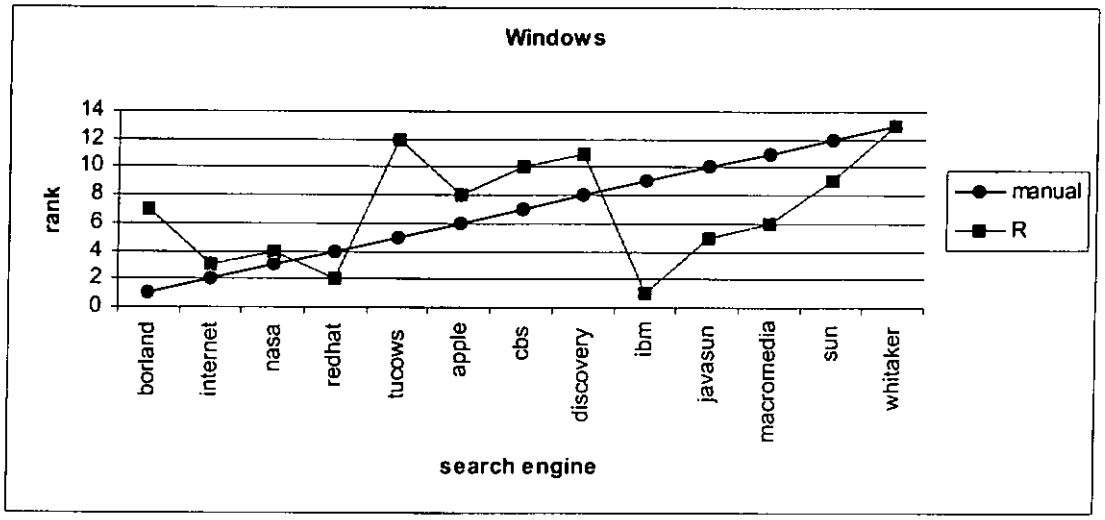

Figure A22 Category Windows

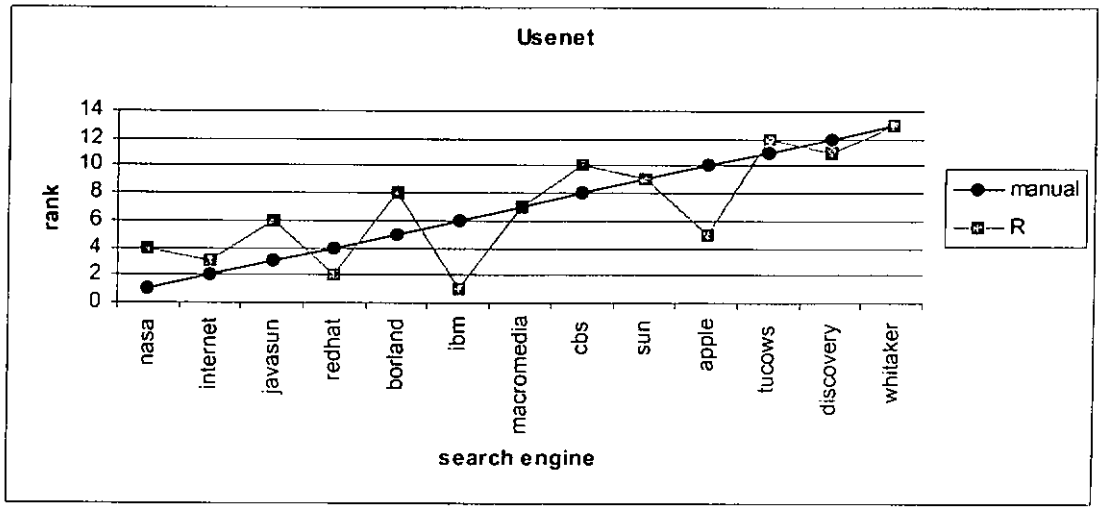

Figure A23 Category Usenet

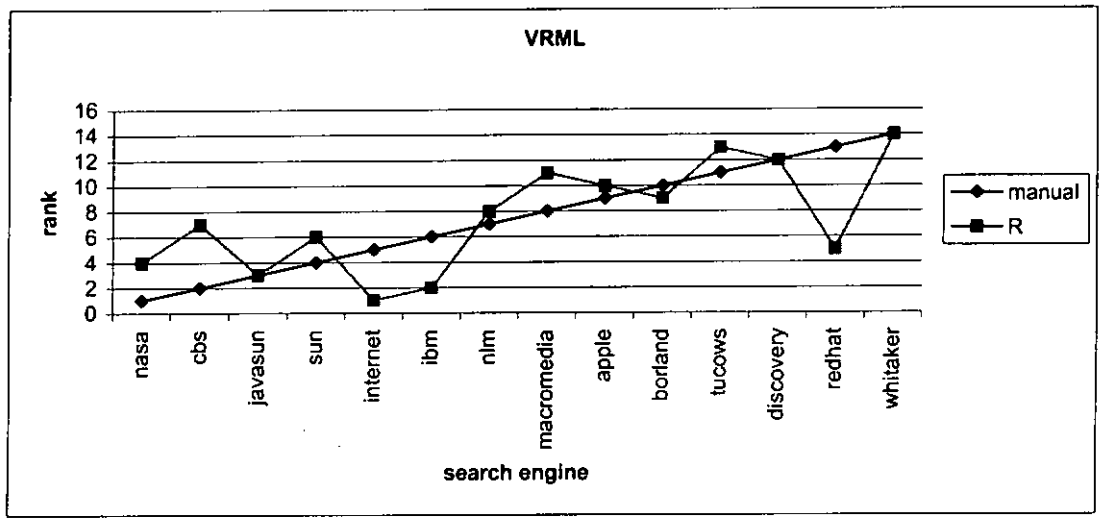

Figure A24 Category VRML

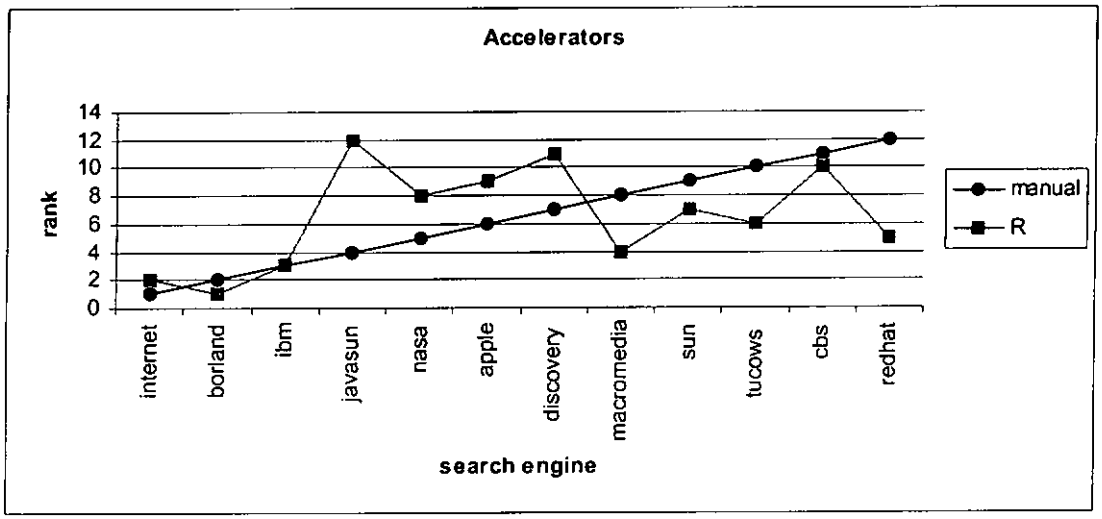

Figure A25 Category Accelerators

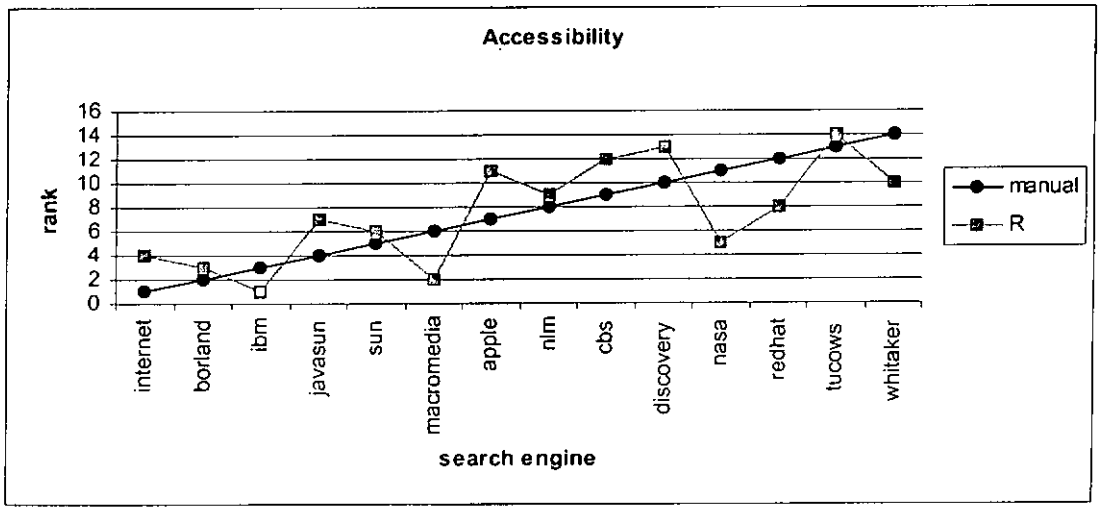

Figure A26 Category Accessibility

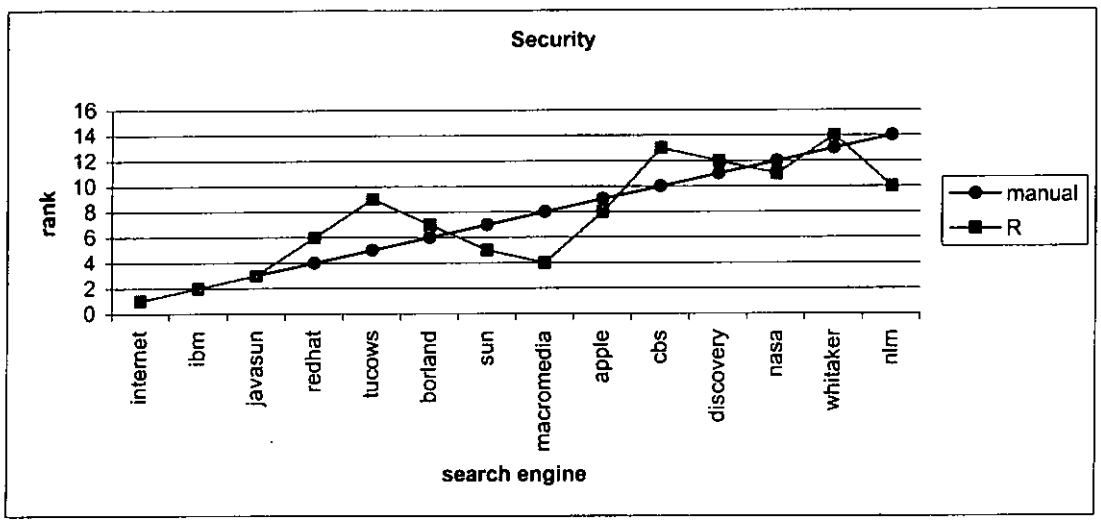

Figure A27 Category Security

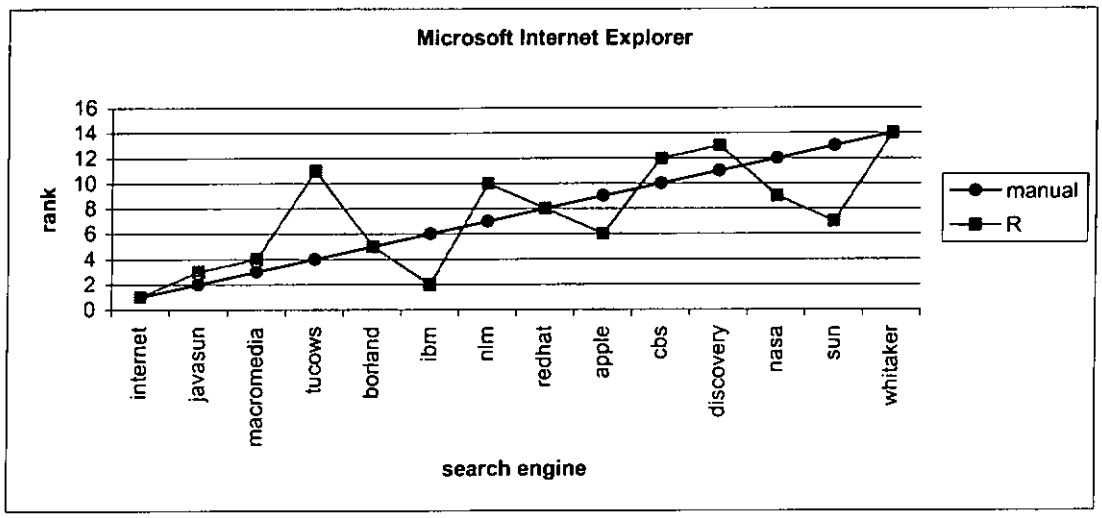

Figure A28 Category Microsoft Internet Explorer

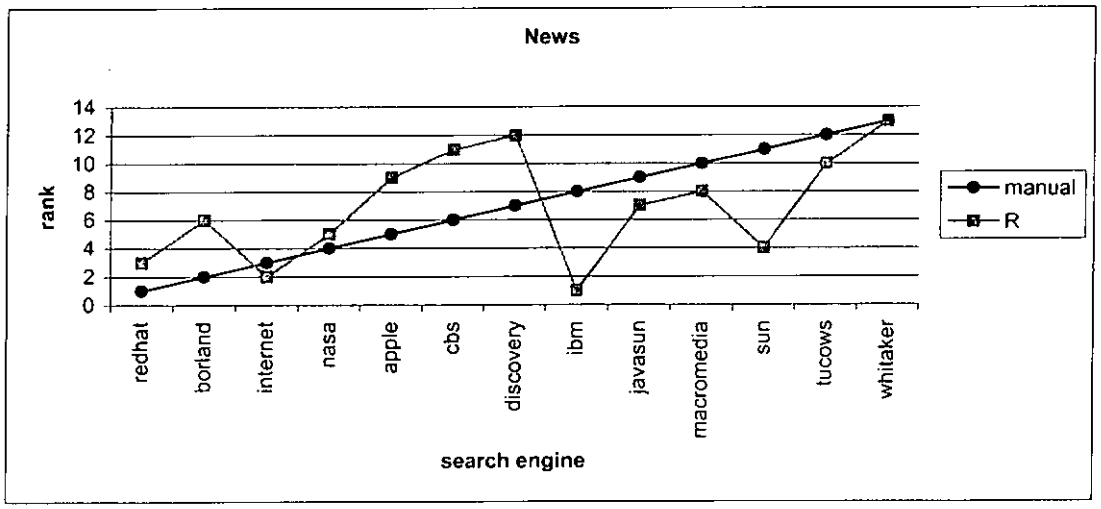

Figure A29 Category News

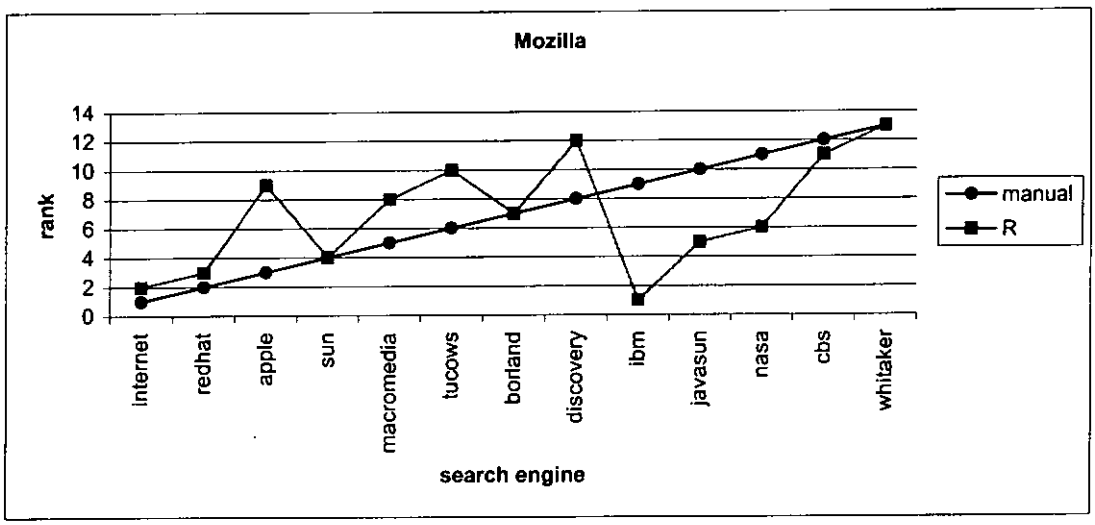

Figure A30 Category Mozilla

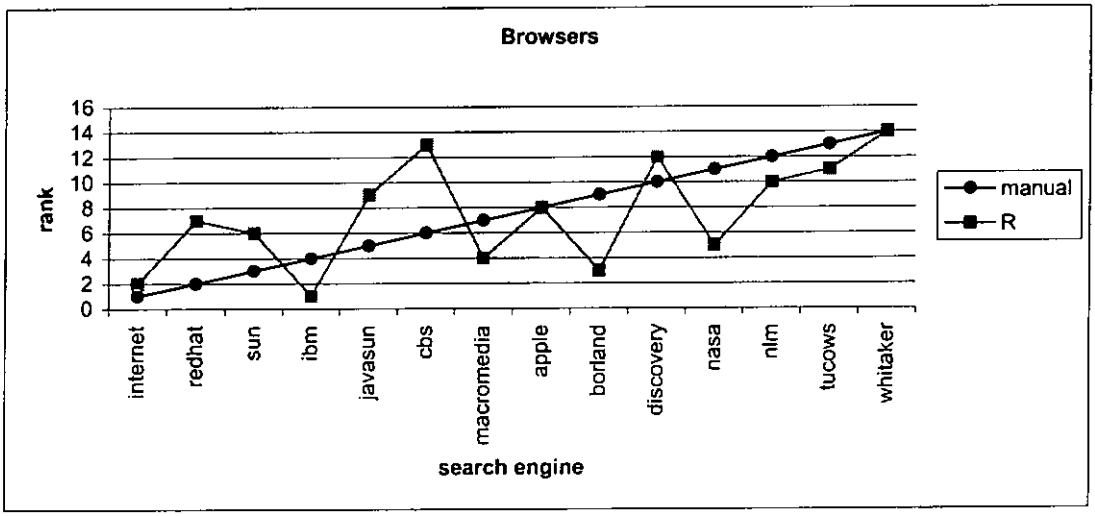

Figure A31 Category Browsers

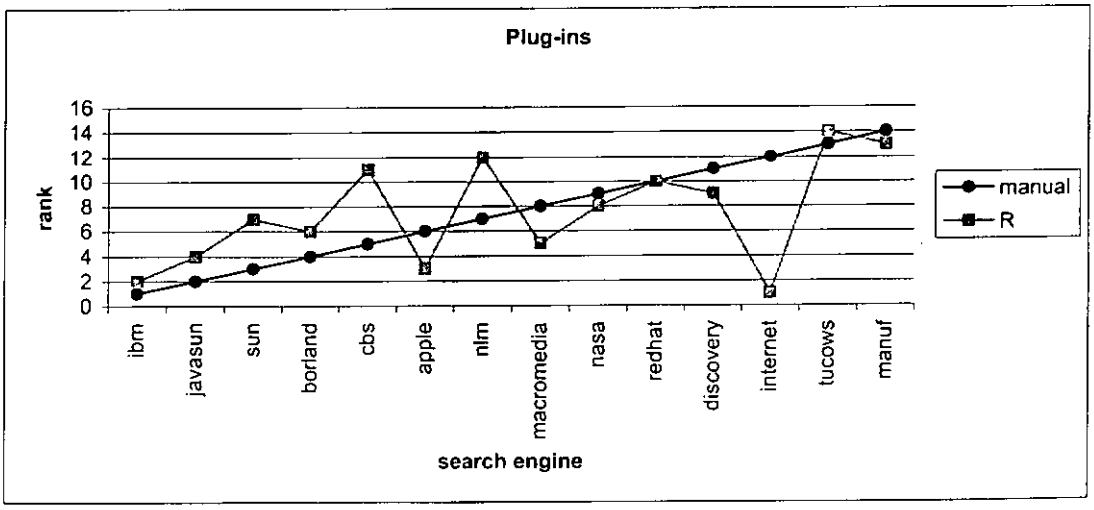

Figure A32 Category Plug-ins

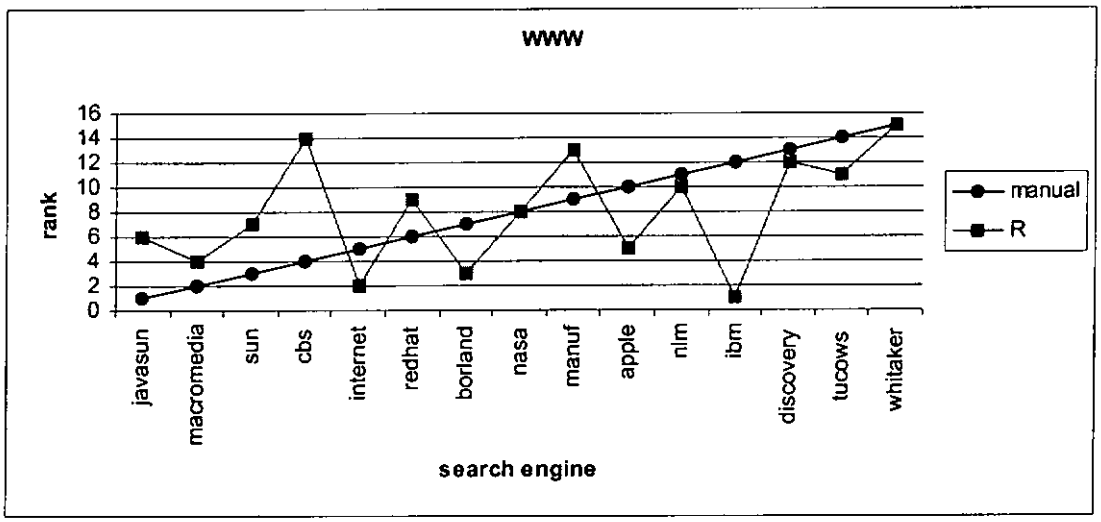

Figure A33 Category WWW

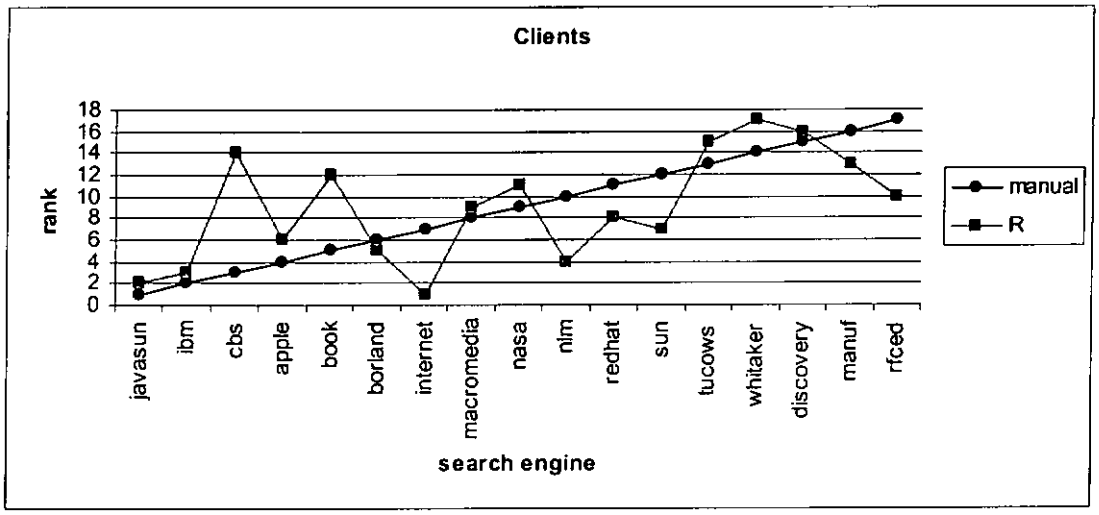

Figure A34 Category Clients

Bibliography

Agichtein01 — E. Agichtein, S. Lawrence and L. Gravano "Learning search engine specific query transformations for question answering". *Proceedings of the 10th World Wide Web Conference*, Hong Kong, 1-5 May 2001, pp.169-178 (2001)

AltaVista03 — *http://www.altavista.com*, 2003

BaezaYates99 — R. Baeza-Yates, B. Ribeiro-Neto "Modern Information Retrieval". *ACM Press / Addison-Wesley Longman* (1999)

Benitez98 — A. B. Benitez, M. Beigi, and S. F. Chang "Using Relevance Feedback in Content-Based Image Metasearch". *IEEE Internet Computing*, Vol.2, No.4, pp.59-69 (1998)

Bergman01 — Michael K. Bergman "The Deep Web: Surfacing Hidden Value". *The Journal of Electronic Publishing*, Vol.7, Issue 1 (2001)

Budzik99 — Budzik, J., and Hammond, K. J. "Watson: Anticipating and Contextualizing Information Needs". *Proceedings of the Sixty-second Annual Meeting of the American Society for Information Science*, Medford, NJ, 1999 (1999)

Callan01 — J. Callan and M. Connell "Query-based sampling of text databases". *ACM Transactions on Information Systems*, Vol.19, No.2, pp.97-130 (2001)

Callan95 James P. Callan, Z. Lu and W. Bruce Croft "Searching distributed collections with inference networks". *Proceedings of the 18th ACM-SIGIR International Conference*, Seattle, Washington, USA, July 1995, pp.12-20 (1995)

Callan99 J. Callan, M. Connell, and A. Du "Automatic Discovery of Language Models for Text Databases". *Proceedings of ACM-SIGMOD International Conference on Management of Data*, Philadelphia, Pennsylvania, USA, 1-3 June 1999, pp.479-490 (1999)

Craswell00 N. Craswell, P. Bailey, D. Hawking "Server selection in the world wide web". *Proceedings of the 5th ACM Conference on Digital Library*, San Antonio, TX, USA, 2-7 June 2000, pp.37-46 (200)

Dogpile03 *http://www.dogpile.com*, 2003

Dolin99 R. Dolin, D. Agrawal, and A. Abbadi "Scalable collection summarization and selection". *Proceedings of the 4th ACM Conference on Digital Libraries*, 11-14 August 1999, pp.49-58 (1999)

Dreilinger97 D. Dreilinger, Adele E. Howe "Experiences with Selecting Search Engines Using Metasearch". *ACM Transactions on Information Systems*, Vol.15, No.3, pp.195-222 (1997)

Exicte03 *http://www.excite.com*, 2003

Family03 *http://www.familyfriendlysearch.com*, 2003

Fuhr99 N. Fuhr "A decision-theoretic approach to database selection in networked IR". *ACM Transactions on Information Systems*, Vol.17, No.3, pp.229-249 (1999)

Gauch96 S. Gauch, G. Wang, and M. Gomez "Profusion: Intelligent fusion from multiple, distributed search engines". *Journal of Universal Computer Science*, Vol.2, No.9, pp.637-649 (1996)

Glover01 E. Glover, G. Flake, S. Lawrence, W. P. Birmingham, A. Kruger, C. L. Giles, and D. Pennock "Improving category specific web search by learning query modifications". *Symposium on Applications and the Internet*, SAINT, San Diego, CA, 8-12 January, 2001 (2001)

Glover99 E. Glover, S. Lawrence, W. Birmingham, C. Lee Giles "Architecture of a Metasearch Engine that Supports User Information Needs". *Proceedings of the 8th International Conference on Information Knowledge Management*, Kansas City, MO, November 1999, pp.210-216 (1999)

Google03 *http://www.google.com*, 2003

Gravano94 L. Gravano, H. Garc'ia-Molina and A. Tomasic "Precision and Recall of GlOSS Estimators for Database Discovery". *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems*, Austin, Texas, 28-30 September 1994, pp.103-106 (1994)

Gravano96 L. Gravano and H. Garcia-Molina "Generalising GlOSS to vector-space databases and broker hierarchies". *Proceedings of the 21st International Conference on Very Large Databases*,

Zurich, Switzerland, 11-15 September 1995, pp.78-89 (1995)

Gravano97 L. Gravano and H. Garcia-Molina "Merging Ranks from Heterogeneous Internet Sources". *Proceedings of the 23rd International Conference on Very Large Databases*, Athens, Greece, 25-29 August 1997, pp.196-205 (1997)

Gravano99 L. Gravano, H. Garcia-Molina, A. Tomasic "GlOSS: Text-source discovery over the Internet". *ACM Transactions on Database Systems*, Vol.24, No.2, pp.229-264 (1999)

Hawking99 D. Hawking, P. Thistlewaite "Methods for information server selection". *ACM Transactions on Information Systems*, Vol.17, No.1, pp.40-76 (1999)

Ipeirotis00 P. G. Ipeirotis, L. Gravano and M. Sahami "Automatic classification of text databases through query probing". *Proceedings of the 3rd International Workshop on the Web and Databases*, Dallas, Texas, 18-19 May 2000, in conjunction with ACM PODS/SIGMOD 2000 (2000)

Ithaki03 *http://www.ithaki.net*, 2003

Ixquick03 *http://www.ixquick.com*, 2003

Lawbot00 Sandip Debnath, Sandip Sen, Brent Blackstock "LawBot: A Multiagent Assistant for Legal Research". *IEEE Internet Computing*, Vol.4, No.6, pp.32-37 (2000)

Lawrence98 S. Lawrence and C. L. Giles "Inquirus, the NECI meta search engine, ". *Proceedings of the 7th International World Wide Web*

Conference, Brisbane, Australia, 14-18 April 1998, pp.95-105 (1998)

Lawrence99 S. Lawrence, C. Lee Giles "Accessibility of information on the Web". *Nature*, Vol.400 pp.107-109 (1999)

Lycos03 *http://www.lycos.com*, 2003

Meng98 W. Meng, K. L. Liu, C. Yu, X. Wang, Y. Chang, N. Rishe. "Determining Text Databases to Search in the Internet". *Proceedings of the 24th International Conference on Very Large Databases*, New York City, NY, USA, 24-27 August 1998, pp.14-25 (1998)

MetaXChem03 *http://www.chemie.de/metaxchem*, 2003

Mitra98 M Mitra, A Singhal, and C Buckley "Improving automatic query expansion". *Proceedings of the 21st International ACM-SIGIR Conference*, Melbourne, 24-28 August, 1998, pp.206-214 (1998)

Mowshowitz02 A. Mowshowitz and A. Kawaguchi "Bias on the web". *Communications of the ACM*, Vol.45, No.9 (2002)

ODP03 *http://dmoz.org*, 2003

Profusion03 *http://www.profusion.com*, 2003

Selberg97 E. Selberg, O. Etzioni "The MetaCrawler Architecture for Resource Aggregation on the Web". *IEEE Expert*, Vol.12, No.1, pp.8-14 (1997)

SoftCrawler03 *http://web2.aw-da-wicked.de*, 2003

Sugiura00 A. Sugiura and O.Etzioni "Query Routing for Web Search Engines: Architecture and Experiments". *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, Netherlands, 15-19 May 2000, (2000)

Voorhees95 E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird "The collection fusion problem". *Proceedings of the 3rd Text Retrieval Conference*, no.500-225 in NIST Special Publications, Gaithersburg, MD, April 1995, pp.95-104 (1995)

Xu98 J. Xu and J. Callan "Effective retrieval with distributed collections". *Proceedings of the 21st International ACM-SIGIR Conference*, Melbourne, Australia, 24-28 August 1998, pp.112-120 (1998)

Yahoo03 *http://www.yahoo.com*, 2003

Yu01 C. Yu, W. Meng, W. Wu, K.-L. Liu "Efficient and effective metasearch for text databases incorporating linkages among documents". *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, Santa Barbara, California, 21-24 May 2001, pp.187-198 (2001)

Yuwono97 B. Yuwono and D. L. Lee "Server ranking for distributed text retrieval systems on the Internet". *Proceedings of the 5th Annual International Conference on Database Systems for Advanced Applications*, Melbourne, Australia, April 1997, pp.41-49 (1997)