NOTE TO USERS

University of Alberta

A Comparative Study of Keyphrase-based Query-specific Clustering on WWW

by

Peng Wang 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science.**

Department of Computing Science

Edmonton, Alberta
Fall 2004

Your file Votre référence
ISBN: 0-612-95875-2
Our file Notre référence
ISBN: 0-612-95875-2

To my grandma and parents

Acknowledgements

First and foremost, I would like to thank Professor Russ Greiner and Dekang Lin as my supervisors. They guided me through the writing of the thesis. I am very thankful that they gave me the flexibility of pursuing my MSc degree while working here in Calgary. Thanks to my committee members for the time and effort they have spent revising my thesis. Also thank you to my friend and classmate Tingshao Zhu for his useful suggestions.

Last but not least, I'd like to thank my girlfriend, and all of my family members for their constant support, encouragement and patience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

1.1.1 What is the problem?

Over the past decades the World Wide Web has become one of the biggest phenomena in history. It has been growing at an exponential rate. The implication of the WWW has touched every corner of the society and it has become one of the most important and efficient communication media in people's daily life: It enables companies to establish 24/7 online showcase windows, it helps scholars gather research literature and data, it saves consumers effort and time by letting them make purchases online... According to the survey by OCLC (Online Computer Library Center) [49], as of June 2002 the information on the visible Web (as opposed to invisible Web [59] which basically means information on the Web that is not accessible to the traditional Web-crawling technology that is used by search engines to build their indexes) was approximately equivalent in size to between 14 and 28 million books. The number of volumes held by Harvard University, which has the largest number of volumes in the 112 Association of Research Libraries (ARL), is under 15 million [49]. In the middle of 1995 the number of Web sites was 23,500 [13], while in 2002 the Web has grown to over 3 million sites as indicated by the OCLC report. There are one and a half million Web pages that are added to the Web every day according to analysts at Alexa Internet in 1998 [40]. As to September 2003 Google has indexed over 3 billion Web pages. However, the actual World Wide Web could be 500 times larger than major search engines now show [25].

Along with the convenience of vast amount of digital information made available by the WWW to any computer connected to the Internet comes this question: *how do people find their desired information on the WWW?* Although "Information explosion" has made the massive amount of data accessible to the general public, searching for the wanted information has turned out to be a challenge. The World Wide Web is described as unorganized, chaotic and incorporating dubious quality information [16]. Search engines emerged as a solution to alleviate the information overload problem. Although they have become very useful as a key to the door of WWW to most people, they are far from perfect. Using a search engine does not require much training. Users simply formulate what they want into a discrete

set of key words/phrases (or a combination of these key terms with or without boolean operators) and send the query to search engines. Search engines will then return a hit list that basically consists of documents deemed "relevant" according to their own different proprietary algorithms. The lists of documents are always ranked from "most relevant" to "less relevant" again weighted by different ranking schemes patented by search engines. The returned ranked document list usually is composed of thousands of documents. To find useful Web pages, users would often have to sift through the long ranked list of documents to dig out the ones they truly want. Hence the search process often turns into a tedious and lengthy manual screening procedure. Although search engines have been always trying to improve their indexing and ranking algorithms in order to deliver fast and better quality search results in the past years, the problem mentioned above remains. Actually wading through such a long ranked list is not only daunting but also impractical. After analyzing the Web log of search engine *Excite*, Jansen et al. found more than 50% of users did not view results beyond the first page [38].

The majority of the search engines available on the Internet utilize "literal search" method: Search engines look for the search terms exactly as they are entered [12]. Literal search is intrinsically problematic. Meanings of words often tend to be semantically obscure when context is not specified. Because the World Wide Web is a big "open source" library contributed by people with different backgrounds in terms of ages, education, ethnics, cultures, languages etc, words in Web documents are more inclined to be polysemous than in better-structured and conceptually more homogeneous document collections. The impact from pop culture on the World Wide Web further escalates this phenomenon. For example, "*soprano*" is normally used to refer to a female singer with the highest singing voice, but if you type in "*soprano*" in a major search engine, chances are a lot of documents that you get will be related to the mafia crime boss "*Tony Soprano*" depicted by a very popular TV series "*The Sopranos*".

Although subjects can inherently have multiple aspects, e.g., "chip" as in high tech industry and as in food industry, certain aspects tend to dominate general purpose search engine's query results for a specific subject. This "bias" can be considered as the reflection of the "bias" embedded in the search engine's ranking

algorithm. Conceivably individual algorithm will be in favor of certain aspects of a subject due to the document distributions of different topics under that subject and search engines' ranking scheme. So when a general-purpose search engine serves the majority of its user base really well, its ranking algorithm can pose problems to other users when their information needs expressed in queries do not accord with search engines' "query bias". For instance, suppose after watching movie "hurricane" which depicts the tragic story of the famous boxer Rubin Carter, users want to know more about of the life of this boxer. If they query "hurricane" on Google, they would have a hard time finding what they are looking for since information about boxer "hurricane" is inundated by Web pages talking about the natural phenomenon hurricane. Although one may argue search engines often provide utilities like boolean operators or modifiers (e.g., "+", "-", ...) for users to further refine their queries in order to disambiguate their queries, Web user study [38] shows only a very small percentage of the Web users would actually use these features. The majority of the users just typed in some simple key terms to start querying instead of taking time to learn how to use the boolean operators and modifiers. Their research also shows that two in three Web queries were comprised of one to two terms which tend to aggravate the ambiguity problem mentioned above.

1.1.2 How are we going to solve the problem?

Traditional information retrieval system's retrieval quality, which can be roughly described as "retrieve as many relevant documents as possible (recall) and at the same time fetch as few as possible irrelevant documents (precision)", vitally depends on the quality of queries composed by users. Unfortunately common users are not very good at formulating their queries. This may be caused by the vague nature of the information needs, e.g., instead of trying to pinpoint the answer to a very well-defined specific question, the user may only be interested to gain an overview of a general question. Or it may because the user is not very familiar with the domain and hence has problem constructing a proper keyword set to describe his topic of interest. Ill-formed queries can certainly hinder a well-built retrieval system from getting satisfactory results.

Basically there are two ways to improve users' satisfaction during information re-

trieval process: improve the retrieval results per se and make the best out of existing results. In this thesis we take the second direction by employing document clustering techniques. Instead of solely relying on users coming up with good queries that is also system-dependent, retrieval result clustering can help users quickly spot what they are truly looking for, disregard unpromising documents and refine their queries. This is achieved by categorizing query results, which normally have high recall but low precision, into topic-coherent groups. This paradigm of document clustering is called query-specific clustering or post-retrieval clustering. Instead of clustering the whole document collection, query-specific clustering applies document clustering techniques to organize search results for users to navigate. Thus query-specific clustering tends to be more context-specific and instead of a retrieval method, it is used as a search result output presentation.

We believe query-specific document clustering technique can be used to relieve the problem even if it is not a thorough solution. It fits into the Web setting and unlike document classification it does not require prior knowledge of labels under which each document should be classified. Hence it has the potential to help users easily find group of documents that is of interest among other irrelevant documents. Instead of statically clustering the whole document set available on the Web, we think it is easier, more efficient, and possibly more accurate to cluster documents returned by the search engine in response to a search query.

Although a lot of research has been done with respect to performance analysis of different document clustering methods, surprisingly, to the best knowledge of the author there is very few similar studies being done in the domain of World Wide Web. Studies have been done mostly either using existing traditional IR system or on well-known lab experimental data collections like TREC [9, 17, 3, 42, 61]. Web resource differs greatly from traditional organized document collection or specialized document collection. It has a much larger variety of topics and document content qualities can vary immensely, which makes the World Wide Web a much more heterogeneous and loosely structured data reservoir. Clustering algorithms are notoriously known for their different performance with respect to different databases. One study also has shown that users exhibit highly distinct behaviors in Web search in comparison with traditional IR systems [38]. Therefore it is imperative and

meaningful for us to study document clustering techniques in the context of the ever-changing World Wide Web. Prior research work has been focused mainly on application of document clustering on the collection in a static fashion (i.e., clustering the whole collection offline) [63, 39, 11]. Document clustering on retrieval results has not been thoroughly studied. Although there have been some research efforts conducted in dynamic post-retrieval document clustering [17, 42], the results of comparative studies of effectiveness between query-specific clustering and traditional similarity-ranking list have been inconclusive.

We wanted to investigate the effectiveness and usefulness of query-specific (post-retrieval) clustering in the domain of the Web. A previous study shows that navigating through the hierarchy generated based on the whole document set to find the relevant documents to the supplied query was less effective than examining the subset of documents yielded by similarity search [50]. Therefore in the query-specific clustering paradigm, clustering techniques are adopted in a post-retrieval fashion and work as an alternative to ranked title list. Furthermore it would be interesting and valuable to explore the possibility and effectiveness of using keyphrase list as the representation of Web documents in clustering. Keyphrase list is defined as a short list of phrases that capture the main topics of a document [62]. Keyphrases provide semantic metadata that deliver the topic of the document in a concise way. We developed *Extoken*, a keyphrase extraction algorithm using HTML formatting elements which is designed specifically for Web documents. Although HTML provides meta elements where the author can provide keyword list that summarizes the content of the page, in reality only a small fraction of the existing Web pages actually come with them. To make things worse, for various purposes (e.g., bumping up the page's ranking in search engine, attracting certain user groups etc), some of the pages give incorrect or misleading keyphrases. Thus it is important for us to develop an automatic keyphrase extraction algorithm in this research. Because one of the goals of designing Extoken is for the application of document clustering on the Web search results, this algorithm incorporates modules to consult WordNet [46] as a means to tailor the keyphrase extraction to the needs of users in terms of search queries.

1.1.3 Research Outline

In this research we examined the feasibility of keyphrase extraction with respect to real Web documents as well as its usage in Web document clustering. Our objectives of the research can be summarized as follows:

- Evaluate the quality of keyphrases generated by our keyphrase extraction algorithm. Specifically, we would like to see if using HTML formatting elements can help improve the quality of keyphrases.
- Compare the effectiveness between query-specific clustering presentation and the traditional ranked page list.
- Investigate how different experimental conditions can influence the effectiveness of the query-specific clustering (e.g., different page representations including keyphrases, full textual content and page summarization returned from the search engine, different numbers of top-ranked pages).

In order to gather "real data" from users and demonstrate our research, we implemented and tested two online prototype systems: *Phrastractor* and *Categorizer*. Phrastractor is an online keyphrase extraction system that demonstrates our keyphrase extraction algorithm Extoken. Based on the given URL, it outputs a list of extracted keyphrases that can help users easily grasp the main idea of the Web page. Users can leverage the system to help them better formulate queries as well. For example, when "www.calgaryflames.com", the homepage of Calgary flames (NHL hockey team of Calgary), is provided to Phrastractor as input, as of June 5, 2004 (the day for Game 6 between Calgary flames and Tampa Bay Lightning for Stanley Cup Final), the top 5 extracted keyphrases are "flame", "Game", "Calgary Flames", "Stanley Cup" and "Iginla" (captain of Flames). *Categorizer* is a fully functional clustering search engine using Google as its backend search engine. It provides clustering as an alternative search result presentation to the original ranked list returned from Google. Both ways to present query results are incorporated into the system for users' convenience. For example, when "salsa" is queried in "Categorizer", in the clustering solution output, users can find node labelled as "danc, latin, histori, music" (the terms in the documents have been stemmed) which contains

Web documents talking about salsa as a genre of music and dance; users will also see node labelled as "hot, recip, sauc" which has documents related to recipes of the hot sauce salsa. The ideal clustering solution would have all documents returned from the search engine in response to the query grouped into conceptually related topics.

We proposed an objective keyphrase evaluation scheme that measures the quality of the keyphrases based on the ranking of the source documents in the retrieved page list from Google in response to the extracted keyphrases as queries. We also carried out comparative study using the clustering search engine to explore the benefit of keyphrases used as document representation as opposed to the whole document and document summarization (search engine snippet). Moreover, we compared the effectiveness between query-specific clustering and conventional page ranking. To avoid subjectivity of human relevancy judgement, we proposed a novel objective relevancy judgement generation method and used it in the framework of our query-specific clustering research. Our study shows that this method is promising and helps to greatly simplify the process of evaluation without introducing human bias. After knowing the relevancy of each individual document to the query, We used E-effectiveness which combines recall and precision used in IR as the measurement of effectiveness of clusters. Hence an effective cluster should contain as many relevant documents as possible (measured by recall) and in the same time as few irrelevant documents as possible (measured by recall). The optimal cluster effectiveness which is represented by the cluster that has the best E-effectiveness in the clustering solution is used as the effectiveness metric for the clustering.

Because there has been some clustering search engines on the Web, in our experiments, we also compared the effectiveness of our system with *Vivisimo* [34], the leading commercial clustering search engine on the Web. Our preliminary experimental results proved that the proposed clustering model in this thesis is very promising.

1.2 Thesis Structure

The rest of the thesis is organized as follows: Because the system we developed is a Web clustering meta search engine, Chapter 2 provides some background on

searching on WWW as well as essential document clustering preliminaries. We also report and analyze the related research work in the area of document clustering and its application on the Web in Chapter 2. Chapter 3 describes our underlying keyphrase extraction algorithm and the online keyphrase extraction demo system we developed. Chapter 4 covers the architecture and design of our Web clustering search engine prototype. Chapter 5 gives details about our experimental setups and methodologies. Subsequently it presents the experimental results followed by analysis and discussion. Finally Chapter 6 outlines the conclusion of our research and provides future directions for expanding the scope of this research as well as areas of future work.

Chapter 2

Background and Related Work

2.1 Searching on WWW

There are numerous Web search tools that are designed to assist Web users to find helpful information on the Web faster and more easily. These tools can be roughly broken down into the following categories:

1. Server side Web search engine (e.g., Yahoo, Google, AlltheWeb)
2. Client side search tool (e.g., Grokker developed by Groxis)
3. Personal search assistant tool (e.g., Google toolbar, Alexa toolbar)
4. Web directories (e.g., open directory project, Yahoo Web directory)
5. Database search engine (e.g., completeplanet)

The next subsection focuses on Web search engine because of our research orientation (our Web document clustering system is a meta search engine) and its dominant status among all Web search tools. Besides, in order to better understand the context of our clustering search engine, it is important to know the underlying technologies behind major search engines on the Web.

2.2 Web Search Engine

Web search engines help users navigate in the information ocean on the Web. Traditional search engine's structure can be illustrated by Figure 2.1. It is mainly comprised of three key components: Web crawler, indexing software and ranking algorithm.

2.2.1 Web Crawler

Before a search engine can help users find relevant Web pages, it has to know their existence. A Web crawler (also called "*spider*" because it crawls over the Web) is a computer software robot that is used by search engines to scour the Web and collect Web pages. Web crawler finds all URLs on a Web page and follow those links to the pages they point to and so on. It normally starts from a set of seed pages (these are usually lists of heavily used servers and very popular pages [12]).

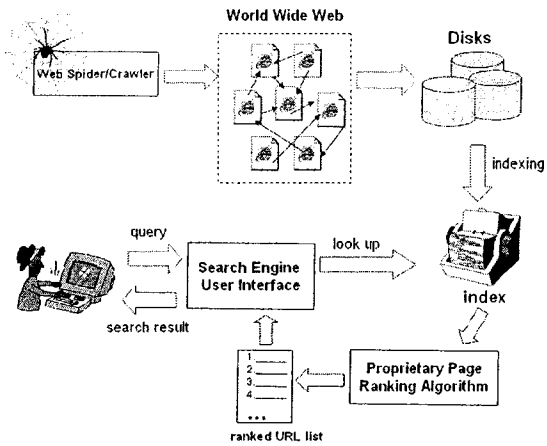

Figure 2.1: Basic structure of Web search engine

Search engine usually uses multiple Web crawlers in parallel to achieve satisfactory speed [37].

2.2.2 Indexing Software

Indexing is the process of building a data structure that can be quickly searched [37]. A naive indexing algorithm would simply store the words and the URL where they are found. Such a simple system is obviously of little practical use and can be easily fooled by "Web spamming" techniques (e.g., adding words irrelevant to the page's content to gain top position for queries). Besides words in Web pages, search engines usually need to store extra information in order to improve query return relevancy and enable ranking. Modern search engines usually exploit all sorts of information provided by the Web page. These could include:

- **Normalized term frequency**. Usually stop words (words that carry no semantic meaning, e.g., conjunctions, prepositions etc) are stripped out and the other words will be indexed.

- **Formatting information**. Search engines may keep track of the words in headings, titles, links [12] as well as other formatting attributes such as capitalization, font size, bold/italic/underline style.

- **Top-position text**. It is deemed that text close to the beginning is often more related to the topic of the page. Search engines sometimes elect to store

the first sentence of a Web document [37]. Others choose to store the first two lines of text [12].

- **HTML meta tags.** Meta tags are invisible part of the HTML page. Authors often use meta tags to give the topic/keywords/description of the pages. Meta tags are usually intended for potential organizers of Web pages (e.g., Web directories editors, search engines). Although meta tags can offer great value for interested readers to understand the page's topic, their validity can not be taken for granted. Page authors may give faulty keywords that are inappropriate or sometimes misleading due to thoughtlessness. Sometimes they even intentionally list keywords that are completely irrelevant to the pages' topics in order to increase the chance of their Web pages being seen by Web users.

This extra information often needs to be encoded into a compact form in order to improve storage efficiency.

2.2.3 Ranking Algorithm

Ranking algorithm enables search engine to present the search results in an orderly fashion which usually means results are sorted by their relevancy to the query. Every search engine has its own ranking algorithm. Ranking algorithm is one of the most important factors that distinguish one search engine from others. "Text analysis" and "popularity analysis" are usually combined to decide the rank of a page. Merely relying on one of them would lead to unbalanced and problem-prone ranking solution. For example, if only "text analysis" is employed, a Web page can artificially boost its ranking for a subject by repeating certain words in its text intentionally even if those words have nothing to do with its true topic. On the other hand, the "popularity analysis" can be fooled by link farms that link to totally unrelated sites just to increase their popularity score. Google's PageRank^{TM} variant ranking algorithm built the foundation of its success. In addition to using PageRank^{TM} as its core ranking mechanism, Google also looks at other elements to enhance its ranking (e.g., where the query terms appear in the document; if the query words are in the headline, the document is very likely to be relevant to the query).

2.2.4 Search Engine Categories

There are different search engines on the Web nowadays. Based on their purposes and characteristics, they can be roughly classified into the following categories:

General-purpose Major Search Engines

According to the sources of their listings, they can be further divided into these types [32]:

- **Crawler-based.** The main sources of indexed Web pages come from Web crawlers. Search engines such as *Google*, *AllTheWeb*, *AltaVista* belong to this type.
- **Human-compiled.** Majority of the results are compiled by human editors. It generally comes from two sources: commercial sites pay to be listed there and human editors catalog existent Web sites into respective categories. *LookSmart*, *Open Directory* fall into this category.
- **Paid lists** Web sites submit themselves to be listed for a fee, e.g., *Overture.*

Meta Search Engines

Meta search engines generally do not have their own Web index databases. Instead they send users' queries to other search engines like *Google* and *Alta Vista.* in parallel and synthesize the results in their own ways. The way meta search engine works can be illustrated by Figure 2.2. Basically meta search engine accepts the query from the user, translates the query into the proper syntax for each of its backend search engines and dispatches it to them simultaneously. It is mainly what meta search engines do after receiving results from other search engines that distinguishes them. There are hundreds of meta search engines and according to their post-retrieval processing, they can be broken down into the following categories:

- **Separate listing.** Meta search engine may simply list different results under respective search engines (i.e., search results are grouped by their original sources), e.g., *Dogpile* [26].

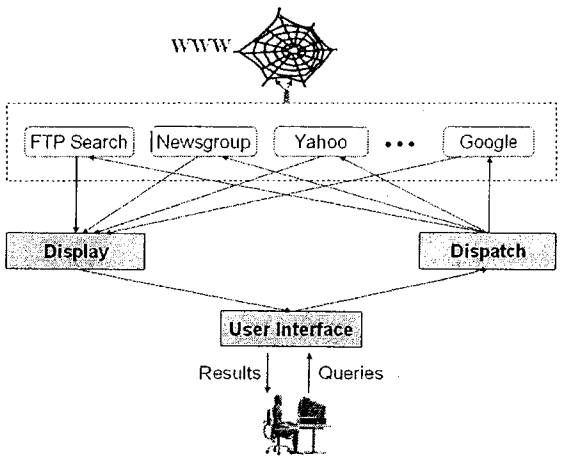

Figure 2.2: System structure of a metasearch engine [37]

- **Unified listing.** Some Meta search engines blend the results from different search engines into a unified list according to their own analysis and ranking algorithms, e.g., *MetaCrawler* [31].

- **Application of IR techniques.** This type of meta search engines takes the query results from other search engines as the input to their information retrieval technologies and provides search results based on that. For instance, *Vivisimo* [34] organizes the results from backend search engines into conceptually-related categories using document clustering technique and presents the clustering solution to users.

- **Novel visual user interface.** By using specially designed user interface, although these meta search engines run the risk of giving users unfamiliar environments to search in, they manage to either provide more information to the user or help them view the information from some other unique perspectives. For example, by giving users a chance to see how it decides a page's relevancy, *SurfWax* [33] provides more clues (e.g., where the keywords that the user just entered appear in the document, an on-the-fly generated abstract of the page as well as the number of links, images contained in the page, etc) to users to help them determine if a page is of interest. *Kartoo* [30] presents the search result using a visual map with that users can interact to modify their queries.

We want to stress here that the properties mentioned above are not mutually exclusive. In fact meta search engines often use a combination of the properties to accommodate the potentially different needs of users.

Meta search engines achieve wider coverage of the Web and relieve the need for the users to learn the query syntax on different search engines. However they are more and more inclined to use influence from commercial interests, which typically means a lot of paid inclusions are mixed in their editorial results (Figure C.1 in Appendix C shows the percentage of paid links in the first page of query result based on a sampling experiment conducted by SearchEngineWatch [32]). The other drawback is a lot of them do not provide advanced search features for users to fine tune their queries.

Specialized Search Engines

In contrast to general search engines, specialized search engines are designed to better serve users' particular needs. Most of specialized search engines focus on a specific domain or subject, e.g., legal search engines, medical search engines, shopping search engines, newsgroup search engines, news search engines, etc. Some specialized search engines are national or regional search engines. Some specialize in particular media such as audio, video, graphics, and images. Others are tailored to target specific age groups, e.g., kids search engines such as yahooligans [36].

2.2.5 Challenges of Web Search Engines

In order to better understand the challenges Web search engine community has to confront, we approach them from two facets: user side and search engine side.

User Side

Users come to search engines to search for information for a wide variety of reasons, e.g., students use the Web to find materials to put into their essays, scholars search the Web for relevant literature to stay ahead in their research areas, customers visit online stores for shopping, job hunters go to online career Web sites to find job opportunities. Their searching skills vary greatly; some of them may be very skillful and some of them may not have much computer skills at all. These factors will

put a dent on how well they can take advantage of the features provided by search engines. According to the Web log study conducted by Jansen et al. [38], only a small percentage of the users actually employed boolean operators or modifiers provided by search engines. In addition, a significant portion of the users who actually exploited these advanced search features used them wrongly. Hence instead of helping users fine-tune their queries, these features impeded those users from fulfilling their information needs.

These facts make us raise such questions: Is there any other better way for users to express their information needs other than the literal query employed by search engines? Instead of forcing users to learn how to correctly use boolean operators or modifiers in order to accomplish more complex information retrieval tasks, how can we enable a mechanism incorporating technologies like natural language processing to understand the ideas and concepts which users can express in a natural way?

Search Engine Side

Elements needed to be stressed on search engine side can be roughly broken down into the followings:

- **Coverage of the Web.** Although search engines have improved a lot in terms of number of pages they indexed, there is still a lot of Web content that they have not been reached [25]. Aside from that, how can Web crawlers from search engines find the "invisible Web", which contains a great deal of authoritative information that is largely comprised of content-rich databases from universities, libraries, associations, businesses, and government agencies across the world?

- **Freshness of the index.** The Web is changing all the time. Every day a vast number of new pages are created and also a great many old pages are deleted. There are also a lot of pages that are moved around in different places. It is crucial for Web search engines to always keep up with the speed of change. According to sampling study conducted by Hearne [16], as of October 2002 the best search engines in terms of having the most recent "snapshot" of the Web are about 2 weeks old.

- **More accurate and robust ranking algorithm.** The ultimate goal of search engines is to improve the relevancy of returned hit list in response to users' queries. Google's PageRank^{TM} has proven to be a huge success due to its high relevancy. The way PageRank works can be summarized as follows:
  - The more inbound links Page A receives, the higher the PageRank of Page A tends to be.
  - The PageRank of the page that links to Page A decides its contribution to the PageRank of Page A.
  - The PageRank of Page A is influenced by all pages that link to Page A.

  However the adoption of "link analysis" as opposed to "text analysis" by Web search engines raises another question: How to effectively prevent "*link spamming*" (i.e., artificially increasing Web sites' ranks in search engines by boosting link population through increasing inbound links illegitimately)? For example, "link farms" are designed to artificially increase the sheer quantity of links that point to certain Web sites in order to fool Google to enhance their ranks [15]. How can Web search engines' ranking algorithms be robust enough to fend off this kind of abuse? Failing to address this issue will inevitably lead to deterioration of search engines' ranking quality. The other problems of PageRank^{TM} stems from its assumption that best sites are those with the most people linking to them. It is questionable that if we can always draw an equation between popularity and authoritativeness. Users may prefer a more authoritative source instead of popular Web sites sometimes.

- **Better user interface.** The majority of the search engines still use the ranked document list as the interface for users to browse for relevant pages. As discussed in the previous chapter, one of its problems is that truly useful pages are buried with hundreds of other irrelevant pages, which makes it a drudgery for users to wade through the long list seeking relevant ones. Other alternatives, such as query result visualization techniques, have been used to replace the ranked list, however none of them is advanced enough for users to switch from traditional ranked list interface.

- **Query adaptability.** Users sometimes either lack the skills to form queries precise enough to satisfy their information needs or simply do not care to tailor their queries to the design of the search engines to make the optimum use of them. Hence more user-friendly query forms like natural language expression would be more acceptable to the general public. However natural language processing research has not reached the level when it can be extensively employed by search engines.

- **Automated multimedia indexing.** Although there are some multimedia search engines on the Web retrieving rich media such as audio, video, graphics and images, they still heavily rely on human reviewers or associated text to index the multimedia data. In the past years numerous multimedia indexing algorithms have been proposed in literature, nevertheless it is still a very challenging problem for researchers to develop automated multimedia indexing algorithms that can meet the efficiency requirements of Web multimedia data [37].

2.3 Web Content Mining

It is important to perceive the status of our research work in the big picture of Web mining in order to gain a complete understanding of our research. *Web mining* is defined as the discovery and analysis of useful information from the World Wide Web [5]. Web mining is the equivalent of data mining of database in the context of the Web. However the heterogeneous, unstructured nature of the Web coupled with its massive amount of ever-changing data pose the need of using different techniques. The goal of Web mining is to develop "more" intelligent tools for information retrieval to help the users in finding, extracting, filtering and evaluating the desired information and resources on the Web [45].

Cooley et al. [5] broke down the current research works in Web mining into two categories: Web content mining and Web usage mining. The taxonomy of Web mining can be illustrated by Figure 2.3. Web usage mining is about mining for user access patterns from Web servers. It mainly consists of pattern discovery and pattern analysis. Pattern discovery is the process of uncovering Web users' browsing

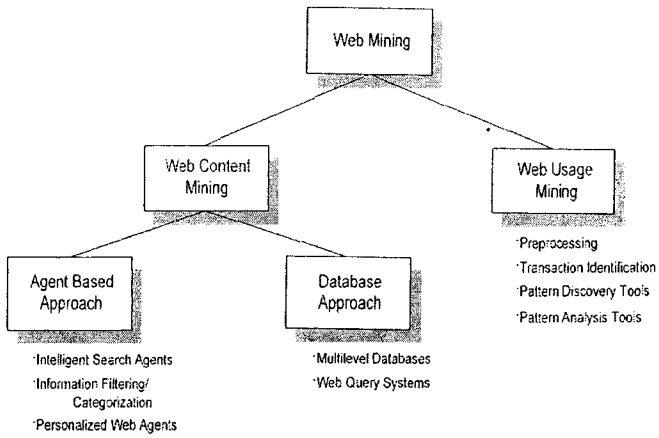

Figure 2.3: Taxonomy of Web Mining[5]

and access patterns hidden in the Web log files. For instance, association rules and sequential patterns can be found from server access logs. Pattern analysis helps us understand and interpret the discovered access patterns [5].

Web content mining is the process of developing automatic intelligent tools for users to organize and retrieve useful data from the unstructured online sources. Web documents are at most semi-structured. They can contain text, HTML tags, hyperlinks, meta tags as well as graphics, images, audio and video. They are also highly heterogeneous in terms of content. Searching through such an unstructured huge collection proves to be a very challenging task. According to Figure 2.3, Web content mining can be achieved from two distinct perspectives: **agent-based approach** and **database approach**. Information retrieval techniques are widely adopted in agent-based approach. Agent-based approach can be further categorized into intelligent search agents, information filtering/categorizaiton and personalized Web agents. Our research work fits into the second one, information filtering/categorization, since the core part of our clustering meta search engine is basically an online Web document clustering system. To take advantage of the well-developed database query languages (e.g., SQL), the database approach tries to make the unstructured or semi-structured Web into a database structure in order to better manage and organize data.

2.4 Document Clustering

Cluster analysis is a technique that allows the identification of groups, or clusters, of similar objects in multi-dimensional space [6]. Document clustering is the application of cluster analysis on document set. We have no intention to cover an extensive in-depth survey of clustering techniques here. Our interest is solely in document clustering, hence the scope of this section is to describe common clustering techniques used in document clustering area.

2.4.1 Preliminaries

Before we start to further discuss Web document clustering, we need to go over some basic concepts with respect to how documents are represented and how we compute the similarity between documents in order to better understand more advanced document clustering techniques. The vector space model is employed in this thesis as the base for document clustering.

Preprocessing

Before documents can be processed by computer algorithms, they have to be "cleaned up". Preprocessing usually consists of the following steps:

- **Stop word removal**. Stop words, which are defined as frequent words that carry no information (i.e., pronouns, prepositions, conjunctions, etc), are stripped off in this stage. Instead of contributing to the topic of the document, these words (such as "the", "this", "and") tend to add noise which decreases the effectiveness of clustering algorithms.

- **Word stemming**. Word stemming is the process of suffix removal to generate word stems [2]. The Porter stemmer [51] is the most well-known algorithm for this task and also the one we used in our research.

Vector Space Model

In order to be automatically processed by computer, documents have to be represented in a "computer-friendly" form. One of the most commonly used document

representations is the **vector space model**. In this model each document d_i is represented by a vector in the multi-dimensional term-space:

$$d_i = (t_{1i}, t_{2i}, ..., t_{ni})$$

where t_{ki} is the weight of the kth term in d_i.

There are different term weighting schemes. We describe the most commonly used ones here:

1. **Boolean weighting**

   The weight of the term is 1 if it is present in the document and 0 if it is not:

   $$t_{ij} = \begin{cases} 1 & \text{if } f_{ij} > 0 \\ 0 & \text{if } f_{ij} = 0 \end{cases} \tag{2.1}$$

   where f_{ij} indicates how many times the ith term appears in document d_j.

2. **Term frequency weighting**

   In this scheme the frequency of the term is used as the weight:

   $$t_{ij} = f_{ij} \tag{2.2}$$

3. **tf – idf weighting**

   This approach is a refinement of term frequency weighting scheme. It recognizes the fact that the more documents that include a certain term, the less discriminating power the term has. In this approach the weight of a term is determined by two factors: tf and idf. tf stands for term frequency which is the raw frequency of the term in a document, while *idf*, *inverse document frequency*, is used to play down the weight of the terms that appear frequently in many documents. So the weight of the term is computed by:

   $$t_{ij} = f_{ij} \times \log(\frac{N}{n_i}) \tag{2.3}$$

   where N is the total number of documents and n_i denotes the number of documents that have the ith term at least once.

The importance of normalization can not be overrated in the vector representation of documents. In order to have a fair comparison between documents of different lengths, document vectors are normalized to be of unit length, i.e., $\|d_i\| = 1$.

Document Similarity Measures

Human beings use semantics to tell the similarity between documents. Unfortunately this ability can not be readily adopted into computer logics. During the past decades, researchers have been exploring different ways for computers to compute the similarity between documents based on quantitative traits. Here we describe two measures that have been widely used in document clustering research.

The first method is the cosine measure [57]:

$$cos(d_i, d_j) = \frac{d_i \cdot d_j}{\| d_i \| \| d_j \|} \tag{2.4}$$

where $d_i \cdot d_j$ denotes the dot product of two vectors d_i and d_j and $\| d \|$ is "the norm" or length of vector d. When document vectors are normalized, cosine measure can be simply represented by

$$cos(d_i, d_j) = d_i \cdot d_j = \sum_{k=1}^{n} t_{ki} t_{kj} \tag{2.5}$$

In the following chapters, we always assume document vectors are normalized. In this scheme the higher the value, the more similar the two documents are to each other. The highest score it can get is 1 when the two documents are identical, and the minimum score is 0 when there is no overlapping term between the two documents. The *Dice*, *Jaccard* and *Overlap* similarity measures are similar to cosine. Despite using different normalization methods, they all use dot product to measure the similarity between documents [54].

The other popular similarity measure is called Euclidean distance which is defined as

$$dist(d_i, d_j) = \| d_i - d_j \| = \sqrt{\sum_{k=1}^{n} (t_{ki} - t_{kj})^2} \tag{2.6}$$

The most prominent limitation of these two measures is that both of them ignore the potential correlation between different terms and the document is deemed as a "bag of terms", i.e., the order of the words is not used.

There are other similarity measure variants in the context of term vector representation. The reason why we pick out these two is solely based on the fact that they are among the most popular ones and have been proved to be effective experimentally. Because essentially all these term vector similarity measures provide the same amount of information to clustering algorithm, it has been mentioned in the literature [54] that there are no appreciable performance difference between them.

Researchers also try to tackle this with different perspectives. Recognizing term vector model measures' assumption about terms independency, the other classes of measures take advantage of the co-occurrence of term pairs to compute the similarity [58]. Due to its requirement of large text corpus, this type of similarity measures is not suitable for query-specific clustering. Another class of similarity measures takes the syntactic relations between terms into consideration [52].

2.4.2 Document Clustering Techniques

From the perspective of the output, document clustering can be divided into two types: *hierarchical clustering* and *partitional clustering* [54].

Hierarchical Clustering

As its name hints, hierarchical clustering results in a tree structure hierarchy. The taxonomy graph sometimes is called a dendrogram. In the hierarchy, nodes represent clusters. Lower level clusters are nested in higher level clusters. The top node includes all nodes. Leaves (nodes on the lowest level) refer to singleton clusters consisting of individual documents. Different levels in the hierarchy reflect different similarity granularity. Small clusters of documents that are found to be strongly similar to each other are nested within larger clusters that contain less similar documents [65]. In the field of text clustering, there are generally two different types of techniques to produce a hierarchical clustering:

Agglomerative A document-pair similarity matrix is required to be created a priori. Each document starts as a singleton cluster. This type of algorithms keeps merging the pair of most similar clusters (update the similarity matrix after combining them) until there is only one cluster left. In this approach the hierarchical structure is created in a bottom-up fashion.

Divisive Starting by putting all documents into one cluster, this technique breaks down clusters into more and more smaller sub-clusters until each cluster is comprised by only one document. So a divisive approach constructs clustering in a top-down manner which is exactly the opposite of agglomerative method.

Since agglomerative approach is much more commonly used in document clustering field, we won't go into more details about divisive clustering. One of the most important reasons for researchers to choose agglomerative over divisive is that divisive is more computationally expensive.

Common agglomerative clustering algorithms include *single linkage*, *complete linkage* and *group average linkage*. A generic agglomerative clustering algorithm can be described as follows:

Hierarchical Agglomerative Clustering Algorithm (HAC) Framework

1. Choose the pair-wise similarity measure between documents.
2. Compute the doc-doc similarity matrix, e.g., the similarity between documents d_i and d_j is stored by entry m_{ij} in the similarity matrix.
3. Combine the two clusters (documents) with the greatest similarity.
4. Update the similarity matrix to reflect the pair-wise similarities between the new cluster and other clusters (documents).
5. Repeat step 3 and 4 until there is only one all-inclusive cluster left.

Similarity measure between document points and clusters need to be determined in order to run an agglomerative clustering algorithm. Normally the only real difference between various HAC algorithms is how they choose which pair of clusters to merge. Commonly used inter-cluster measures include:

- $D_{MIN}(C_i, C_j) = \min_{d_i \in C_i, d_j \in C_j} Sim(d_i, d_j)$
- $D_{MAX}(C_i, C_j) = \max_{d_i \in C_i, d_j \in C_j} Sim(d_i, d_j)$
- $D_{UPGMA}(C_i, C_j) = \frac{\sum_{d_i \in C_i, d_j \in C_j} Sim(d_i, d_j)}{|C_i||C_j|}$

where $\mid C \mid$ is the size of cluster C and $Sim(d_i, d_j)$ indicates the similarity between document d_i and d_j. $D_{MIN}(C_i, C_j)$, $D_{MAX}(C_i, C_j)$ and $D_{UPGMA}(C_i, C_j)$ are used in complete-linkage, single-linkage and UPGMA (Unweighted Pair Group Method with Arithmetic Mean) HAC algorithms respectively.

HAC algorithms usually have a time complexity higher than $O(n^2)$. Because they are computationally more expensive than partitional clustering methods (they generally have a time complexity of $O(n)$), HAC algorithms are not good candidates in traditional static clustering which is computed over the entire document collection. However, its computational disadvantage can be ignored when it comes to query-specific clustering due to the small size of document sets (documents or document summaries corresponding to the top n results in response to a query). Furthermore we agree on the point brought up by Tombros et al. [61]:

> ... effectiveness is of primary importance, whereas efficiency is a factor that is heavily dependent on technological advances. One may also view the improved effectiveness that can be achieved as a motivation for the development of more efficient algorithms ...

Partitional Clustering

Instead of generating a tree-like dendrogram, partitional clustering scheme produces a number of disjoint flat clusters. It first attracted researchers because of its low computational requirements, which are normally in the order of $O(N)$ to $O(N \log N)$ [56].

There are a lot of different partitional clustering algorithms. However basically the common framework of partitional clustering algorithm can be summarized as follows:

Partitional Clustering Algorithm Framework

1. The number of clusters desired is determined a priori.
2. Randomly choose k (the number of clusters desired) instances as seeds. Each seed represents an initial cluster.
3. Allocate each instance to a cluster to optimize a predetermined criteria.
4. Iterate step 3 until clustering converges or other stopping criteria are met.

One of the most widely used partitional clustering methods is K-Means [9]. In K-Means clustering scheme, documents are represented by multi-dimensional vectors. The distance (or similarity) between individual document and a cluster is measured by the distance (or similarity) between cluster *centroid* and document. Given a

collection of documents A which is defined as

$$A = \{d_1, d_2, ..., d_k\}$$

the centroid C_A is (vector addition)

$$C_A = \frac{1}{|C|} \sum_{i=1}^{k} d_i \tag{2.7}$$

The centroid is the mean of all points in a cluster. K-Means document clustering algorithm is briefly described here:

k-Means Clustering Algorithm

1. Select k document vectors as seeds. Thus the seeds are also the centroid of the initial clusters.
2. Assign each document to the nearest cluster. This is usually measured by calculating the distance between the document vector and the cluster's centroid vector.
3. Update the centroid of each cluster.
4. Iteratively repeat step 2 and 3 until the clustering does not change any more or other criterion is met.

Depending on the different approaches of getting initial document seeds, K-Means has a number of variants. Due to the randomness of the initial seeds generation, the clustering results are generally nondeterministic (different runs of clustering produce different clustering solutions).

It is fairly easy to transform a partitional clustering algorithm into a hierarchical clustering algorithm through recursively partitioning the clusters obtained from the last run of the partitioning algorithm. The tree-like structure (dendrogram) can be constructed in a top-down fashion by the iterative application of partitioning algorithms. Hierarchical agglomerative clustering (e.g., simple-linkage, complete-linkage) always produces the same taxonomy for the source document set. On the contrary generally partitional clustering methods will yield different results for each run of the algorithms.

Other Clustering Methods

To gain the best sides from both classes of clustering methods, some hybrid clustering algorithms were proposed by researchers. For example, Cutting et al. [9] devised a buckshot algorithm which combines group-average HAC and K-Means approach.

They used HAC to generate the initial seeds for K-Means based on a sample of instances of $\sqrt{n}$ which results in $O(n)$ time complexity. Recently Zhao and Karypis introduced the "Constrained Agglomerative Clustering" which outperformed some commonly used partitional and agglomerative hierarchical clustering methods [70] in terms of the quality of the produced hierarchical clustering. Constrained Agglomerative Clustering is a class of hybrid clustering methods that uses partitional clustering to generate the initial data partitions and then applied HAC algorithms on top of them. Finally HAC is again used to combine the subtrees into the complete tree-like hierarchy.

One of the other classes of document clustering methods worth mentioning is graph-based document clustering method. The relationship between terms and documents is modelled by graphs. Instead of using multi-dimensional vectors to represent documents, vertices in the graph are used to denote documents. The edges connecting vertices are usually used to either express the similarity between documents or the relationship between terms and documents. However Zhao and Karypis [69] found in their comparative experiments that the performance of this class of document clustering methods was inferior to VSM (Vector Space Model) approaches.

2.5 Related Work

Clustering techniques have been widely used in information retrieval to improve document search and retrieval for a long time. They were initially adopted for the sake of efficiency. It is assumed that the time needed for checking the information request against each document (*near-neighbor* or *similarity search*) in a huge document collection would be excessive; by grouping similar documents together, the process of retrieval can be accelerated by comparing the information request to each group's representation (centroid vector of the group) [56]. Cluster search is used to enhance similarity search which is also motivated by the thought that by automatically grouping similar documents together the recall of retrieval can be improved. Typically document collection is clustered into either flat partitions or hierarchical components. Two methods have been developed to match an information request against the document hierarchy achieved by statically clustering document collection to find the best cluster candidate: *top-down search* and *bottom-up search*. The

theoretical soundness of this paradigm relies on Van Rijsbergen's Cluster Hypothesis [54]:

> ... documents relevant to a request are separated from those which are not relevant, i.e., that the relevant documents are more like one another than they are like non-relevant documents.

However results derived from studies done to determine whether cluster search outperforms similarity search have been inconclusive at best if not conflicting. Although some studies have shown cluster search did increase the effectiveness of retrieval [65], other research endeavors found that it can be inferior to similarity search paradigm [39]. The quadratic time complexity associated with HAC algorithms also greatly limits the popularity of adopting clustering technologies as an alternative to the traditional near-neighbor search.

Under the concept of the pre-categorized clustering search, Cluster Hypothesis is practically constrained to the following: If document d_1 and d_2 are both either relevant or irrelevant to query q_1 they MUST also be either both relevant or both irrelevant to new query q_2. This boils down to the question: Document collection is static, but should we also perceive "being similar" as a static concept? We believe the answer is no. As Hearst and Pedersen said in [17]:

> ... because documents are very high-dimensional, the definition of nearest neighbors will change depending on which neighbors are on the street ...

Due to the high dimensionality of document, one document may be deemed "similar" to another document in one circumstance but on other occasion they may appear to be irrelevant to each other. In the domain of Web clustering research circumstance is defined by user queries. Each user-defined information request (query) can be interpreted as a unique angle to "read" the document. Moreover, by tailoring the document set to the query, the chance of placing relevant documents in the same cluster is increased [17]. Tombros, Villa and Rijsbergen have experimentally proved that effectiveness of static clustering (clustering over the whole document set) is consistently inferior to the effectiveness of query-specific clustering [61].

Cutting et al. [9] first proposed using document clustering techniques as an information access tool in its own right. They implemented the "Scatter/Gather" document collection browsing system based on dynamic clustering. The system allows users to navigate between different levels of details in the collection. In order to facilitate the online clustering requests, they also introduced two linear time partitional clustering algorithms: *Buckshot* and *Fractionation.* Buckshot is a K-Means and HAC hybrid algorithm and Fractionation is a HAC-like fast linear algorithm.

Hearst and Pedersen are among the earliest researchers that explored using clustering method to view retrieval results [17]. Instead of applying clustering directly to the entire document collection offline, they opted to cluster the documents retrieved as search result in response to a query dynamically.

Allen, Obry and Littman [3] developed an interactive interface that can be used to explore the search result document set structure. The results are represented by a visual dendrogram generated by parallel and distributed clustering in their system. Their target document corpus was a set of 25,625 articles from the Academic American Encyclopedia and Ward's hierarchical clustering algorithm was employed. They found clustering was most helpful for users when the returned documents fell into categories naturally. The result presented in the paper is highly descriptive and they did not give statistics data to validate their conclusion though.

Kural, Robertson and Jones [42] investigated the validity of improving retrieval precision by clustering retrieval result. They clustered the search result returned from the Okapi probabilistic search engine and recruited graduate students to perform user study. They used precision as the performance measurement. In contrast to Hearst and Pedersen's conclusion that users are capable of identifying the optimal cluster [17], their experiment showed that users were not always able to spot the best cluster. Interestingly they found users tend to be able to find the worst cluster (most irrelevant cluster) which can be used to improve the precision of ranked list through excluding the last-ranked cluster. They also argued that using best precision clusters in assessing clustering solution against ranked list gives the clustered output an unfair advantage. They pointed out that although in their experiments optimal clusters did outperform ranked list in terms of the precision measurement,

there is little practical significance; they explained that the smaller the cluster sizes the higher the chance of outperforming the ranked list. The reasons are listed as follows [42]:

- The effect of divergence is more pronounced: one extra relevant document makes a bigger difference to precision in a set of six documents than in a set of 20 documents;
- Given a fixed number of documents, the more clusters there are, the more choices to select from.

We believe that the sensitivity of clustering effectiveness to the cluster size exhibited in their experiments is mainly caused by their solely reliance on precision for measurement. An extreme example is we can achieve a "100%" precision by simply treating each document as one cluster. They used C3M clustering method and did not compare its performance to other more widely used ones. So it is possible that their choice of clustering algorithm may have affected their results too.

Partitional clustering algorithms used to be considered inferior to hierarchical clustering algorithms in terms of retrieval effectiveness. However recent researches [60, 70, 69] suggested that partitional clustering actually consistently performed better than agglomerative methods with respect to cluster quality. Steinbach, Karypis and Kumar [60] studied a variant of K-means, "bisecting" K-means, and found that it outperformed the hierarchical clustering algorithms in their experiments. They also offered an explanation as to why HAC performed poorly in their work. They attributed the success of "bisecting" K-means to the ability of partitional algorithms to recover from "bad" clusters. According to their work a significant portion of one document's nearest neighbors are of different classes. K-means clustering bypasses this nature of documents by looking at "global" property (the similarity of documents to all other documents in the same cluster). We also want to mention that previous studies that show HAC is more effective than partitional clustering were mostly carried in the context of static clustering over the entire document collection instead of query-specific document set.

Zamir et al. [68] came up with several novel text hierarchical agglomerative clustering algorithms and used them to cluster retrieval results on the Web. The

rationale behind their algorithms is that documents grouped into same clusters always share a set of words or phrases. They found that their algorithms performed consistently better than the widely used group-average HAC method. However the way they created their test collections is questionable. They got their test collections by merging 1 to 8 randomly chosen base collections which are search results in response to different queries. Thus the test collections basically contain distinctly different document sets. This characteristic of the test collections differs them from the document set obtained by sending a query to the search engine, because although the query result documents often reflect different aspects of one query, they are not completely unrelated. So the inherent heterogeneity in the test collection actually lessens the challenge presented to the clustering methods. The other limitation of their algorithms is some documents can be left unclustered.

To the best of our knowledge, Zamir and Etzioni [67] are the first to study application of clustering technique on post-retrieval results in the domain of the Web. They implemented MetaCrawler-STC, a clustering Web search engine prototype on top of MetaCrawler [31]. They proposed the Suffix Tree Clustering (STC) algorithm which is both linear time and incremental. STC differentiates itself from other popularly used clustering algorithms by allowing overlapping clusters (documents can appear in more than one cluster). STC utilizes proximity information among terms inside a document and groups documents together based on the common phrases shared among them. They claimed that STC outperformed Group-Average HAC as well as some other linear time clustering algorithms in terms of precision improvement over the ranked list. Their evaluation methodology is based on the assumption that was made by [17]: users are always able to select the cluster with the highest relevant document density. They did not mention how they came up with the ten queries that they used to construct the document sets in their experiment. MetaCrawler-STC always presents the user with 10 top scored clusters which leaves documents not being included in those clusters "invisible" to the user. No explanation was given regarding why "10" was chosen.

Macskassy, Banerjee, Davison and Hirsh carried out the user experiment on how human performed clustering on Web pages [43]. Their results revealed users tend to show little in common with their manually built clusters. They concluded

that it is questionable whether effective clustering can be achieved based on mere knowledge of query. Kural, Robertson and Jones also quoted this as a disadvantage that harms search output clustering effectiveness [41]. As opposed to the above assertion, we believe that the user study can only lead to the conclusion that there is no universally best clustering. It will be too assertive to infer that since the grouping does not perfectly reflect individual user's own perception of document clustering, the clusters are just misleading or useless to users. We acknowledge the fact that clustering as a means of classification is actually very subjective. Yahoo's Web directory [35] is constructed by hundreds of human compilers and indexers. Hence it is a consistent integration of subjectivity which proves to be one of the most popular Web directories that helps millions of Internet users. Good clusters do not have to meet each user's individual expectation completely which is also impossible. As long as they are topically coherent and can functionally help users better identify relevant documents and disregard unrelated documents, they should be deemed as satisfactory.

The work conducted in this research is to investigate the effectiveness of post-retrieval (query-specific) clustering in the domain of the World Wide Web. We proposed a keyphrase extraction algorithm using HTML formatting elements and explored its application in retrieval results clustering. Through experimental evaluation, we studied and compared the variation of effectiveness of clustering influenced by different representations and n (number of top ranked retrieved documents) as well as its effectiveness against the traditional ranked list. Our preliminary experimental results suggest that query-specific clustering can be more effective than the ranked list and keyphrase based representation of documents can noticeably increase the quality of document clustering over other commonly used representations.

Chapter 3

Phrastractor

3.1 Overview

Phrastractor is an online keyphrase extraction prototype system that demonstrates our keyphrase extraction algorithm: *Extoken*. Extoken is a keyphrase extraction algorithm tailored specifically for Web documents. It combines the statistical lexicon information with the HTML formatting elements to rank keyphrases. Unlike keyphrase extraction algorithms based on machine learning technique [62, 66], it does not require training data. Neither does it ask users to provide cryptic input parameters. In addition to using the traditional bag-of-terms paradigm of document representation, Extoken takes advantage of the ubiquitous HTML syntactic tags in HTML documents to better calculate the keyphrases that capture the topics of the documents. The HTML elements (e.g., <head>,<title>,<a href=...> ...) are used to format the HTML documents, however, we believe they also potentially communicate important semantic information related to the topics of the documents. Different formatting often represents how important the particular chunk of text is in serving the main points of the source document. Thus it would be a huge waste not to use this readily available information to extract keyphrases out of Web documents.

Keyphrase extraction on the Web can be very useful considering its potential applications. It can be used to generate concise document abstraction as a replacement or enhancement to the current document summarization or snippet offered in the ranked documents list by major search engines. It can potentially help users better decide whether the document is relevant or not. It can also be used to help users refine their queries. A lot of times users either lack the expertise to pick out the right words to describe their information needs or simply do not want to spend too much time contemplating the proper queries. This partially explains the user search pattern observed in [38]: Instead of further searching through the following search return pages, users only view the first one or two pages returned by the search engines. Partially relevant items located in the first one or two pages are often used as a jumping off point to find relevant items. So the keyphrases associated with each document in the hit list can help users learn about a topic and work as hints to help users find the right words to describe their needs more precisely. Users then will

be able to make more informed modification to their queries to retrieve documents they expect.

In the following sections, we first use some examples to demonstrate how users can take advantage of the functionality of Phrastractor, our online keyphrase extraction system. Then we describe the design of the algorithm (i.e., *Extoken*) that powers the system.

3.2 Examples of Using Phrastractor

Phrastractor is the online keyphrase extraction system that we developed to demonstrate the effectiveness of the Extoken algorithm. It extracts keyphrases from Web pages and provides necessary GUI components for users to interact with the keyphrases in terms of forming queries. We believe good keyphrases of the Web page can not only be regarded as high-quality topical summarization but also be used as good query terms to retrieve the source document from the Web. More about this will be discussed in Chapter 5.

Figure 3.1 shows the front page of Phrastractor. Firstly users need to specify the target Web page's URL. It also has two input parameters that users can change in order to customize Phrastractor's output. The combobox provides a means for users to change the ranking scheme used by Extoken to extract keyphrases. Users can choose from *HATF*, *HLOGN*, *FREQ* and *Meta Keywords*. The first three schemes will be discussed in the following section. *Meta Keywords* will direct Extoken to extract the keywords embedded in the page's HTML meta tags if they are available. The default scheme is *HATF*. Users can also specify the number of keyphrases (n value) they want Phrastractor to output. Then the top-ranked n keyphrases yielded from Extoken would be picked.

Suppose the user wants to extract keyphrases from Prof. Dekang Lin's Web page (http://www.cs.ualberta.ca/~lindek/). He will type the URL in the address box in the page and click the "EXTRACT" button to initiate the keyphrase extracting process. The result page will be something like what is shown in Figure 3.2. Since users did not change the default, "HATF" term ranking scheme is used and the top-5 ranked keyphrases (because of ties, there might be more than 5 keyphrases) are displayed (For comparison purpose, keyphrases extracted by using HLOGN and

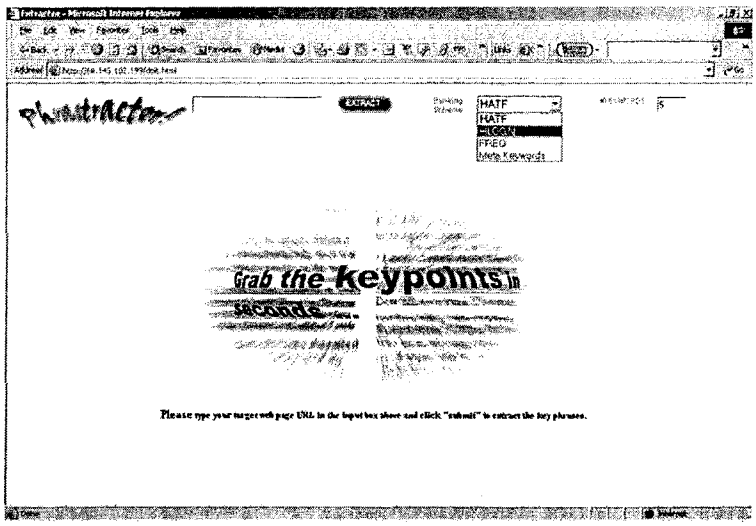

Figure 3.1: Phrastractor front page.

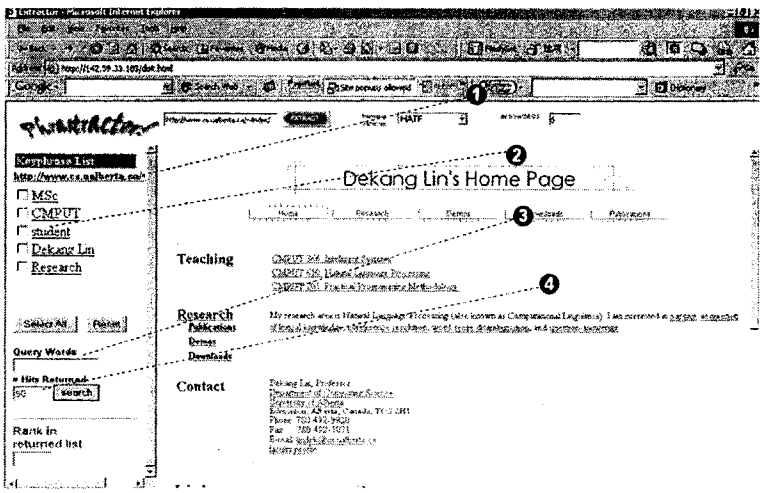

Figure 3.2: Phrastractor result page for HATF scheme.

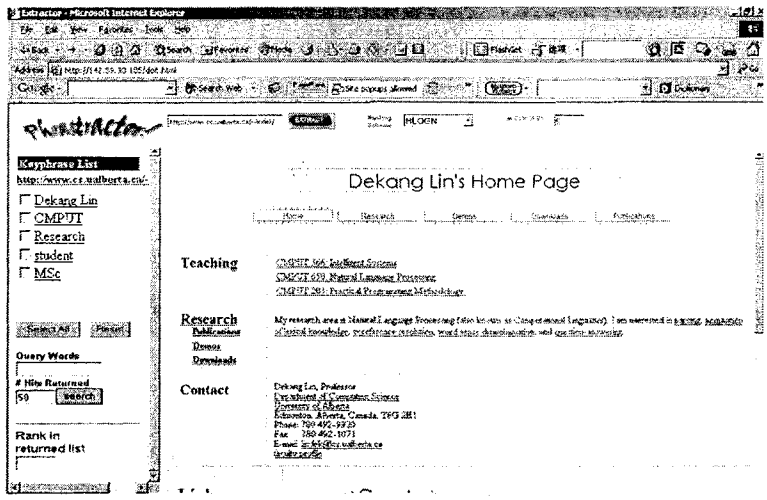

Figure 3.3: Phrastractor result page for HLOGN scheme.

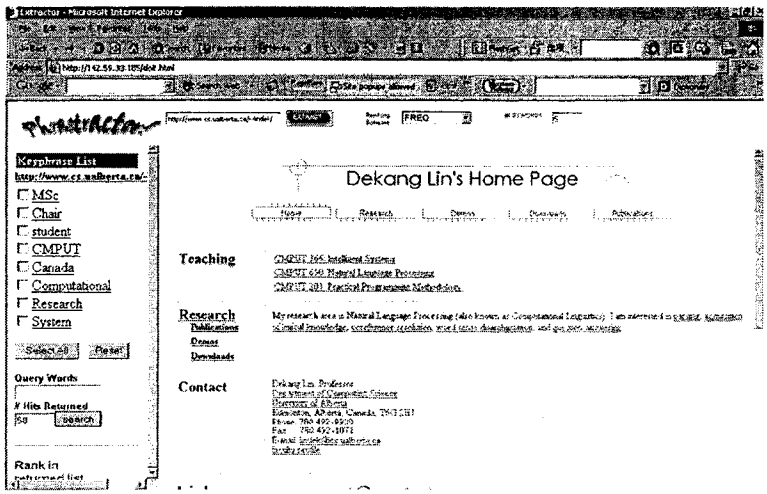

Figure 3.4: Phrastractor result page for FREQ scheme.

FREQ are shown in Figure 3.3 and Figure 3.4). The client area of the browser is split into two panes. The left pane shows the extracted keyphrases from the Web page and the right pane displays the actual Web page. Key parts of the page have been annotated with numbers and we describe each of them as follows:

1. It displays the URL of the page shown in the right pane.

2. This is the list of keyphrases extracted by Phrastractor. We can see that "MSc", "CMPUT", "student", "Dekang Lin" and "Research" are displayed as keyphrases for this page. By browsing the extracted keyphrases, a new user to this page can gain a rough idea of what the page is about. If users click on any of the keyphrases, that particular keyphrase will be sent to Google as the query to retrieve more results related to that phrase. The query results will be displayed in the right pane. Phrastractor also provides means for composing query terms consisting of more than one keyphrase: users can toggle on the checkbox to the left of each keyphrase to add it in a query. After selecting all the wanted keyphrases, users can hit the "search" button below to submit the composed query to Google. Suppose users want to know more about Prof. Dekang Lin, so they click "Dekang lin" in the keyphrases list. Figure 3.5 shows what they will see: the query results are displayed in the right pane. Each returned result document entry is comprised of the page title, snippet and page URL. We give description of some visual elements in Figure 3.5 as follows:

   (a) Page entry highlighted in red indicates that this is the exact source page based on which Phrastractor extracted the keyphrases. If it is among the top-ranked pages, the keyphrases that are used can be considered as good query terms to retrieve the page.

   (b) If the page shares the same top domain as the source page, its entry will be highlighted in pink color. For example, in Figure 3.5, since our original Web page's URL is "www.cs.ualberta.ca/~lindek", the second URL "www.cs.ualberta.ca/people/faculty/lindek.html" is displayed in pink.

   (c) This box shows the actual rank position of the source page in the retrieval results from Google. "-1" indicates that the source page is not in the

returned page list. In Figure 3.5, the source page is the first one in the returned ranked list.

3. This box displays the query composed by the user. As the user checks on and off keyphrases, the resulting query is updated here.

4. Users get to specify the number of hits that they want to get from Google in this box.

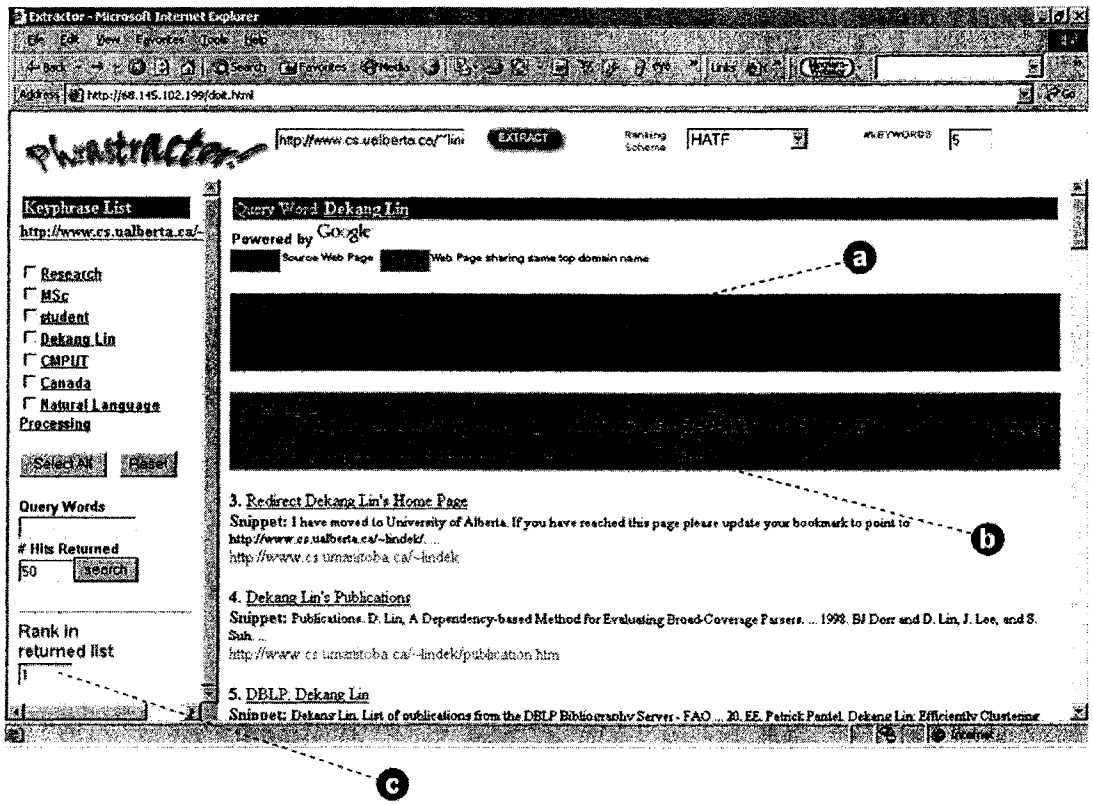

Figure 3.5: Search result of keyphrases query.

We give another example here. Suppose users put "www.smarttech.com", the URL of "SMART Technologies" (the leading manufacturer of the interactive whiteboard), in Phrastractor and click "Extract". Figure 3.6 shows the extraction result. The keyphrases listed are "SMART", "Interactive", "Whiteboard", "Interactive Whiteboard", "SMART Board" and "SMART Technologies Inc". From the keyphrases, users may get the main topic of the page quickly: about interactive

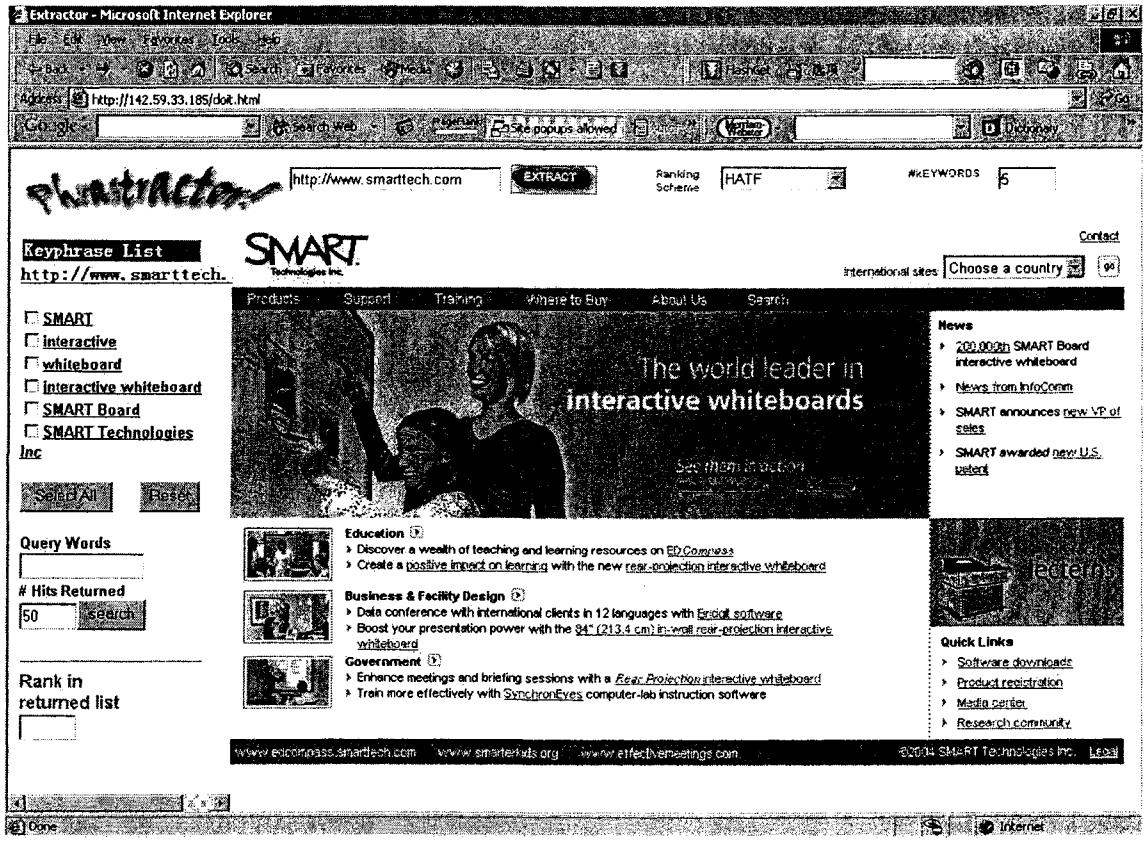

Figure 3.6: Keyphrases extraction results of example #2.

whiteboard and SMART Technologies Inc. Users might want to search "interactive whiteboard" on the Web, so they click that keyphrase. The query result is shown in Figure 3.7. We can see that the original page is the second ranked URL and there are two other pages in the result that come from the same web server "www.smarttech.com".

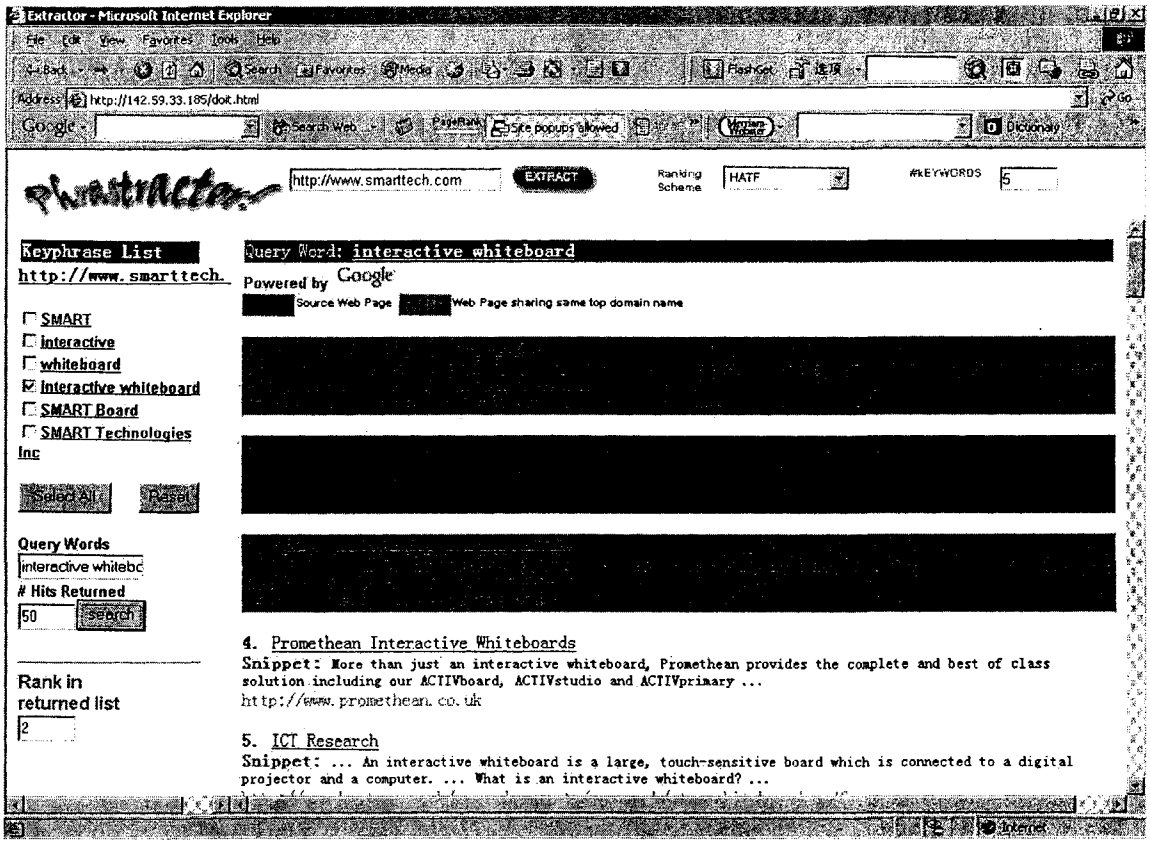

Figure 3.7: Search result of example #2.

3.3 The Extoken Algorithm

The overall architecture of Extoken is outlined in Figure 3.8. Although Extoken was designed primarily for HTML document, it also can handle plain text document. It accepts both local file and Web document (based on the corresponding URL). In order to facilitate Web document clustering, it can process a list of URLs as input and it will automatically download all the Web documents according to the URLs and process them locally. WordNet (more information regarding WordNet is given

later) and snippet information are used only when Extoken is used in the context of Web document clustering.

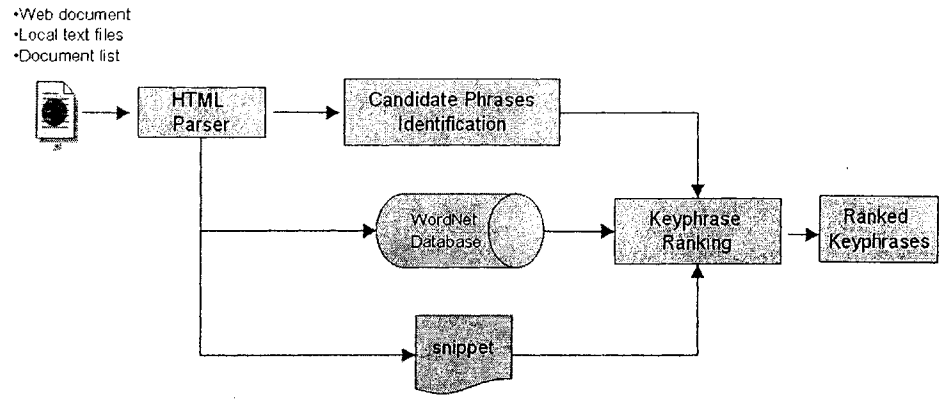

Figure 3.8: Extoken structure.

In the following sections we describe Extoken at a conceptual level for clarity purpose. The actual software can be regarded as an efficient implementation of the algorithm.

3.3.1 HTML Parser

Before starting to parse, the HTML parser has to tackle the HTML frameset. In HTML, frameset is defined by the tag <frameset>. It is used to organized multiple frames and each frame contains one document. When multiple pages are presented to users in frameset, users get to see all pages at once, so when frameset is detected, Extoken HTML parser would download all pages contained in the frameset and put contents from different pages into one composite document for further parsing. This is handled in a recursive manner so a frameset in a frameset can also be dealt with. We also noticed that a lot of Web pages on the Internet are poorly formatted (e.g., missing tags, unmatched nested pairs of tags ...), so fault tolerance has to be addressed in the design of the parser too. After one pass of the source document, the information output by the HTML parser is illustrated in Figure 3.9.

We explain the output as follows (we give an example after the description of the output):

HTML Meta Information The following HTML tags as well as associated text

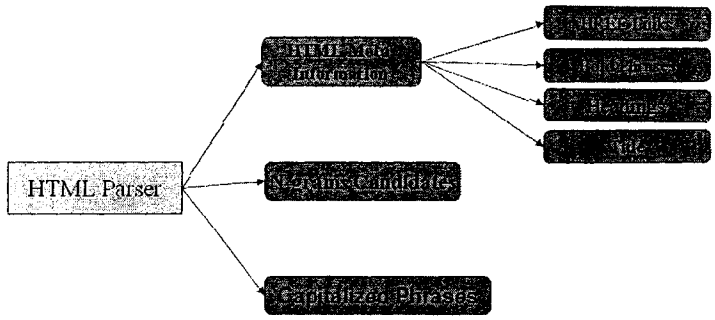

Figure 3.9: HTML parser output.

are stored to aid keyphrase ranking:

- HREF links (<a>). Links inside one document are endorsements to the pages they point to; the referenced documents most likely have contents that are related to the topic of the source document. Thus the text of hyperlinks in HTML document is deemed useful.
- Bold/Big font (<b>, <big>). The author of a HTML document usually stresses certain text by making it in bold or bigger font. Hence text rendered by such formatting is more likely to be related to the document topic.
- Head tags (<h1> to <h6>). Head tags are used to structure the HTML document, and they usually function as the headings of sections of text. Therefore they can deliver important topical information.
- Title. Content enclosed by the title tag of the HTML document normally reveals the main theme of the document.

N-grams candidates Term (word) sequence is also known as n-gram where n denotes the length of the phrase. For example, "information retrieval" is a 2-gram while "experiment" is a 1-gram. N-grams as well as their corresponding occurrence frequencies are calculated by HTML parser. This is carried out in a case-insensitive manner. The n-grams generation process can be described as follows:

1. During the first pass, HTML parser extracts tag tokens which correspond to the

text enclosed in each pair of tag elements. For example, there are in total 4 tag tokens in the following HTML snippet:

```
<h1>Abstract</h1><p>This is part1.</p><h1>Introduction</h1><p>Here
goes introduction.</p>
```

In the meantime, HTML parser also calculates the frequencies for all 1-grams. Stop words and terms consisting of all non-alphabets are discounted from 1-gram.

2. Each tag token is further broken up into sentences by identifying sentence boundaries (e.g., punctuation marks such as ",.:;?!" and new lines).
3. Apostrophes along with other symbols listed below are replaced with spaces in sentences.

$$\backslash t()[]/\backslash\text{“”}' - \{\} <> \quad (\backslash t \text{ is tab})$$

4. N-grams ($20 \geq n \geq 2$) and their occurrence frequencies are generated based on each sentence by a pass through all extracted tag tokens. Extoken uses a window structure as in [47] to extract n-grams. For instance, from sentence "web document clustering", we can generate two 2-grams: "web document" and "document clustering", and one 3-gram: "web document clustering". Next, pruning is carried out. First, trivial n-grams (n-gram occurring only once in document) are discarded. Second, only the maximal length n-grams are kept. Maximal length n-gram is defined as n-gram that has no other n-gram of the same occurrence that subsumes it. For example, if "computer science" and "computer science department" both occur 4 times, "computer science" won't be of maximal length.

Capitalized Phrases Capitalized phrase is defined as n-gram ($2 \leq n \leq 10$) with each word meeting one of the following criteria:

1. The first letter is capitalized.
2. The whole word is uppercase.
3. Neither of the above two but the word is in the following set:

$$\{\text{“}of\text{”}, \text{“}on\text{”}, \text{“}at\text{”}, \text{“}and\text{”}, \text{“}\&\text{”}, \text{“}the\text{”}, \text{“}for\text{”}\}$$

For instance, "International Business Machine" and "Department of Computer Science" are both capitalized phrases. However even when a phrase meets the above requirements, it may not be a desired capitalized phrase. For example, in "Actually I do ...", "Actually I" is not desirable, although it is legitimate according to the above rules. A number of heuristic rules were developed to improve the algorithm to better handle situations like this.

Suppose we have a HTML document as follows:

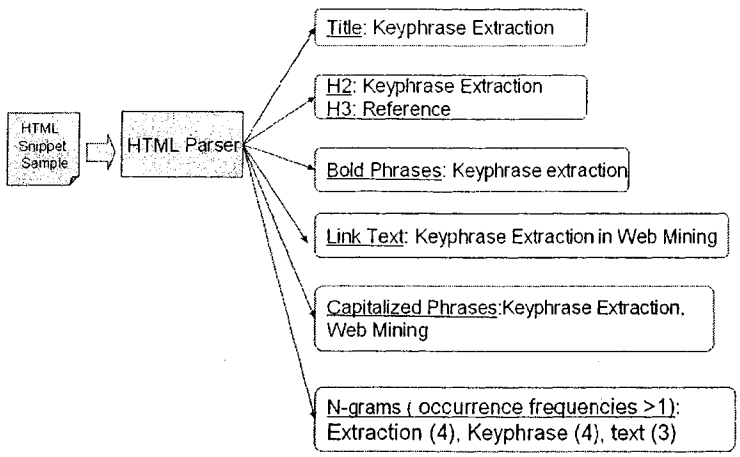

Figure 3.10: HTML parser output example.

```
<title>Keyphrase Extraction</title> <h2>Keyphrase Extraction</h2>
<b>Keyphrase extraction</b> is the process of topical phrases
identification in the text document. Keyphrases can be used in a
variety of purposes, including summarization of text, labelling
and text clustering. <h3>Reference</h3> <a
href="http://webmining.org/paper.html">Keyphrase Extraction in Web
Mining</a>
```

Figure 3.10 shows the parsing result of running the HTML parser on the above HTML snippet. The number in the parenthesis associated with each "N-grams" phrase is its occurrence frequency.

3.3.2 Candidate Phrases Identification

Based on the output from HTML parser, Extoken performs the following procedures to collate results:

1. **Find promising capitalized phrases.** Extoken uses the following rules to identify promising capitalized phrases:
   - Capitalized phrases appearing in headers.
   - Capitalized phrases in bold/large fonts.
   - Capitalized phrases contained in hyperlink text.
2. **Clean out n-gram candidates ($n \geq 2$).** Stop phrases are filtered out. Stop phrases are n-grams that are completely comprised of stop words. Trailing

Formula	Name	Abbr.
$K \times H_i + (1 - K) \times (\frac{f_i}{f_{max}})$	HTML-aided Augmented Normalized Term Frequency	HATF
$K \times H_i + (1 - K) \times \frac{1+\log f_i}{1+\log f_{max}}$	HTML-aided Normalized Log	HLOGN
$\frac{f_i}{f_{max}}$	Normalized Frequency	FREQ

Table 3.1: Keyphrase weighting schemes.

stop words are also trimmed for n-grams. If after the trimming, n-gram ($n \geq 2$) becomes 1-gram, it should be discarded since 1-grams have already been processed separately.

3. **Readjust the occurrence frequency of n-gram (n $\geq$ 2).** This is to handle the following scenario: if the document has three "a b c", two "a b d" and one "a b", before the readjustment, the frequency count of "a b" is 6. This step would adjust its frequency count to 1.

4. **Merge 1-gram and n-gram (n $\geq$ 2).** Redundant 1-grams as well as 1-grams with occurrence frequency less than 2 are discarded. Redundant 1-gram is defined as 1-gram that is not of maximal length; this means there is at least one n-gram ($n \geq 2$) that subsumes this 1-gram and has the same occurrence frequency. The rationale behind this step is that if whenever a particular word appears, it appears in a word sequence, it is most likely that the word itself only serves as a part of the word sequence as a whole.

5. **Handle inflectional morphology.** In particular, Extoken converts plural to singular for 1-grams and recalculate the occurrence frequency for the term. For instance, if *cats* has a frequency of 3 and *cat* occurs 2 times, Extoken removes the entry for *cats* and bumps up the occurrence frequency of *cat* to 5.

3.3.3 Keyphrase Ranking

Keyphrase ranking gives scores to keyphrases based on their within-document frequencies and relevant HTML meta information. Such a ranking system also allows Extoken be able to yield varying numbers of keyphrases. Extoken implemented three different keyphrase weighting schemes which are presented in Table 3.1. K ($0 \leq K \leq 1$) is the coefficient that is used to adjust the weight attached to HTML

meta information associated with the phrase and normalized within-document frequency of the phrase respectively. In the implementation of Extoken, 0.5 is assigned to K which means that HTML formatting meta information and normalized within-document frequency contribute to the final score of the phrase with equal weight. Raw within-document frequency is always normalized in our formula because we want to reduce the influence of occurrence frequency on the ranking score (if in document d, p_i appears 20 times and p_j occurs 2 times, it does not necessarily mean that p_i is ten times as important as p_j).

HATF This keyphrase weighting scheme is inspired by $ATF1$ formula [7]. f_i is the number of times p_i appears in the document and f_{max} is the maximum frequency of any word in the document. H_i is the HTML meta information score of phrase p_i. H_i is defined as:

$$H_i = \frac{b_b w_b + b_c w_c + b_t w_t + b_l w_l + b_h w_h}{w_b + w_c + w_t + w_j + w_h} \tag{3.1}$$

where w_b, w_c, w_t, w_l, w_h are the relative weights associated with bold/large font, capitalized phrase, title, hyperlink text and heading respectively and b_b, b_t, b_l, b_h, b_c are binary values indicating whether p_i is in bold/large font, title, hyperlink text, heading and is capitalized phrase. In Extoken, $w_b = 2, w_c = 3, w_t = 5, w_l = 2$ and $w_h = 3$. These values were obtained heuristically in our research and our experiments have proved that they work very well. For instance, if "Computer Science" appears in bold font and heading and occurs 3 times and the maximum frequency of any word in the document is 8, Extoken calculates the score for "Computer Science" as follows: Firstly

$$\frac{f_i}{f_{max}} = \frac{3}{8} = 0.375$$

secondly

$$\frac{b_b w_b + b_c w_c + b_t w_t + b_l w_l + b_h w_h}{w_b + w_c + w_t + w_j + w_h} = \frac{1 \times 2 + 1 \times 3 + 0 \times 5 + 0 \times 2 + 1 \times 3}{2 + 3 + 5 + 2 + 3} = 0.53$$

So finally we have

$$0.5 \times 0.53 + 0.5 \times 0.375 = 0.4525$$

as the HATF score of phrase "Computer Science".

HLOGN HLOGN is designed based on LOGN [4]. Logarithm is the other common technique used in IR to normalize raw term frequency. H_i in its formula is calculated in the same way as in HATF. Using the same example that we used for HATF, we can calculate the HLOGN score for "Computer Science" as follows: First

$$\frac{1+\log f_i}{1+logf_{max}} = \frac{1+1.1}{1+2.1} = 0.68$$

second (this is the same as in HATF)

$$H_i = 0.53$$

Finally the HLOGN value is

$$0.5 \times 0.53 + 0.5 \times 0.68 = 0.605$$

FREQ In this scheme, HTML meta information is not used and the weighting scheme is solely based on normalized frequency of the phrase. This scheme is mainly provided as a comparison to the other two as we wanted to comparatively study how HTML-aided method stacks up with traditional statistics based method.

3.3.4 WordNet and Snippet Based Expansion

In this research, we were mainly interested in the application of keyphrase as a technique in the context of query-specific Web page clustering. Thus according to the characteristics of the application, we introduced WordNet and search engine generated snippet into Extoken to expand the keyphrase output.

WordNet [46] is a lexical reference system that distinguishes itself from conventional lexicon by the fact that it is organized by semantic relations. Word meaning in WordNet is represented by synonym sets which are called synsets. Different relations between synsets form the network structure of WordNet. The most common relations used in WordNet include synonymy, antonymy, hyponymy, and meronymy. Since synonymy and hyponymy are used in Extoken, we need to describe them in further details here. The definition of synonymy is described as [46]:

> two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value.

For instance, *auto* is synonymous to *car*. Obviously synonymy is symmetric. Hyponymy represents the "*x is a kind of y*" relation between words. Hyponymy is asymmetrical so x is a hyponym of y while y is a hypernym of x: e.g., *canine* is a hypernym of *dog* and *dog* is a hyponym of *canine*.

When Extoken is used in the application of query-specific document clustering, user query is known to the system. If the query terms are provided as an input parameter, Extoken will consult WordNet for synonyms, hypernyms and hyponyms of the query as a whole and its individual components. Extoken notes all the synonyms, hypernyms and hyponyms appearing in the processed document and appends them to the result of the ranked keyphrase output as the source for document clustering processing.

In the context of query-specific Web document clustering, a query submitted by the user depicts the topic of interest to him/her. So by finding the synonyms of the query terms in the returned documents, we can potentially highlight the parts of the documents from the orientation of the user's interest. Moreover we can extract the relevant portion of the documents that may be deemed "irrelevant" because of the use of different vocabularies of the same topic. Hypernyms/hyponyms of query terms take the sense of the query terms to a more general or specific level of concept. For example, if the user queries "dog", the hyponyms of "dog" such as "pooch", "poodle" can be very interesting to the user. The vocabulary comprised of synonyms, hypernyms and hyponyms of the user query terms is of great interest to the user. For one thing, it will help users who want to know more about the document having the vocabulary, and for another by stressing the words that are of no interest to users, they can quickly discard it without further perusal. So if the query term is polysemous, by exposing the words that have the sense not intended by them, users can disregard the corresponding documents immediately. By getting the synonyms, hypernyms and hyponyms of the query terms, we enable Extoken to take the user's information need into account when extracting topical phrases. Therefore the document representation composed of keyphrases is tailored to the interest of the query submitter, which can potentially translate into better clustering results.

When the query is sent to Google, along with the ranked list we also receive

snippets associated with each document. A snippet is an excerpt from the returned result document that contains the query terms. It shows the context in which the query terms appear in the document [29]. The snippet actually provides text in the proximity of the query terms. Text around the appearance of the query terms often contains information relevant to the query. Therefore Extoken also takes snippet as an optional input parameter. Extoken processes the snippet as follows: First the snippet is broken up into sentences by using the same method that we discussed in section 2; second, for each sentence, Extoken extracts non-stopword terms (1-grams) and capitalized phrases and these phrases are defined as snippet vocabulary; lastly Extoken appends snippet vocabulary to the ranked keyphrase output.

Chapter 4

Categorizer

4.1 Overview

Categorizer is a Web clustering search engine prototype that we designed and implemented in this research. Categorizer was developed to demonstrate the utility of document clustering technique in the context of the World Wide Web. It is different from other traditional Information Retrieval systems, because it is designed to be functional on the Web which contains vast amount of heterogeneous and unstructured information. Categorizer can be classified as meta search engine in the family of Web search engines. It does not have its own indexes. Instead it processes the search results returned from the search engine Google [27]. The reasons why we choose Google as the back-end search engine can be summarized as follows:

- Google is currently one of the best search engines and also the most popular search engine according to Appendix D. Since our research interest is the performance of document clustering on the Web, when it comes to comparing the effectiveness between traditional ranked list results and clustering solutions, it is meaningful and fair for us to use the most effective and popular one as our back-end search engine;

- Google provides SOAP-based APIs as a programmatic way for users to search [28]. Since search engines always redesign their search result format interface, one of the biggest problems of the traditional way of parsing search results from Web pages is that parsing program has to be maintained on a regular basis in order to be in of sync with the search result Web page format change. So using these APIs makes our program more robust and easier to maintain.

Web search engines tend to return thousands of results in response to a query. Among the long list of returned Web documents, only some of them are actually relevant to users' information needs. Hence it is often a frustrating and painstaking undertaking for users to single out relevant items by wading through the ranked list. Possibly because of this, users seldom view results beyond the first page [38]. The aim of Categorizer is to improve the effectiveness and efficiency of the process by clustering Web documents. It organizes the returned document list into groups which represent conceptually related topics. Therefore it can potentially help users skip unrelated items and focus on relevant document groups.

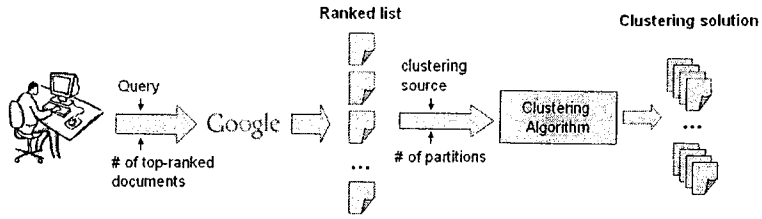

Figure 4.1: Flowchart of Categorizer

Figure 4.1 shows the flowchart of Categorizer. When Categorizer receives the query from the user, it performs the following procedure:

1. Forward the query to search engine Google and ask for the specified number of top-ranked pages;

2. Cluster the retrieved pages based on the specified page representation scheme (i.e., Key Phrases, Full Text or Snippet). For "Key Phrases", we use Extoken algorithm to extract the key phrases from the Web pages. HATF ranking scheme is used because in our experimental study, HATF and HLOGN both outperformed FREQ and the difference between them was not statistically significant (see Chapter 5 for more details about the experimental results).

In the following sections, we give an example of using Categorizer followed by the description of system design of Categorizer.

4.2 An Example of Using Categorizer

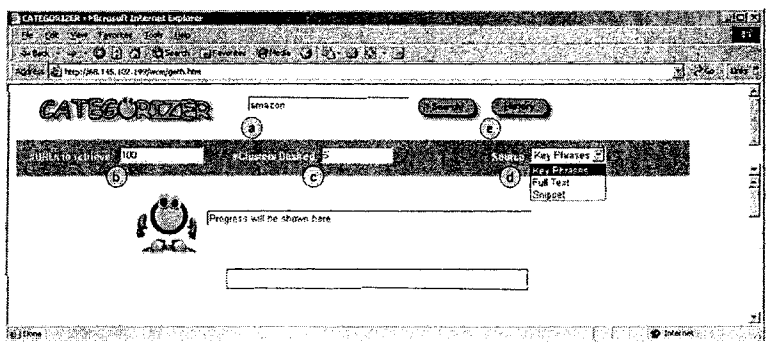

Figure 4.2: Categorizer Search Page

The search page of Categorizer is shown in Figure 4.2. It is quite simple and straightforward. Key elements of the page have been labelled in the figure:

a. Query input box. The user inputs his/her query here. Since the query will be dispatched to Google first, it follows the same rules as required by Google. Basically query terms can be any descriptive words. Terms are separated by blank spaces. Refining or narrowing search is as simple as adding more words to the search terms that have already been entered.

b. The number of URLs to be retrieved. This is the number of top-ranked URLs in the query result returned from Google that will be clustered. The default is 100. For faster response, users can decrease the number.

c. The number of clusters. We acknowledge that requiring this parameter could cause confusion to users since it is not obvious what number to put there. However, as we discussed in section 2.5, although we can easily change our partitional clustering algorithm to yield hierarchical results, we intentionally choose not to do so. Bringing in hierarchical result entails effort from users to sift through the tree-like structure to find a good cluster which has a dense set of "useful" Web pages. Our research goal here is to investigate the effectiveness of the plain simple flat clustering solution without the complication of visually more complicated form (i.e., dendrogram). Generally speaking, accepting the default which is set to 5 is a reasonable choice for most situations.

d. Source of clustering. Users can specify different sources of clustering: Keyphrase, Full Text and Snippet:

   - **Keyphrases**. Keyphrases are generated by the Extoken algorithm (Section 3.3) which extracts keyphrases from the original Web document.
   - **Full Text**. The full textual content of the original Web document (multimedia content such as images, graphics, audio and video are not taken into account).
   - **Snippet**. It refers to the summarization of a Web page in the search result returned from the search engine. In the search result of Google, it is defined as "an excerpt from the returned result page with your query

terms bolded. These excerpts let you see the context in which your search terms appear on the page, before you click on the result." [29].

e. Query history. Users can view what has been queried in history through here. When this button is clicked, a new page will pop up showing a list of history queries that have been submitted to the search engine recently. This history is obtained by querying the local cache database.

Suppose the user submits query "David Copperfield" to Categorizer. The clustering results are shown in Figure 4.3. The page provides an integrated interface where users can navigate the results without having to switch between different browser windows. The result page has a frame consisting of two panes. Initially the right pane shows the flat list of ranked URLs returned from Google in response to the query (in this case, it is the ranked list of URLs for query "David Copperfield"). The left pane of the frame displays a tree-like structure that users can navigate to examine the clustering results. The top-most node represents the query upon which the clustering solution is generated. The number in the parenthesis associated with the node indicates the total number of URLs that have been processed. Usually it is a number a little bit less than the number specified by the user in the query input page, because some URLs may not be available at the query time or the type of documents corresponding to the URLs are not processable to Categorizer (e.g., MS Excel document). The folder nodes nested right below the top-most one stand for all the output clusters. The number associated with the internal folder node tells users the number of Web pages that the cluster contains. The top five most frequently used non-stopword terms inside each cluster are used as the label for the cluster. They function as the visual hint telling users the topic of the respective cluster. In order to maximize the discrimination power, any term appearing in more than one cluster is discarded. We choose this way to summarize the content of the cluster, because it is succinct, direct and easy to implement. The similar mechanism was also implemented in the Scatter/Gather system to render the topical terms [9]. We would like to point out that clustering node labelling warrants further research in its own right. Users can expand the internal cluster nodes by clicking it. The leaf nodes nested under the cluster nodes represent Web documents contained in clusters. Each

leaf node is visually represented by the title of the Web page which is also a link to the actual Web page. When the user clicks the page link, the corresponding Web page will be loaded in the right pane. The page titles corresponding to the Web pages also work as a more detailed view of the content of the cluster.

Through browsing all the generated cluster nodes, the user can gain an overview of the result of the query "David Copperfield". They can easily perceive the different angles of the subject defined by the query. The five clusters approach the same subject through distinct orientations: magic (famous magician David Copperfield), novel (the classic novel authored by Charles Dickens), video (movies adopted from the novel) and show tickets (magician David Copperfield's show in Las Vegas). After users decide which sub-topic they are truly interested in, they can expand the cluster folder to view the related Web pages without having to spend time going through irrelevant pages digging for their "nuggets". If they want to know more about the show in Vegas, they may want to expand the fifth node. By looking through the page titles under this node, they can quickly find that the fourth document seems to have the information they are looking for. Whereas in the ranked list provided by Google (the right pane in Figure 4.3), because this page is placed as the 26th ranked URL, users won't be able to find it easily.

4.3 System Architecture and Design

A high-level system architecture of Categorizer can be illustrated by Figure 4.4. Categorizer comprises three major components: **Processing Unit**, **Web User Interface** and **Coordinating Unit**. Processing Unit is the back end processing center which downloads Web URLs to local disks, performs the major computing task of document clustering and formats the clustering results. Web User Interface is the online Web site that accepts users' queries, informs users the current status of processing and presents the results to them. The Coordinating Unit bridges the communication between Processing Unit and Web User Interface. Each user search session can be briefly described as follows:

1. The user inputs the query and specifies searching parameters on the main query input page;

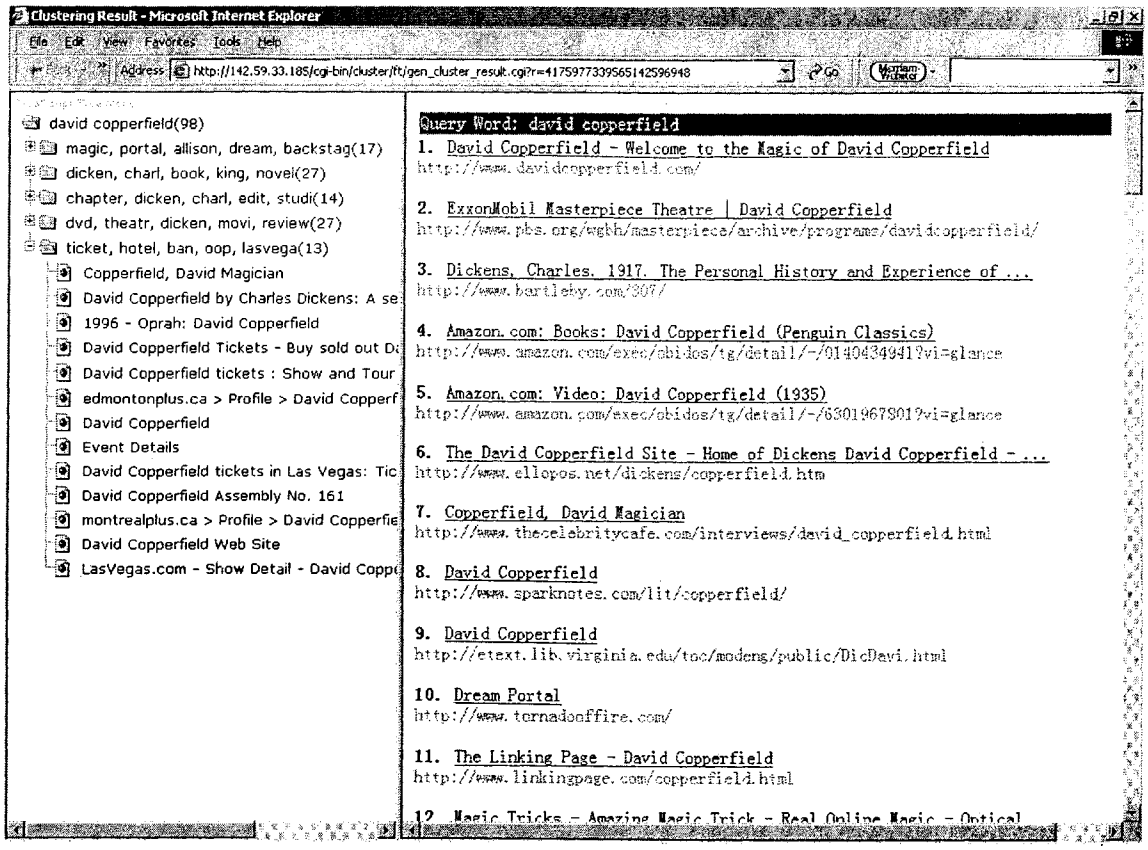

Figure 4.3: Web page that shows the clustering result.

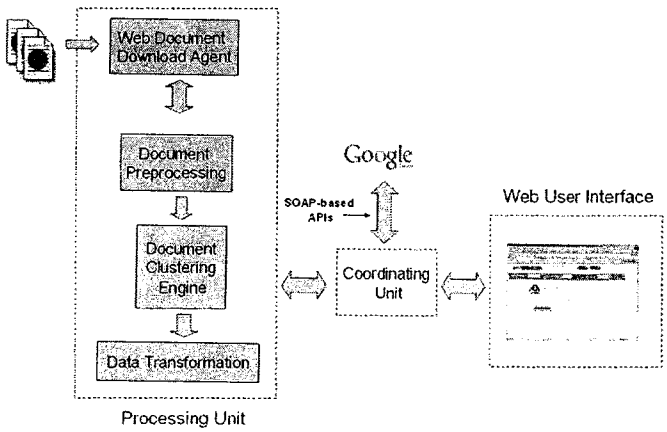

Figure 4.4: System Architecture of Categorizer

2. User query as well as searching parameters are sent to the Coordinating Unit;

3. Coordinating Unit forwards the query to backend search engine (i.e., Google). It receives the search result and passes it to Processing Unit;

4. Processing Unit initiates downloading of Web pages when applicable. After that it preprocesses the Web documents and starts document clustering. When document clustering is finished, it formats the clustering result;

5. When Processing Unit is working on the Web documents, Coordinating Unit periodically checks its progress. The status will be reported to the user. When Processing Unit is done, Coordinating Unit will send the clustering result to Web User Interface;

6. Web User Interface pushes the clustering result represented by a visual folder-tree back to the user;

7. The user browses the clustering result and finds what he/she is seeking.

We will further discuss each of the three major modules in the following subsections.

4.3.1 Processing Unit

Processing Unit embodies four modules: download agent, document preprocessing, document clustering engine and data transformation.

Download Agent

Users are given three choices when they submit queries (see Figure 4.2). These choices give users the freedom to specify from which source document clustering will be performed . Among the three different sources, snippet is the best in terms of time efficiency. No separate downloading is needed since snippets come with the search result returned from Google. Moreover, because lengths of snippets are substantially shorter than their corresponding Web pages, it is much faster to cluster them. However, generally speaking the efficiency gain for snippets is at the cost of effectiveness. The clustering result based on snippets is inferior to that of full document and extracted keyphrases according to our experiments. We will further discuss their comparative performance in the next Chapter.

When the whole documents are required (i.e., users opt for either "full text" or "keyphrases"), Web Document Download Agent will be invoked by Processing Unit. Web Document Download Agent can retrieve files using HTTP, HTTPS and FTP protocols. The working process of the Download Agent can be described as follows:

1. Parse the individual URLs from search query result fed by Coordinating Unit;
2. Check the types of documents represented by the URLs and reject those that are not supported by the system. Currently Categorizer can only accept HTML and plain text documents. So other types of documents indexed by Google such as PDF, PostScript, MS Word documents will be filtered out;
3. Consult with local cache database and download the Web documents when necessary (see Appendix B for a detailed description of the optimization for this step).

Once the Download Agent finishes downloading, document preprocessing will be started.

Document Preprocessing

Document Preprocessing module takes different steps to process downloaded Web documents based on the desired source of clustering (i.e., "Keyphrases", "Full Text" and "Snippet") specified by users.

Keyphrases The Extoken algorithm is applied to the document to extract the keyphrases. In the output of Extoken, each extracted keyphrase is weighted by a score (this shows how important Extoken thinks the phrase is in expressing the topic of the Web page). In order to let clustering algorithm "understand" the weighting score, some sort of transformation needs to be done. We take the IR view that a document is a "bag of words", so the occurrence frequency of the terms reflects the importance of the terms in the context of the document. As we discussed in section 2.4.1, vector space model is the most popular model used to represent documents in document clustering. VSM is also used in our Web clustering engine. In light of this, it is intuitive and interpretable for us to simply translate the weighting score directly into term frequency. The

ranking score $r(k)$ associated with each keyphrase k is a floating-point number that is between zero and one. The translated frequency $f(k)$ of keyphrase k can be obtained in the following way:

$$f(k) = round(r(k) \times 10) \quad 0 < r(k) \leq 1 \tag{4.1}$$

where round() is a function that rounds floating-point number to integer. For example, if $r(k)$ is 0.36, $f(k)$ will be 4. By doing this, the relative importance of different keyphrases is preserved and it is interesting to see how this can be used to enhance traditional clustering algorithm. In the next Chapter, we will take a close look at how our keyphrase-based artificial term frequency stacks up to the genuine term frequency in the scope of Web document clustering.

Full Text and Snippet Since clustering algorithm does not understand HTML, the first step is to strip off HTML tags from Web documents. After that stop words are taken out in order to reduce noise. Except for the common stop words used in traditional IR system (e.g., and, the, ...), owing to the characteristics of the Web, we supplemented the stop list with Web unique stop terms (e.g., email, contact, www, ...). As a standard term normalization process, stemming is also performed (the famous *Porter stemmer* [51] is used). Stop words removal and stemming were discussed in section 2.4.1.

For brevity's sake, in the following we will refer to the keyphrase-based artificial term frequency and full text as keyfreq and fulltxt respectively.

Document Clustering Engine

Partitional clustering algorithms were considered "inappropriate" for clustering search result due to their requirement of some a priori parameters [61]. The most common one is the number of desired clusters. However we believe a flat structure provided by partitional clustering can be as good as a hierarchy in terms of using clustering to represent the search result output. As we will reveal in the following sections, although there is no specific value that is always optimal when it comes to the number of clusters, it is not difficult to come up with some ranges which generally perform somewhere near the neighborhood of the optimum result. Though a hierarchy structure naturally avoids the requirement of forcing users to specify the

number of clusters to output, it actually shifts the burden to the user side partly. Users will have to go through the tree to find the nodes that contain relevant information. In practice, it is often more difficult to browse a multi-level taxonomy than a flat one-level clustering especially as labelling clusters properly is still a challenging research problem. Due to the multi-aspect nature of text documents, even when we have a human-edited well-defined taxonomy (e.g., Yahoo! [35]), documents can intrinsically belong to more than one categories. Instead of sifting through the ranked document list, users have to wade through the taxonomy to find relevant documents. Considering the problems of automatically generated node labels and clustering errors, it is not hard to imagine that browsing hierarchy could potentially become a frustrating retrieval process itself when the user gets disoriented navigating it. Lastly, hierarchical clustering is very effective for information reduction. However due to the response time constraint imposed by the interactive nature of query-specific clustering, typically only a small number of documents (no more than several hundred) need to be clustered. Therefore it is also unnecessary and inappropriate to use hierarchical clustering.

We want to caution that our point is not one-level flat structure is superior to multi-level dendrogram. However given the relatively easy concise presentation it poses, flat clustering solution is definitely a promising approach we should further study. Probably the best way to organize documents is to wisely choose when to use hierarchy and when not to. For example, when there are "too many" documents nested in one node we probably should consider splitting it into another level of clustering in order for the users to browse more easily.

VSM (Vector Space Model, covered in section 2.4.1) is used to represent the documents in this research. We chose a variant of vector-space K-Means clustering algorithm, bisecting K-Means, as our clustering algorithm that powers the clustering system. This algorithm was first presented in [60]. We used this algorithm because of the high-quality clustering solutions generated by it in traditional IR document collections reported in previous studies [69, 60, 70]. Previous experimental studies have shown that this algorithm outperformed popular hierarchical agglomerative algorithms (e.g., UPGMA, single-link, complete-link) and some other partitional algorithms both in outputting hierarchical presentation [70] and in producing flat

partitions [69, 60].

The K-Means algorithm can be deemed as an optimization process driven by a clustering criterion function [69]. Since the goal of K-Means algorithm is to maximize the similarities between documents and the centroid of the cluster where they are, the criterion function to optimize can be represented by the following (We use cosine measure for similarities between document vectors) [70]:

$$\sum_{r=1}^{k} \sum_{d_i \in S_r} cos(d_i, C_r) \tag{4.2}$$

where k is the number of clusters generated, S_r denotes the rth cluster, d_i is the ith document in S_r and C_r defines the centroid of S_r. See formula (2.5) for calculation of cosine measure between document vectors.

The K-Means clustering algorithm with bisections can be described as follows [60, 70]:

Bisecting K-Means Algorithm

1. Select a cluster to split.
2. Use K-Means algorithm to bisect the cluster into two sub-clusters.
3. Sub-clusters refinement. Randomly visit each document in the cluster. If assigning the document to the other sub-cluster can optimize (improve) criterion function (4.2), move the document to the other sub-cluster. Repeat step 3 until the criterion function can't be improved any more.
4. Iteratively repeat step 1, step 2 and step 3 until the desired number of clusters are produced.

For cluster selection scheme, we opted to select the cluster whose bisection would optimize the value of function 4.2 the most.

Because the clustering solution is sensitive to the seed documents picked, bisecting K-Means, like other partitional clustering algorithms, is nondeterministic. However previous experimental studies ([69, 60]) show K-Means via bisecting can consistently output relatively good results. Moreover bisecting K-Means offers low computational requirement. The time complexity is $O(n \log n)$.

Document ranking within a cluster node retains its original ordering in the ranked title list returned from the search engine (i.e., Google). So when d_1 precedes d_2 in the top-ranked list, d_1 will still be ahead of d_2 if they reside in the same cluster.

The high degree of independence between all the components in the system also allows other clustering algorithms be plugged in easily if needed.

Data Transformation

This module prepares the clustering solution data that is generated by document clustering engine for Web User Interface module. The document clustering engine outputs clustering result in a raw and generic format (thus the data can also be easily used in other applications, e.g., our experimental testing script). In order for the script on client side to easily render the clustering solution, the raw output result has to be reformatted to let the script read. Besides, the transformation process also calculates other required information based on the clustering output for the client (i.e., ranking of the cluster nodes as well as the ordering of documents located in the same node, text for labelling the cluster node).

4.3.2 Web User Interface

This part of the system is visible to the user. Web User Interface accepts a query from the user and send the result back to him/her. It is built with HTML pages and CGI scripts. The interface is similar to mainstream search engines. The designing philosophy behind the system is to minimize the time that it would take for users to learn how to use the system. As Resnick and Lergier pointed out [53]:

> People are very focused on their overall goals and do not want to focus on learning the search engine interface or even tailoring their queries to the design of a search engine they are familiar with.

So by making the interface of our system similar to popular Web search engines, we can not only save users valuable learning time but also improve the chance of users using it correctly.

The model of Web User Interface can be illustrated by Figure 4.5. Next, we examine the work flow of Web User Interface module as labelled in Figure 4.5:

1. The user submits query on search page. Figure 4.2 shows the search page that receives user queries and searching parameters.

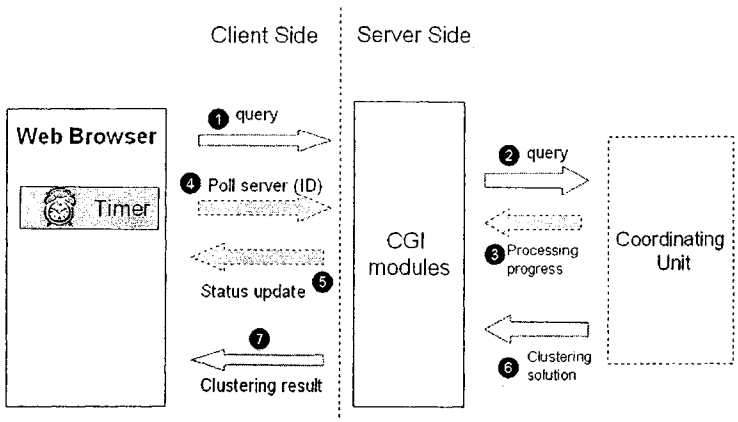

Figure 4.5: Web User Interface Model

2. Web User Interface on Web server side forwards the query to Coordinating Unit to start processing;

3. Coordinating Unit informs Web User Interface module of progress of the processing performed by Processing Unit;

4. A timer embedded in client side Web page triggers periodic status query submission through CGI. Because HTTP protocol is "stateless", in order to identify each query session, a session ID is created when user query is submitted. When client polls the server for status information, session ID is used to identify the client. Thus Categorizer is able to support multiple users simultaneously;

5. CGI modules on server side retrieve the status information based on session ID and push the data back to the client side. Script on client side Web page renders the progress status to keep users informed. Figure 4.6 is a snapshot of the Web page that shows users the status of processing progress.

6. Coordinating Unit notifies Web User Interface that the document clustering is done and sends the result to it;

7. CGI modules on server side push the clustering solution to client side and script in client side Web page displays the clustering result to the user.

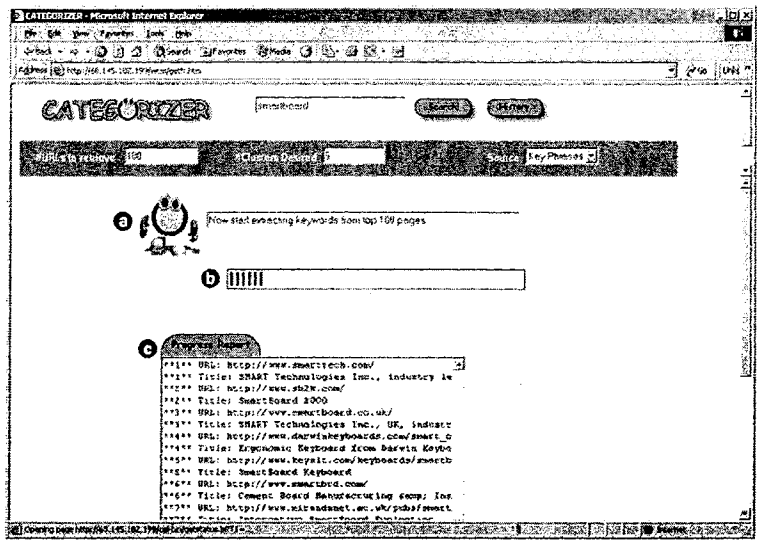

Figure 4.6: Web page that shows the progress of a query processing

4.3.3 Coordinating Unit

Coordinating Unit works as the intermediate between Web User Interface and Processing Unit. Its main functionalities can be summarized as follows:

- Accept user query and other search parameters (i.e., desired source to be clustered, the number of top ranked Web pages to be retrieved) and dispatch the query to Google;
- Forward search result from Google to Processing Unit and inform Web User Interface the current processing status of Processing Unit. Thus Web User Interface can visually indicate to the user which page is being downloaded and how much work has been done;
- When clustering is done, Coordinating Unit will be notified and it is its job to push the results to the Web User Interface.

Chapter 5

Performance Analysis

Because our research focused on the keyphrases extraction algorithm Extoken and its application in query-specific document clustering, naturally the experiments we did can be divided into two parts: evaluation of Extoken per se and experimental study of keyphrases application in query-specific document clustering on the Web.

5.1 Keyphrase Extraction Performance Evaluation

Previous work on automatic keyphrase extraction evaluation either relies on human judgement or calculates the overlap between author-assigned keyphrases which come with the documents and keyphrases extracted by the system that is being evaluated ([62, 66]). In this research we decided not to use these traditional methods because of the following reasons:

1. Human judgement brings subjectivity and individual bias to the evaluation. Additionally it is often hard for other researchers to validate and reproduce the evaluation results.

2. We are interested in Web document keyphrase extraction; however unlike academic journals, most Web documents do not have author-specified keywords. Even when keywords are present in the meta tags of HTML documents, they are often misleading and imprecise due to reasons mentioned in section 1.1.2.

In our research, a method called "how-Google-thinks-of-them" is used to judge if the extracted keyphrases are "good" or "bad". We believe that if the keyphrases are good, they should be able to convey the main idea of the page. Furthermore as the characteristics of the Web page, they can distinguish this document from others. Since Google now has de facto become the most popular search engine on the Web, we think if the keywords are good, sending the keywords to Google should be able to retrieve the original Web document among top results in the return list. In other words, the number of extracted keyphrases required to construct a query that can be used to retrieve the original Web document as well as the ranking of the document in query return can be used as objective measures to gauge the effectiveness of the extracted keyphrases. This evaluation model is very similar to the "content-based addresses" idea proposed by Martin and Holte [44]. In their research, Martin and Holte presented a method that could extract words and phrases from the target

Web page that were used to construct a query to Alta Vista retrieving the original page in the top-10 search results. They call the query "content-based address". The true URL of the Web page can be invalid due to the move of the page, however the "content-based addresses" can always retrieve the Web page as long as it is still on the World Wide Web and indexed by the search engine.

We believe the quality of extracted keyphrases should be evaluated in the context of the application of the keyphrase extraction algorithm. It is difficult and possibly of little meaning to just evaluate the quality of the keyphrases per se. Therefore the "how-Google-thinks-of-them" method actually evaluates the effectiveness of keyphrases extracted by the Extoken algorithm in the context of "content-based address". In section 5.2 we will evaluate how effective the keyphrases generated by Extoken are when they are used in the domain of query-specific Web document clustering.

5.1.1 Experimental Data

Because we are interested in keyphrase extraction for the Web documents, we collected our experimental document set from the Web. In order to evaluate the effectiveness of the keyphrases generated from Extoken algorithm in an unbiased manner, our collection of documents is comprised of Web documents coming from different sources and covering different subjects. Thus we can minimize the possibility that the experimental results are distorted by certain characteristics of a particular group of HTML documents. All Web documents in the collection were downloaded according to their respective URLs. The URLs were obtained from the following two sources:

***Yahoo!* Web Directory [35]** To cover different subjects, we randomly selected 2 URLs from each of the top categories in Yahoo Web directory. So in total we got $14 \times 2 = 28$ URLs. However a potential problem with these URLs is that most of the URLs in Yahoo directory are either Web portals (including home pages of organizations, companies and universities) or resource collection types. To compensate this bias, we also have the other category of URLs.

TREC Web documents datasets [23] Two steps were adopted to get the URLs:

1. We randomly picked out 24 topics (each topic consisting of several words briefly describing a subject) from TREC Web Test Collections and constructed query from the topic by removing stop words.
2. In order not to be biased by any specific search engine, we sent the 24 queries to three popular search engines: Google [27], AskJeeves [24] and MSN [21]. We then randomly chose 2 URLs from the top 50 results for each query. Because we would check the ranking position of the URLs in Google later, the only condition the URLs have to meet is that they should be indexed by Google (of course these URLs have to be nothing but HTML documents). So We obtained another $3 \times 24 \times 2 = 144$ URLs from this category.

We then downloaded all the Web documents corresponding to the URLs (frameset was handled by recursively downloading all pages embedded in the frames). Therefore in total the experimental dataset contains $28 + 144 = 172$ Web documents. We give each page an unique page ID number. The ID number ranges from 1 to 172.

5.1.2 Experimental Methodology

For each of the document in the dataset, we first ran it through Extoken using 3 different weighting schemes, i.e., HATF, HLOGN and FREQ (see section 3.3.3 for a detailed description of the schemes), to get top 10 ranked keyphrases. To eliminate the possibility that the results are trivial, we also used another scheme, RANDOM, as a comparison. RANDOM scheme simply randomly chooses non-stop words from the source document. After this step we acquired 4 keyphrases list corresponding to different schemes for each document.

For each keyphrases list $\{kp_1, kp_2, ..., kp_{10}\}$ (kp_n is the nth ranked keyphrase) of the document whose URL is U, the following procedure was performed:

- For $n = 1$ to 10 do
  - Formulate a query by combining $kp_1, ..., kp_n$.
  - Send the query to Google and retrieve the top-10 results.
  - Record the position of U.

Because normally Google presents 10 search results for a query in the first result page as default, if the query comprised of keyphrases can retrieve the original page

within top-10 results, we count it a successful retrieval. Hence the more successful retrievals the better the scheme is and the less number of keyphrases required to do a successful retrieval, the more effective it is.

5.1.3 Experimental Result

In the experiment, we explored the impact of different settings and options on results. Conceivably the more keyphrases we put into a query, the better chance it would have to retrieve the original document in top-ranked results from the search engine. Thus when n (n is the number of top-ranked keyphrases used in the query) is big enough, it does not matter how the components of the query are chosen any more. Due to this reason, we picked out $n = 3$ and $n = 5$ to illustrate the experimental results. We observed the same patterns for other values of n (in our research, we performed experiments for n from 1 to 10 with an increment of 1).

"RANDOM" scheme is the baseline since it represents the way to retrieve the source document that merely depends on the number of terms from the source document. We found when the value of n is increased, the performance of "RANDOM" approaches the other 3 schemes which indicates that the comparison of quality of keyphrases in terms of composing effective "content-based addresses" is less meaningful; this is because the number of terms is able to provide enough information for the search engine to retrieve the original document regardless of the quality of the terms with respect to topicality. We chose to present results for top-3 and top-5 keyphrases here as representatives of the experimental results because of the reason described above and the fact that results of other n values exhibit the same patterns. Experimental results are summarized in Table 5.1, 5.2 and 5.6.

Table 5.1 and Table 5.2 present the experimental results for top-3 and top-5 ranked keyphrases respectively. Each row has the results for a document whose page ID is shown by the first cell. "1" is used to indicates a successful retrieval in the table. The last row shows the total number of successful retrievals for different weighting schemes.

We used the Sign test (non-parametric test) to test the statistical significance of the results. The main reasons why this test is deemed appropriate for our experimental results are the following:

Page ID	HATF	HLOGN	FREQ	Random
1	1	1	1	1
2	0	0	0	1
3	0	0	0	0
4	1	0	0	0
5	1	1	1	0
...	...	...	...	...
172	1	0	0	0
Total	81	79	52	22

Table 5.1: Evaluation results of top-3 keyphrases from different weighting schemes

Page ID	HATF	HLOGN	FREQ	Random
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	0	1	0
5	1	0	1	0
...	...	...	...	...
172	1	1	0	0
Total	129	133	90.00	102.00

Table 5.2: Evaluation results of top-5 keyphrases from different weighting schemes

	HATF	HLOGN	FREQ
top-3	<0.0001	<0.0001	0.0134
top-5	0.0232	0.0108	0.8392

Table 5.3: Significance levels for different weighting schemes (successful retrieval)

	HATF	HLOGN	FREQ
top-3	<0.0001	0.001	<0.0001
top-5	<0.0001	<0.0001	0.983

Table 5.4: Significance levels for different weighting schemes (ranking score)

	HATF	HLOGN
top-3	0.016	0.0232
top-5	0.0018	0.0006

Table 5.5: Significance levels for HATF, HLOGN

Page ID	Top-3				Top-5			
	HATF	HLOGN	FREQ	Random	HATF	HLOGN	FREQ	Random
1	46	21	44	48	50	48	44	48
2	21	21	35	46	48	48	50	44
3	38	21	11	38	49	48	41	48
4	43	21	4	16	50	48	43	6
5	46	21	49	1	47	48	43	36
...	...	...	...	...	...	...	...	...
171	37	3	28	19	49	45	35	28
172	44	3	34	37	44	45	34	37
Average	34.34	30.21	30.44	23.54	41.33	47.13	33.52	33.73

Table 5.6: Ranking scores for different weighting schemes

- The samples of our experiments are correlated (different weighting schemes were applied to the same document sets).
- We can not safely assume that the population where the samples were drawn has a normal distribution.
- The samples of our experiments were randomly drawn from the Web.
- We only care about the direction of the difference, i.e., the subject can either have a successful retrieval or not have a successful retrieval.

Table 5.3 shows significance levels at which HATF, HLOGN and FREQ outperformed RANDOM. HLOGN and HATF are significantly better than RANDOM. FREQ did not show significance for top-5 results. In Table 5.5 we present significance levels at which HATF and HLOGN outperformed FREQ. We can see that keyphrase extraction algorithm based on HTML formatting information (i.e., HATF and HLOGN) significantly outperformed simple statistics based approach FREQ. The two-tailed Sign test gave significance levels of 0.9393 and 0.8191 for the comparison between HATF and HLOGN for top-3 and top-5 results respectively, which means neither HATF nor HLOGN is significantly better than the other. Due to the reason that we mentioned before (bigger n value makes retrieving source page in top-ranked results easier), in Table 5.2, we can see that the gaps between different schemes shrink compared to Table 5.1. For example, in Table 5.1, in terms of total number of successful retrievals, HLOGN is 259.1% better than RANDOM, but in Table 5.2, HLOGN is only about 30.4% better than RANDOM. HATF is 55.8%

better than FREQ for top-3 keyphrases, while for top-5 keyphrases, HATF is 43.4% better than FREQ for the total number of successful retrievals.

Table 5.6 shows the ranking scores of the test documents for different weighting schemes. The ranking score S_p of page P is calculated as follows:

$$S_p = 50 - R_p + 1$$

where R_p is the ranking place of page P in the ranked results returned from the search engine. Because we only use rankings within top 50, for page that shows up after top-50 results, its ranking score is 0. For example, if a page appears as the top-1 URL in the search result, its ranking score will be 50. If a page is the $50th$ ranked URL in the search result, its ranking score will be 1. So higher value of ranking score is better. Each row in Table 5.6 corresponds to the results for one document except that the last row gives the results averaged over all documents. We used Wilcoxon signed-ranks test (also a non-parametric test) to test the statistical significance between different schemes in terms of ranking score. The reason why we do not use the Sign test is that for ranking score, we do care about the magnitude of the difference between each pair of scores. From this table, we can see that in terms of average ranking scores, HATF and HLOGN again performed better than RANDOM. Wilcoxon test showed that HATF and HLOGN are both significantly better than RANDOM. The significance levels are given in Table 5.4.

From the results we presented, it follows that in our experimental environment, keyphrase extraction aided by HTML formatting elements analysis (i.e., HLOGN and HATF) significantly outperformed simple statistics based scheme (i.e., FREQ) in terms of using keyphrases to retrieve target Web documents from the search engine. The experimental results also demonstrated the usefulness and applicability of the technique of using HTML elements to help keyphrases extraction on top of lexical statistics. The effectiveness of extracted keyphrases is proven through the comparison between HLOGN, HATF, FREQ and RANDOM schemes. On the other hand, the performance difference between HATF and HLOGN is not significant and hence it suggests HLOGN and HATF can be used interchangeably.

5.2 Query-specific Document Clustering Experimental Results

Through experimental evaluation we aim to achieve the following goals:

1. Compare the effectiveness of clustering based on documents represented by keyphrases (keyfreq) extracted from Extoken with the effectiveness obtained by clustering the whole documents (fulltxt) and document summaries (snippet);

2. Compare the retrieval effectiveness between query-specific clustering presentation and traditional ranked-list output;

3. Evaluate the degree at which the number of top ranked documents retrieved affects the clustering quality;

4. Examine how the number of output partitions affects clustering quality.

In the rest of the section, we first describe our datasets and experimental methodology, followed by the description of the experimental results and the discussion of the results.

5.2.1 Document Collections and Relevancy Judgement

Because we are interested in the performance of the clustering method on the Web, we did not use standard IR document collections. In our experiment, we built our document sets based on the search results of different queries (i.e., each document set was derived from saving the search results of a particular query to Google). Because the initial retrievals were performed via Google, in the following sections when we speak of documents, we are referring to the Web documents corresponding to the URLs returned by the search engine in response to queries.

We in total have 11 document sets. The process of getting these document collections is summarized as follows:

1. **Define the source queries.** Since the platform that we carried out our experiments on is the Web, we used the following guidelines to define our queries:

- Queries should cover different subjects in order to minimize the potential bias that could be caused by homogeneous document sets;
- Queries should come from the real world instead of being crafted artificially in order to reflect the real information needs that exist;
- Non-trivial queries. We define trivial query as query reflecting information need that can be easily fulfilled by conventional search engines. To be more specific, we are interested in investigating the cases when clustering can be used to help and so we can find out if they do live up to the expectation or not. For example, if users are looking for the home page of CNN, chances are they can satisfy their information needs by simply typing "CNN" in Google.

In light of the above guidelines, we collected some of our source queries from the compiled popular keywords searched by users on major search engines [18]. Interestingly enough, the chronically organized keywords ranking often exposes the characteristics of the period when the keywords were most popular, e.g., recent trend in pop culture, breaking news that happened then etc. We picked out some queries from these popular words because we wanted our data to incorporate information needs that have a contemporary flavor apart from the fact that they are from the real world. Most of the other queries came from the log of our online Web clustering search engine prototype system (i.e., *Categorizer*). Since users of this system normally have some knowledge of what the system is and how it can help, the queries that they submitted usually are either problems they hope this system can help them with or information they can not easily get from conventional search engines. Query "salsa" actually was inspired by Zamir and Etzioni's paper [67]. We included this query simply because it is a typical example of when traditional search engines fall short;

2. **Construct the document collection based on source queries.** Each of the selected source queries was sent to Google and the top 200 pages along with their snippets were downloaded to create the respective document collections. Document collections were time stamped and named after the corresponding queries. We chose 200 pages because on one hand there are enough pages

Document Set Index	Query	Enhanced Query
1	*Amazon**	amazon rainforest
2	*Arnold schwarzenegger***	Arnold Schwarzenegger governor
3	*cayenne**	cayenne porsche
4	*David copperfield***	david copperfield magic
5	*eagle**	eagle bird
6	*hotel california**	hotel california reservation
7	*kobe bryant***	kobe bryant assault
8	*malibu***	malibu surf
9	*martha stewart***	martha stewart trial
10	*paradise hotel***	paradise hotel bomb kenya
11	*salsa**	salsa sauce

Table 5.7: Experimental queries information.

so that it is meaningful for clustering algorithm to organize and reduce information, and on the other the number of pages is still small enough that users won't be overwhelmed by the amount of information in the results. For each of the downloaded document, we ran it through the Extoken keyphrase extraction algorithm (the default HATF weighting scheme was used) to get the corresponding keyfreq (see definition in 4.3.1). Thus we generated 11 collections of 200 documents, 200 keyfreq lists and 200 snippets from the search results of the source queries.

Because the Web is changing constantly, by downloading all the documents in each collection we would be able to publish the data set on the Web so that other people can validate and replicate our experiments.

The queries used to generate our document collections are displayed in Table 5.7. The first column indicates the index number of the document collection with corresponding query being displayed in the second column in the same row. Queries annotated with "*" came from Categorizer log file and queries with "**" were collected from [18].

Table 5.8 summarizes some statistics of the eleven document sets. The total numbers of distinct terms with respect to the three representations for each of the document collection are shown in the table. "Clean-up" refers to the standard document preprocessing procedure (i.e., stemming and stop words removal). The last row displays the average number of terms over the 11 document collections.

Document Set	Before Clean-up			After Clean-up		
	fulltxt	keyfreq	snippet	fulltxt	keyfreq	snippet
1	16737	6902	1165	12046	5309	915
2	12406	4607	902	9217	3820	681
3	14399	6136	1148	10832	5094	908
4	9786	3905	970	7203	3201	747
5	9257	3924	1317	6771	3110	1051
6	9522	3506	1007	7512	2907	805
7	12522	6810	984	9387	5466	742
8	11392	4555	1170	8670	3722	942
9	12272	5344	1222	8753	4190	953
10	11485	4657	1136	8609	3746	912
11	12993	4615	1166	10005	3810	949
Average	12070	4996	1108	9000	4034	873

Table 5.8: Experimental document sets stat.

n	1	2	3	4	5	6	7	8	9	10	11	Ave.
top-50	8	7	9	16	2	9	15	5	5	6	3	7.73
top-60	9	8	10	16	2	10	20	7	8	7	3	9.09
top-70	10	9	13	18	2	11	23	7	9	8	3	10.27
top-80	12	10	15	20	2	11	25	7	9	10	7	11.64
top-90	13	10	18	22	2	12	27	9	10	11	9	13.00
top-100	15	13	20	24	2	13	29	9	12	11	9	14.27
top-110	16	14	23	26	2	13	30	9	13	13	9	15.27
top-120	16	14	25	27	2	13	34	10	13	13	11	16.18
top-130	16	16	28	28	2	13	34	10	14	13	12	16.91
top-140	16	16	28	28	2	15	36	11	15	13	13	17.55
top-150	16	16	29	30	2	16	39	12	17	13	13	18.45

Table 5.9: Number of relevant Web documents per query.

In the following section we will examine if the information reduction improves or degrades the quality of clustering.

To gauge the performance of the IR system, we need to know whether a document is relevant to the query. Traditionally human relevancy judgement is used to gain the relevancy information. This method would not only introduce human bias and errors but also require laborious work from user sides. Besides it is very hard to validate the results or replicate the experiments. Therefore, here we propose a query-expansion relevancy generation method that works as the premise for a comparative study of different clustering schemes. Using this method, relevancy judgement can be done without the interference of human.

The process of our query-expansion relevancy generation method can be described as follows:

1. Disambiguate the original source query in order to produce our "relevant documents". By analyzing the original queries, we found that they were either semantically unclear or polysemous in some sense (e.g., "Amazon" could mean the online bookstore or the famous river in South America). We believe that by adding more keywords to each of the source query, we can narrow down its meaning and hence improve the quality of the retrieval. We also believe that if the added keywords are carefully chosen, we can make the query very precise that the retrieval precision of the top-ranked results from Google can be regarded roughly as 100% (i.e., top ranked documents can all be deemed relevant) without distorting the experimental results. Hence we got a hypothesized relevant document set based on the assumption that the enhanced queries reflect users' true information needs much more precisely and in the meantime they work so well with Google that the top retrieved results can all be treated as relevant. For example, if the original query is "soprano", it could mean a female singer with the highest singing voice or a mafia crime boss depicted in a popular TV show. By adding one more word "singer" we can get a much cleaner document list which has a relatively coherent topic. Consequently we would regard all the top-ranked documents in the results as relevant.

2. Send the enhanced queries to Google to retrieve the "relevant" document list. Using the example given above, the enhanced query will be "soprano singer";

3. Find the relevant documents in the document collection derived from the source queries by comparing them to the "relevant document lists" obtained through the enhanced source queries. So suppose document set $S_q(n)$ contains the top n documents we get from source query q and document set $S_{q_e}(m)$ is acquired by taking top-m ranked documents in response to query q's enhanced version q_e, $S_q(n) \cap S_{q_e}(m)$ denotes the relevant Web documents incorporated in document set $S_q(n)$. In our experiments, n ranges from 50 to 150 with an increment of 10 and m is 300.

The third column of Table 5.7 displays the enhanced queries corresponding to the source queries. Table 5.9 shows the numbers of relevant documents for different numbers of top-ranked documents in the document collections. The numbers in the first row are the Document Set indexes. Each row corresponds to the top-n pages set and the first column specifies the value of n. The last column in the table displays the numbers of relevant documents averaged over all the document collections. With the enhanced queries served as the detailed information request description, human inspection of the results found that the results were pretty conservative which means those documents deemed "relevant" tended to be truly relevant, although sometimes some other documents that did not fall into the "relevant" category were also relevant. But because all experiments were performed based on these experiment settings, we consider the comparative studies fair.

5.2.2 Experimental Metrics

Without doubt the ultimate best way to evaluate if the search result output is "good" or "bad" is to let end user decide if the result is satisfactory. The only criteria that users have to measure the "goodness" of retrieval result would be if the result is what they are expecting and how easy it is for them to identify the relevant documents. However, it is not easy to implement such an ideal evaluation scheme in a controlled and unbiased manner in order to achieve consistent and objective evaluation results. This is due to the volatile nature of the Web and the variety

of users and their anticipations which can be influenced by users' demographics, beliefs, education backgrounds and personal preferences etc.

Traditional IR systems use precision and recall to evaluate the performance. If *Predicted* denotes the documents retrieved from the document collection N by the IR system in response to information request q and *Relevant* denotes all the relevant documents in document collection N, precision and recall can be defined as follows [54]:

$$Precision(Predicted, Relevant) = \frac{|Predicted \cap Relevant|}{|Predicted|} \tag{5.1}$$

$$Recall(Predicted, Relevant) = \frac{|Predicted \cap Relevant|}{|Relevant|} \tag{5.2}$$

Normally precision and recall are calculated based on the ranked list output by IR system. However, clustering solution produced by document clustering system consists of either a flat structure of partitions (partitional clustering algorithms) or a dendrogram (hierarchical clustering algorithms) which makes it impossible to apply precision/recall evaluation scheme directly.

Based on the concepts of traditional IR recall and precision, E effectiveness function was first proposed by Jardine and Van Rijsbergen [39] to evaluate clustering system. E effectiveness function is defined as:

$$E(P, R) = 1 - \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{5.3}$$

where P and R are precision and recall over the document set inside a cluster. Suppose for query q, we have the ranked list $R = \{p_1, p_2, ..., p_n\}$ (p_i is the ith ranked page in the ranked list) as search results from search engine S. Then we have clustering solution C_R which is obtained through clustering documents in the ranked list R: $C_R = \{c_1, c_2, ..., c_m\}$ where $c_1, ..., c_m$ are clusters (partitions) contained in C_R. Let $E(c_i)$ denote the E effectiveness function value of cluster c_i and $|c_i|$ denote the size of cluster c_i. If D_R denotes all the relevant pages in R, the precision and recall of c_i can be calculated in the following way:

$$P_{c_i} = \frac{|c_i \cap D_R|}{|c_i|} \tag{5.4}$$

$$R_{c_i} = \frac{|c_i \cap D_R|}{|D_R|} \tag{5.5}$$

Hence, the E effectiveness of Cluster c_i can be calculated as follows:

$$E_{c_i}(P_{c_i}, R_{c_i}) = 1 - \frac{(\beta^2 + 1)P_{c_i}R_{c_i}}{\beta^2 P_{c_i} + R_{c_i}}$$

β in the E effectiveness function is the parameter used to adjust the relative importance attached to precision and recall. Typically β can be 0.5 ,1 or 2. Thus each cluster in a clustering solution has an E effectiveness value as the measure of its quality. When both P and R are 1, E value is 0. The less the E value is, the better quality the cluster has. E effectiveness function and its variants have been widely used as the effectiveness measurements for document clustering systems ([61, 69, 60]).

The optimal cluster effectiveness is the best E value (the least E value) obtainable from a clustering solution. As in previous document clustering researches [39, 61], we use MK1 to denote optimal cluster effectiveness. The definition of MK1 of clustering solution C_R is given here:

$$MK1 = \min\{E_{c_1}, ..., E_{c_m}\} \tag{5.6}$$

The cluster that produces the least E value is called the optimal cluster and its E value represents the optimal cluster effectiveness. So in the context of query-specific clustering, the optimal cluster effectiveness stands for the best effectiveness available from a clustering presentation when users select one cluster to explore. It has been assumed in previous research [67, 9] that users were able to select the cluster with the highest effectiveness. However the validity of this assumption was questioned in [42]. We have no intention to argue here about whether or not users can find the most effective cluster, namely the optimal cluster. We agree with Tombros, Villa and Van Rijsbergen [61] on that optimal cluster measures eliminate any bias that may be introduced from sources external to the clustering itself. There are basically two problems here: the quality of the clustering solution and how users can find the best quality cluster. The former is the focus of our experimental evaluation while the latter warrants research in its own right. In the case of query-specific clustering, whether or not users can identify the optimal cluster depends on factors that are irrelevant to the quality of clustering such as cluster presentation, node labelling etc. By using optimal cluster effectiveness as our evaluation measure, we can be assured that the variation in effectiveness across different experimental parameters

in our experiments is caused by internal parameters such as different document representations, number of top-ranked documents etc [61].

In order to compare the optimal cluster effectiveness and ranked list effectiveness, we use another two measures: MK1-k and MK3. Suppose c_k $(1 \leq k \leq m)$ is the optimal cluster in C_R, the definition of MK1-k and MK3 of the ranked list R are given here:

$$MK1 - k = E(\{p_1, p_2, ..., p_{|c_k|}\}) \tag{5.7}$$

$$MK3 = \min(E(D_i)), 1 \leq i \leq n \quad and \quad D_i = \{p_1, p_2, ..., p_i\} \tag{5.8}$$

where $E(D_i)$ denotes the E effectiveness value of document collection D_i. MK1-k was introduced in [9] to measure the effectiveness of ranked list. MK1-k is based on MK1 in that the effectiveness of the ranked list is measured using a comparable threshold to MK1: Suppose the optimal cluster has j documents, we use j as the cut-off value for the ranked list returned from the search engine to calculate E effectiveness. In other words, we calculate the E value of the top-j ranked documents in ranked list as MK1-k value. Although MK1-k can tell us how the ranked list presentation performs under the same condition of optimal cluster, it is an unfair comparison since MK1-k is not an optimal value for the ranked list. As Tombros et al. [61] pointed out, the rank position for each query where the optimal effectiveness (least value of E effectiveness) can be derived is not being considered. To account for the optimality of the ranked list, MK3 was introduced in [39]. It represents the best effectiveness we can get from the ranked list like MK1 does for the clustering solution.

5.2.3 Experimental Methodology

For each of the document collection, because we wanted to investigate the variation of effectiveness across the progressive numbers of top-ranked documents retrieved (denoted by n), we calculated the E values ($\beta = 1, 2, 0.5$) for different n values ($50 \leq n \leq 150$ with an increment of 10). By adjusting the value of β, we can get the effectiveness measures for different experimental parameters in terms of the relationship between precision and recall. When β is 1, precision and recall are treated as equally important for the quality of clustering. While β is 0.5, the E effectiveness formula thinks precision is four times as important as recall. Likewise recall is treated four times as important as precision when β is 2. These three different

β values enable us to accommodate different user requirements in the real world, therefore they provide a complete picture for us to understand the effectiveness of the system.

Because we were also interested to know how the effectiveness varied across different numbers of partitions yielded by clustering, for each of the sub-collection corresponding to different n value, we performed the clustering for different practical numbers of partitions denoted by m ($2 \leq m \leq 20$). By saying "practical" we meant meaningful numbers that could be used in a real online clustering system. We set the upper bound to 20, because if there are too many clusters ($m > 20$) presented to users, on one hand the time they have to spend sifting through the partitions will defeat the purpose of using cluster to reduce information and moreover it poses a challenge for users to find the optimal cluster.

So we ran a total of $11 \times 11 \times 19 = 2299$ experiments for different experimental parameters and document collections.

To sum up, we used optimal cluster effectiveness MK1 to measure cluster effectiveness and MK1-k and MK3 measures for ranked list. These measures are all based on E effectiveness function which allows us to vary the relative importance between precision and recall.

Finally, to test the statistical significance of the results (E effectiveness values), we used the Wilcoxon signed-ranks test. Croft [8] suggested using this test based on the characteristics of the distribution of E values and the assumptions of the Wilcoxon signed-ranks test.

5.2.4 An Example

To better illustrate our evaluation scheme, in this section we offer an example to demonstrate how those experimental metrics are used in our experiments.

We use the following settings for this example:

$$n = 10, m \in \{2, 3\}, \beta \in \{0.5, 1\}$$

where n denotes the number of top-ranked pages retrieved and m denotes the number of partitions that will be generated by clustering.

According to the above settings, we have the following scenario:

# of top-ranked pages	E effectiveness ($\beta = 1$)	E effectiveness ($\beta = 0.5$)
1	1.000	1.000
2	0.667	0.583
3	0.714	0.688
4	0.500	0.500
5	0.333	0.375
6	0.400	0.464
7	0.455	0.531
8	0.333	0.444
9	0.385	0.500
10	0.429	0.545

Table 5.10: E effectiveness values for the ranked list when $\beta = 1$.

	E effectiveness	
Cluster	$\beta = 0.5$	$\beta = 1$
C_{2_1}	0.25	0.25
C_{2_2}	0.82	0.8

Table 5.11: E effectiveness values for clusters for $m = 2$.

	E effectiveness	
Cluster	$\beta = 0.5$	$\beta = 1$
C_{3_1}	0.063	0.143
C_{3_2}	0.75	0.75
C_{3_3}	1	1

Table 5.12: E effectiveness values for clusters for $m = 3$.

$n = 10$						
	$\beta = 0.5$			$\beta = 1$		
m	MK1	MK1-k	MK3	MK1	MK1-k	MK3
2	0.25	0.5	0.375	0.25	0.5	0.333
3	0.063	0.688	0.375	0.143	0.714	0.333

Table 5.13: Results in example

1. Retrieve query return. Suppose for query q (e.g., q could be "salsa") we retrieve the top 10 Web pages from Google (i.e., $n = 10$):

$$P_q =< P_1, P_2, ..., P_{10} >$$

The enhanced query q_e (e.g., q_e could be "salsa sauce") is used to get another 10 Web pages from Google in order to find relevant pages in P_q:

$$P_{q_e} =< P_2, P_{15}, P_4, P_{20}, P_5, P_8, P_{12}, P_{14}, P_{16}, P_{17} >$$

Therefore, within P_q, the relevant pages are:

$$P_r = P_q \cap P_{q_e} =< P_2, P_4, P_5, P_8 >$$

2. Calculate MK3. MK3 is the optimal effectiveness of the ranked list P_q. Table 5.10 shows different E-effectiveness values corresponding to different numbers of top-ranked pages. We describe how we compute the E effectiveness value for top-ranked 5 pages for $\beta = 0.5$ to illustrate the calculation of E effectiveness function:

$$P_{top-5} = \frac{3}{5} = 0.6$$

$$R_{top-5} = \frac{3}{4} = 0.75$$

$$E_{top-5} = 1 - \frac{(\beta^2 + 1)P_{top-5}R_{top-5}}{\beta^2 P_{top-5} + R_{top-5}} = 1 - \frac{(0.5^2 + 1) \times 0.6 \times 0.75}{0.5^2 \times 0.6 + 0.75} = 0.375$$

From Table 5.10, we can find that when the cut-off value is 5, we have the best E value 0.375 for $\beta = 0.5$. Table 5.10 shows that for $\beta = 1$ when the cut-off values are 5 and 8, we get the best E value 0.333. Therefore MK3 for P_q is 0.375 for $\beta = 0.5$ and 0.333 for $\beta = 1$.

3. Calculate MK1 for clustering solutions and MK1-k for the ranked list. Suppose when $m = 2$, our K-Means bisecting clustering algorithm outputs clustering C_2:

$$C_2 = \{\{P_1, P_2, P_4, P_5\}, \{P_3, P_6, P_7, P_8, P_9, P_{10}\}\}$$

Let's call the clusters in C_2 C_{2_1} and C_{2_2} respectively. We also describe how we compute the E effectiveness value for C_{2_1} for $\beta = 0.5$ to illustrate the

calculation of E effectiveness function for clusters:

$$P_{C_{2_1}} = \frac{3}{4} = 0.75$$

$$R_{C_{2_1}} = \frac{3}{4} = 0.75$$

$$E_{C_{2_1}} = 1 - \frac{(\beta^2+1)P_{C_{2_1}}R_{C_{2_1}}}{\beta^2 P_{C_{2_1}} + R_{C_{2_1}}} = 1 - \frac{(0.5^2+1)\times 0.75 \times 0.75}{0.5^2 \times 0.75 + 0.75} = 0.25$$

Tables 5.11 shows the E effectiveness values of the clustering for $\beta = 0.5$ and 1. Because C_{2_1} has the best E effectiveness values among all clusters, it is the optimal cluster in clustering solution C_2. Therefore MK1 is 0.25 when $\beta = 0.5$ and 1. Since the optimal cluster C_{2_1} has 4 pages, 4 is used as the cut-off value for calculating MK1-k of the ranked list. According to Table 5.10, MK1-k is 0.5 for $\beta = 0.5$ and 1.

Suppose when $m = 3$, our clustering algorithm outputs clustering C_3:

$$C_3 = \{\{P_2, P_4, P_5\}, \{P_1, P_3, P_7, P_8\}, \{P_6, P_9, P_{10}\}\}$$

Table 5.12 displays the E effectiveness values for the clusters in C_3.

We can find in Table 5.12 that C_{3_1} is the optimal cluster when $\beta = 0.5$, hence MK1 values of clustering solution C_3 are 0.063 for $\beta = 0.5$. Because C_{3_1} has 3 pages in the cluster, we can find in Table 5.10 that the corresponding MK1-k values for the ranked list P_q is 0.688 for $\beta = 0.5$.

For $\beta = 1$, in Table 5.12, we can find that C_{3_1} has the best E effectiveness value among the three clusters, the MK1 values of C_3 is the E effectiveness value of cluster C_{3_1} (i.e., 0.143). Consequently, from Table 5.10, we can get the MK1-k value of P_q: 0.5.

4. Compare the results. The results we get from the example can be compiled and displayed in Table 5.13. From Table 5.13, it follows that the clustering solutions are better than the ranked list for $\beta = 0.5$ and 1 in this example.

5.2.5 Empirical Results

Fulltxt vs. Keyfreq vs. Snippet

In this set of experiments, we were focused on how different representations of Web pages (i.e., fulltxt, keyfreq and snippet) could affect the performance of the

online query-specific Web documents clustering system. We also wanted to know if using keyphrases extracted from Web documents could improve the quality of the clustering.

The "MK1" columns of Table 5.15 show the optimal cluster effectiveness averaged over all the MK1 values obtained from the experiments using different experimental parameters (i.e., various n and m). Each row corresponds to the results of the document collection whose collection index number is specified by the first column. Remember less value is better. As a baseline and comparison, Table 5.16 presents the MK1 values averaged over different n values for trivial clustering: singleton cluster (i.e., each document is one cluster) and all-inclusive cluster (i.e., only one cluster that is comprised of the whole document set). It's quite obvious that these trivial clustering schemes are inferior to the clustering with a reasonable m value.

The first row of Table 5.18 shows the significance levels at which fulltxt scheme varies from keyfreq scheme regarding the average optimal cluster effectiveness (i.e., MK1). No statistical significance was achieved between them for different β values. We can find the same result for the average MK1 values between fulltxt and snippet which is shown in the second row of Table 5.18.

If we simply summarize the MK1 results of different Web page representations by averaging the MK1 values over all document collections in Table 5.15, the results can be potentially distorted. So instead we used the concept of "relative effectiveness" to compare the normalized effectiveness obtained from different page representation schemes. The procedure to calculate the "relative effectiveness" for each document set is described as follows:

1. Find the smallest E value $E_{min}(S, \beta)$ in each β value category for each document set S over the different representation schemes (i.e., fulltxt, keyfreq and snippet).

2. Divide the E value obtained from different representation schemes by $E_{min}(S, \beta)$ to get the relative effectiveness measure.

Note that when relative effectiveness is 1.0, it means it is the best. The higher it is the worse the corresponding representation performs. We then calculated the

"average relative effectiveness" for each of the representation schemes over the entire document collections. When a representation scheme has an average relative effectiveness score close to 1.0, its clustering effectiveness was the best in most of the document sets. Likewise a high average relative effectiveness score means that the representation scheme performed poorly compared to other schemes.

Table 5.14 presents the relative effectiveness for keyfreq, fulltxt and snippet representations of Web pages. The "Avg" row represents the average relative effectiveness over all the document collections. From this table we can see that for each value of β, overall keyfreq is always the best player comparing to fulltxt and snippet. Although fulltxt does not lead to the best overall solutions, its overall performance is close to keyfreq. Actually fulltxt is only 1.8%, 2.9% and 4.4% worse than keyfreq when β is 1, 2 and 0.5 respectively. In contrast, snippet is 21%, 27% and 21% worse than keyfreq for β being 1, 2 and 0.5 respectively. Interestingly we found that although the numbers of times when fulltxt and snippet are the best among the three schemes are more than the numbers of times when keyfreq is the best (fulltxt:keyfreq:snippet = 18:8:10), keyfreq has the best overall effectiveness in terms of E value. The reason is that keyfreq tended to yield consistently good results while the other two were inclined to fluctuation. Table 5.17 shows the standard deviations of the relative effectiveness for the different representation schemes over the 11 document sets. The worst performance of keyfreq is 35% worse than the best in that case. However, the worst case for fulltxt is 225% worse than the best and the worst case for snippet is 305% worse than the best.

To compare the maximum optimal cluster effectiveness obtainable from the three representation schemes, for each document collection we found the best MK1 values for different experimental settings in terms of different values of n and m. Figure 5.1, 5.2 and 5.3 show the frequency distribution of best MK1 values across all the document sets with the value of β set to 1, 2 and 0.5 respectively. Figure 5.1 and 5.3 (β is 1 and 0.5 respectively) show clear pattern that for most of the document sets keyfreq has the best MK1 values (the line of keyfreq stays on the bottom among the three representation schemes), while the line of fulltxt stays somewhere in the middle of the lines of keyfreq and snippet. Figure 5.2 (β is 0.5) gives us a different picture with regard to the best MK1 values from the three schemes. In Figure 5.2

	Relative Effectiveness								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
Query	keyfreq	fulltxt	snippet	keyfreq	fulltxt	snippet	keyfreq	fulltxt	snippet
1	1.11	1.05	1.00	1.11	1.00	1.00	1.08	1.10	1.00
2	1.02	1.00	1.39	1.07	1.00	1.48	1.00	1.05	1.42
3	1.10	1.00	1.67	1.13	1.00	1.58	1.00	1.06	1.69
4	1.06	1.00	1.45	1.08	1.00	1.38	1.00	1.00	1.57
5	1.00	1.00	1.28	1.02	1.00	1.64	1.01	1.00	1.20
6	1.03	1.03	1.00	1.00	1.03	1.02	1.10	1.10	1.00
7	1.23	1.00	1.25	1.26	1.00	1.28	1.15	1.00	1.18
8	1.00	1.88	2.56	1.00	2.25	3.05	1.00	1.78	2.41
9	1.27	1.08	1.00	1.27	1.00	1.04	1.35	1.16	1.00
10	1.09	1.00	1.09	1.15	1.00	1.20	1.11	1.00	1.07
11	1.22	1.32	1.00	1.22	1.38	1.00	1.19	1.26	1.00
Avg	**1.10**	1.12	1.34	**1.12**	1.15	1.42	**1.09**	1.14	1.32

Table 5.14: Relative effectiveness of clustering solutions for different representation schemes. The best average effectiveness in each β category averaged over the entire document collections appears in bold font.

fulltxt exhibits advantage over keyfreq and snippet. Figure 5.4 is a more direct comparison between the three Web page representation schemes with respect to the best MK1 values frequency distribution for different β values over all document sets. For best MK1 values with $\beta = 1$, 47% of the time they are produced by keyfreq which is 23.7% more than what fulltxt accounts for and 213% more than what snippet accounts for. For best MK1 values for $\beta = 0.5$, more than half of them are provided by keyfreq scheme which is 103.7% more than what fulltxt accounts for and 206% more than what snippet accounts for. When β is 2, however, fulltxt outperforms the other two schemes in terms of contribution to the best MK1 values. It accounts for 59% of the best MK1 values which is 78% more than what keyfreq accounts for and 637.5% more than what snippet accounts for.

Discussion

The most important observations we have made in the previous results description are:

1. Keyfreq outperformed both fulltxt and snippet in terms of best optimal cluster effectiveness with the exception being recall-oriented search ($\beta = 2$).

2. Although when it comes to average optimal cluster effectiveness, there is no

	Keyfreq								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
Query	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
1	**0.49**	0.83	0.70	**0.51**	0.83	0.52	**0.43**	0.82	0.73
2	**0.47**	0.96	0.75	**0.47**	0.94	0.56	**0.43**	0.97	0.82
3	**0.23**	0.88	0.65	**0.27**	0.88	0.43	**0.16**	0.87	0.74
4	**0.35**	0.74	0.56	0.42	0.76	**0.37**	**0.23**	0.70	0.62
5	**0.61**	0.97	0.88	**0.43**	0.95	0.76	**0.70**	0.98	0.91
6	0.66	0.82	**0.57**	0.64	0.79	**0.46**	**0.64**	0.83	**0.64**
7	**0.49**	0.86	0.53	0.54	0.86	**0.32**	**0.38**	0.85	0.62
8	**0.25**	0.78	0.67	**0.20**	0.75	0.59	**0.27**	0.79	0.72
9	**0.65**	0.93	0.76	0.62	0.90	**0.58**	**0.66**	0.94	0.83
10	**0.59**	0.89	0.77	**0.53**	0.86	0.58	**0.61**	0.89	0.81
11	**0.61**	0.79	0.69	**0.55**	0.77	0.60	0.63	0.81	**0.55**
	fulltxt								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
Query	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
1	**0.46**	0.83	0.70	**0.46**	0.82	0.52	**0.44**	0.83	0.73
2	**0.46**	0.95	0.75	**0.44**	0.93	0.56	**0.45**	0.96	0.82
3	**0.21**	0.88	0.65	**0.24**	0.87	0.43	**0.17**	0.87	0.74
4	**0.33**	0.73	0.56	0.39	0.74	**0.37**	**0.23**	0.70	0.62
5	**0.61**	0.98	0.88	**0.42**	0.95	0.76	**0.69**	0.98	0.91
6	0.66	0.84	**0.57**	0.66	0.82	**0.46**	**0.64**	0.83	**0.64**
7	**0.40**	0.81	0.53	0.43	0.81	**0.32**	**0.33**	0.81	0.62
8	**0.47**	0.79	0.67	**0.45**	0.76	0.59	**0.48**	0.81	0.72
9	**0.55**	0.92	0.76	**0.49**	0.89	0.58	**0.57**	0.94	0.83
10	**0.54**	0.88	0.77	**0.46**	0.86	0.58	**0.55**	0.88	0.81
11	**0.66**	0.79	0.69	0.62	0.76	**0.60**	0.67	0.79	**0.55**
	snippet								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
Query	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
1	**0.44**	0.84	0.70	**0.46**	0.84	0.52	**0.40**	0.84	0.73
2	**0.64**	0.96	0.75	0.65	0.95	**0.56**	**0.61**	0.97	0.82
3	**0.35**	0.88	0.65	**0.38**	0.88	0.43	**0.27**	0.87	0.74
4	**0.48**	0.77	0.56	0.54	0.79	**0.37**	**0.36**	0.73	0.62
5	**0.78**	0.98	0.88	**0.69**	0.96	0.76	**0.83**	0.99	0.91
6	0.64	0.84	**0.57**	0.65	0.82	**0.46**	**0.58**	0.83	0.64
7	**0.50**	0.87	0.53	0.55	0.87	**0.32**	**0.39**	0.86	0.62
8	**0.64**	0.81	0.67	0.61	0.78	**0.59**	**0.65**	0.82	0.72
9	**0.51**	0.94	0.76	**0.51**	0.93	0.58	**0.49**	0.95	0.83
10	**0.59**	0.89	0.77	**0.55**	0.88	0.58	**0.59**	0.89	0.81
11	**0.50**	0.80	0.69	**0.45**	0.78	0.60	**0.53**	0.81	0.55

Table 5.15: Evaluation results for clustering and ranked list averaged over results from different numbers of clusters and different numbers of top-ranked documents for all queries.

Query	Singleton			All-inclusive		
	$\beta = 1$	$\beta = 2$	$\beta = 0.5$	$\beta = 1$	$\beta = 2$	$\beta = 0.5$
1	0.85	0.90	0.70	0.76	0.56	0.83
2	0.84	0.89	0.68	0.78	0.59	0.85
3	0.89	0.93	0.77	0.67	0.45	0.77
4	0.91	0.94	0.81	0.61	0.39	0.72
5	0.33	0.44	0.17	0.96	0.90	0.97
6	0.85	0.90	0.69	0.77	0.57	0.84
7	0.93	0.95	0.84	0.55	0.33	0.66
8	0.78	0.85	0.60	0.83	0.67	0.89
9	0.82	0.88	0.66	0.79	0.61	0.86
10	0.82	0.88	0.65	0.80	0.62	0.87
11	0.73	0.81	0.55	0.85	0.70	0.90

Table 5.16: MK1 values for trivial clustering averaged over results from different numbers of top-ranked documents for all queries.

$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
keyfreq	fulltxt	snippet	keyfreq	fulltxt	snippet	keyfreq	fulltxt	snippet
0.09	0.26	0.44	0.09	0.36	0.56	0.10	0.22	0.42

Table 5.17: Standard deviations of relative effectiveness for different representation schemes.

	$\beta = 1$	$\beta = 2$	$\beta = 0.5$
keyfreq	0.734	0.278	1
snippet	0.102	0.105	0.278

Table 5.18: Significance levels for keyfreq and snippet

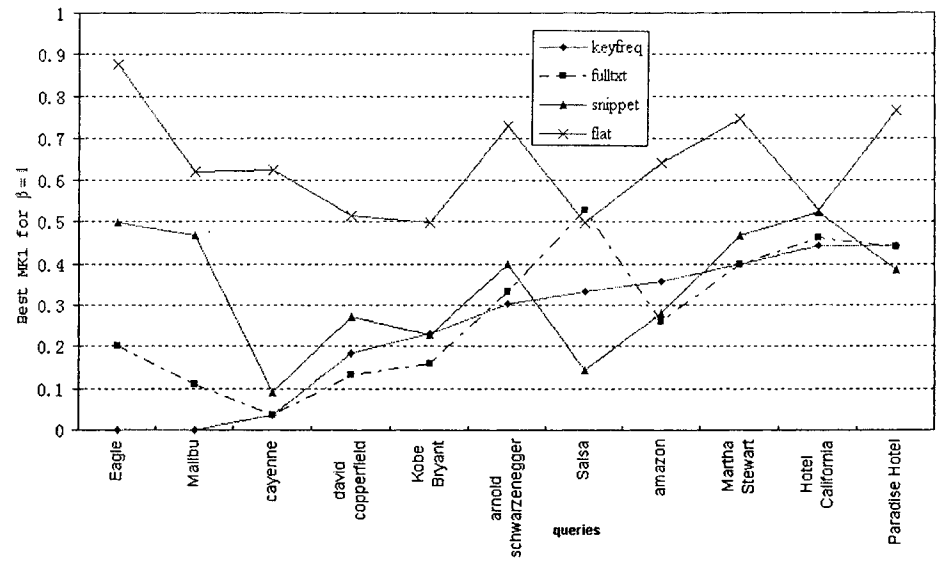

Figure 5.1: Frequency distribution of best MK1 ($\beta = 1$) value across different queries.

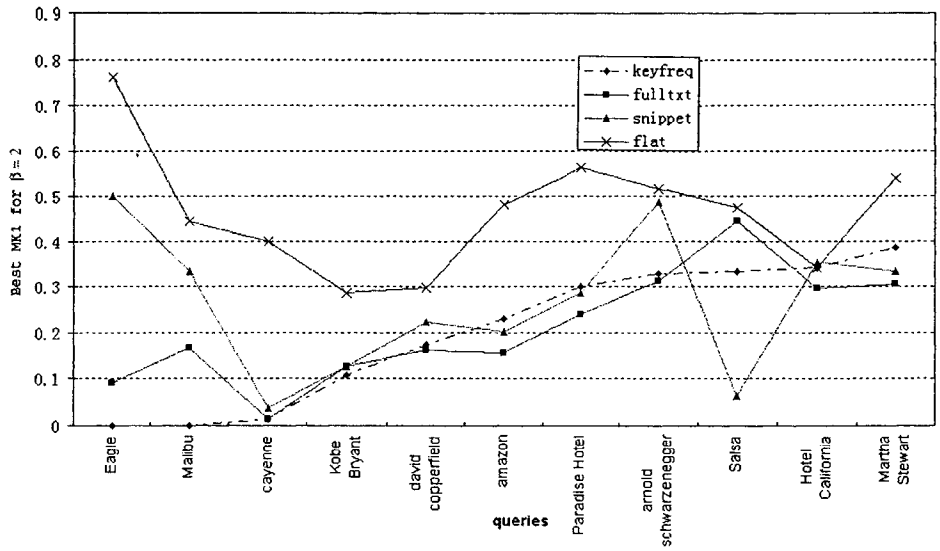

Figure 5.2: Frequency distribution of best MK1 ($\beta = 2$) value across different queries.

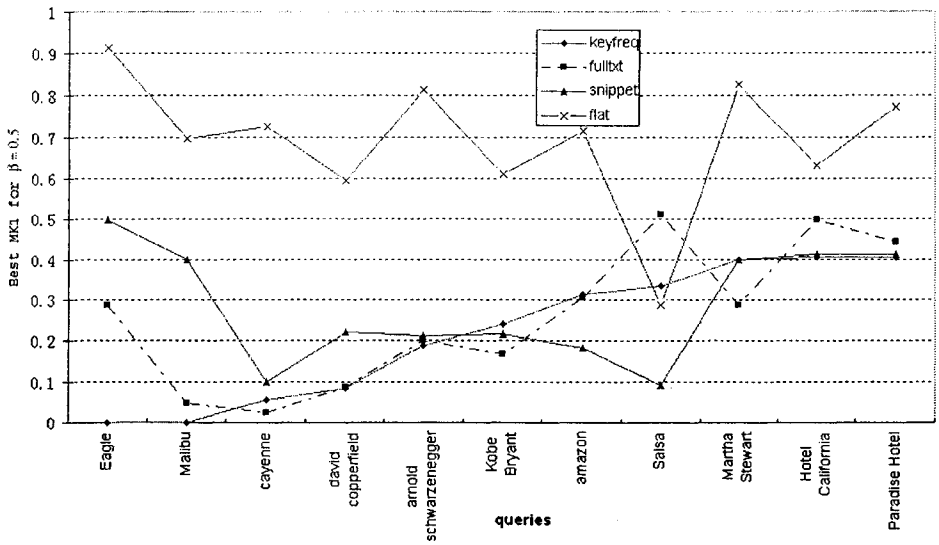

Figure 5.3: Frequency distribution of best MK1 ($\beta = 0.5$) value across different queries.

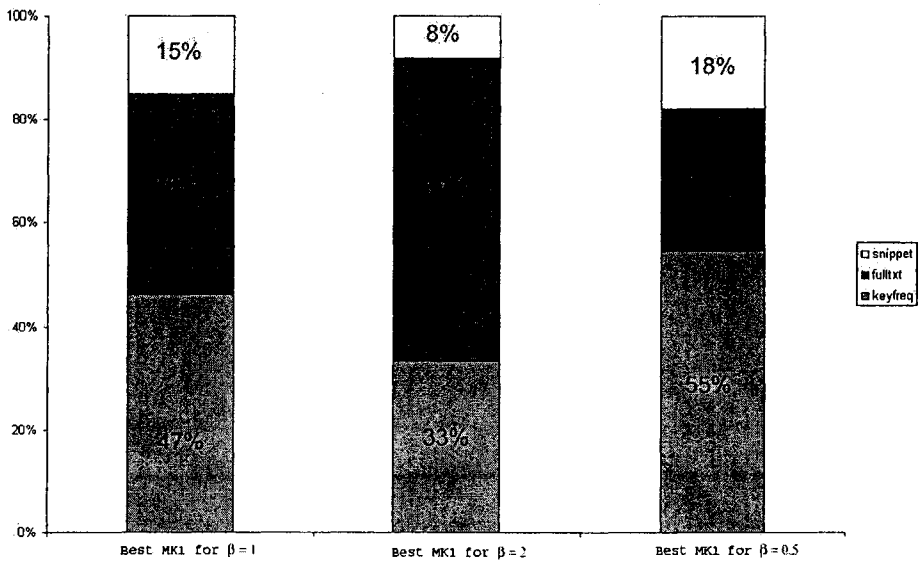

Figure 5.4: Frequency distribution of best MK1 values for different document representation schemes.

statistical significance shown between the three schemes in our experiments, keyfreq tends to give consistently good quality clustering solutions. Even though sometimes fulltxt and snippet yielded better clustering solutions, the quality of their clustering outputs fluctuates and is not very reliable.

On average keyphrase representation contains only 41.3% (44.8% after stop words removal and stemming) of the original Web pages' vocabulary. However clustering based on keyphrases yielded better quality clustering solution and experiments have shown it can output effective clustering consistently and reliably. We believe the reason is through keyphrases extraction "noisy" or "unimportant" data are filtered out while topic-related words and phrases are emphasized through their ranking scores. Consequently clustering algorithm can work on "clean" and more coherent documents which leads to better quality clustering output. Although snippet also reduces the information and inevitably drops some of the "noise" present in the original Web document, it also loses a lot of topically vital terms which makes it the worst player among the three Web page representation schemes regarding clustering quality.

Although clustering on keyphrases is computationally more efficient than full documents (just like snippet), keyphrases extraction itself requires more processing than snippet extraction. This seems to be an impediment that would prevent keyphrases from being employed in online Web clustering system. However we believe the architecture of current online Web search system can readily adopt keyphrases extraction module without causing major system overhead. For example, Google caches all the Web pages it indexed. When the Web page is retrieved and indexed, keyphrase extraction can be performed. The generated keyphrases can be stored with other information related to that page. When users initiate a clustering request to a query, the stored keyphrases corresponding to the retrieved Web pages can be clustered directly without having to download the pages and calculate keyphrases in real time. Besides the keyphrases can be either served as the page summary or presented with snippet in traditional ranked list as supplement information helping users catch the topic of the associated Web page and judge the relevancy. Even though the Web page based on which keyphrases are generated is not retrieved at query time, its freshness won't be compromised because of the

ongoing efforts major search engines are putting to ensure that their index database is up-to-date [48].

Clustering vs. Ranked List

We compared the effectiveness of clustering solutions represented by MK1 with the effectiveness of traditional ranked list presentation represented by MK1-k and MK3 in the experiments. Table 5.15 presents the average MK1 values as well as MK1-k and MK3 values for all document sets over different numbers of partitions ($2 \leq m \leq 20$) and different numbers of top-ranked documents ($50 \leq n \leq 150$). The best E values corresponding to different β categories in each row have been printed in bold font. We translated the data into relative effectiveness the way we did in last section. The corresponding relative effectiveness comparison is presented in Table 5.21.

As seen in Tables 5.15 and 5.21, when β is 1 and 0.5, overall the optimal cluster effectiveness of query-specific clustering based on all three different document representation schemes is much better than the optimal effectiveness of the ranked list. Ranked list performs relatively better when β is 2 (recall is emphasized) in terms of optimal effectiveness (MK3). Table 5.19 shows the significance levels at which query-specific clustering outperformed the ranked list in terms of MK1 and MK3. When $\beta = 1$ and 0.5, all three page representation schemes significantly outperformed the ranked list. For $\beta = 2$ (recall-oriented searches), none of the three schemes achieves statistical significance.

β	keyfreq	fulltxt	snippet
1	0.002	0.001	0.002
2	0.232	0.103	0.681
0.5	0.003	0.003	0.0005

Table 5.19: Significance levels for different β values for average MK1 over MK3.

Without any exception, query-specific clustering for all three Web page representation schemes outperforms the ranked list substantially with respect to MK1-k (The one-tailed P value given by Wilcoxon signed-ranks test is less than 0.0001 for all three schemes and different β values).

β	keyfreq	fulltxt	snippet
1	0.0005	0.0010	0.0005
2	0.0010	0.0005	0.0010
0.5	0.0010	0.0015	0.0005

Table 5.20: Significance levels for different β values for best MK1 over MK3.

Figures 5.1, 5.2 and 5.3 show how the best MK3 (indicated by "flat" in the figures) values stack up against the best MK1 values for clustering based on different representation schemes for β valued 1, 2 and 0.5 respectively. From these figures we can see in terms of maximum effectiveness, the ranked list method is greatly inferior to query-specific clustering on a consistent basis. In Table 5.20, we present the significance levels at which query-specific clustering using different page representations outperformed the ranked list in terms of best MK1 and MK3. All three schemes (i.e., keyfreq, fulltxt and snippet) achieve significance for maximum effectiveness.

Discussion

From the above experimental results, we have seen that query-specific clustering considerably outperformed the ranked list consistently both in terms of performance on average and optimal effectiveness under different experimental conditions. The statistical significance is achieved for most experimental conditions except for the average optimal cluster effectiveness of the recall-oriented searches ($\beta = 2$). We used two different measures (i.e., MK1-k and MK3) to evaluate the effectiveness of the ranked list while using optimal cluster effectiveness (MK1) to measure the effectiveness of clustering solution. We also compared the performance of the ranked list and query-specific clustering for different values of β which can reflect various requirements that can exist in the real world in terms of relationship between precision and recall.

Based on the experimental results, we can draw the conclusion that query-specific clustering can be used as an alternative search result presentation instead of the traditional ranked list. It has the potential to present retrieval results in a more effective way in terms of precision and recall in contrast to the ranked list. However we want

	keyfreq								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
Query	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
1	**1.00**	1.69	1.43	**1.00**	1.63	1.02	**1.00**	1.91	1.70
2	**1.00**	2.04	1.60	**1.00**	2.00	1.19	**1.00**	2.26	1.91
3	**1.00**	3.83	2.83	**1.00**	3.26	1.59	**1.00**	5.44	4.63
4	**1.00**	2.11	1.60	1.14	2.05	**1.00**	**1.00**	3.04	2.70
5	**1.00**	1.59	1.44	**1.00**	2.21	1.77	**1.00**	1.40	1.30
6	1.16	1.44	**1.00**	1.39	1.72	**1.00**	**1.00**	1.30	**1.00**
7	**1.00**	1.76	1.08	1.69	2.69	**1.00**	**1.00**	2.24	1.63
8	**1.00**	3.12	2.68	**1.00**	3.75	2.95	**1.00**	2.93	2.67
9	**1.00**	1.43	1.17	1.07	1.55	**1.00**	**1.00**	1.42	1.26
10	**1.00**	1.51	1.31	**1.00**	1.62	1.09	**1.00**	1.46	1.33
11	**1.00**	1.30	1.13	**1.00**	1.40	1.09	1.15	1.47	**1.00**
Avg.	**1.01**	1.98	1.57	**1.12**	2.17	1.34	**1.01**	2.26	1.92

	fulltxt								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
Query	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
1	**1.00**	1.80	1.52	**1.00**	1.78	1.13	**1.00**	1.89	1.66
2	**1.00**	2.07	1.63	**1.00**	2.11	1.27	**1.00**	2.13	1.82
3	**1.00**	4.19	3.10	**1.00**	3.63	1.79	**1.00**	5.12	4.35
4	**1.00**	2.21	1.70	1.05	2.00	**1.00**	**1.00**	3.04	2.70
5	**1.00**	1.61	1.44	**1.00**	2.26	1.81	**1.00**	1.42	1.32
6	1.16	1.47	**1.00**	1.43	1.78	**1.00**	**1.00**	1.30	**1.00**
7	**1.00**	2.03	1.33	1.34	2.53	**1.00**	**1.00**	2.45	1.88
8	**1.00**	1.68	1.43	**1.00**	1.69	1.31	**1.00**	1.69	1.50
9	**1.00**	1.67	1.38	**1.00**	1.82	1.18	**1.00**	1.65	1.46
10	**1.00**	1.63	1.43	**1.00**	1.87	1.26	**1.00**	1.60	1.47
11	**1.00**	1.20	1.05	1.03	1.27	**1.00**	1.22	1.44	**1.00**
Avg.	**1.01**	1.96	1.54	**1.08**	2.07	1.25	**1.02**	2.16	1.83

	snippet								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
Query	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
1	**1.00**	1.91	1.59	**1.00**	1.83	1.13	**1.00**	2.10	1.83
2	**1.00**	1.50	1.17	1.16	1.70	**1.00**	**1.00**	1.59	1.34
3	**1.00**	2.51	1.86	**1.00**	2.32	1.13	**1.00**	3.22	2.74
4	**1.00**	1.60	1.17	1.46	2.14	**1.00**	**1.00**	2.03	1.72
5	**1.00**	1.26	1.13	**1.00**	1.39	1.10	**1.00**	1.19	1.10
6	1.12	1.47	**1.00**	1.41	1.78	**1.00**	**1.00**	1.43	1.10
7	**1.00**	1.74	1.06	1.72	2.72	**1.00**	**1.00**	2.21	1.59
8	**1.00**	1.27	1.05	1.03	1.32	**1.00**	**1.00**	1.26	1.11
9	**1.00**	1.84	1.49	**1.00**	1.82	1.14	**1.00**	1.94	1.69
10	**1.00**	1.51	1.31	**1.00**	1.60	1.05	**1.00**	1.51	1.37
11	**1.00**	1.60	1.38	**1.00**	1.73	1.33	**1.00**	1.53	1.04
Avg.	**1.01**	1.66	1.29	1.16	1.85	**1.08**	**1.00**	1.82	1.51

Table 5.21: Relative effectiveness for clustering and ranked list averaged over results from different numbers of clusters and different numbers of top-ranked documents for all datasets.

to caution that we are not making a statement here that query-specific clustering is superior to the ranked list in general and we should replace the ranked list with clustering solution indiscriminately. Due to the way our queries were selected, we actually focused on the issue of whether clustering is better or more effective in situations when traditional ranked lists fall short. Later we will discuss exactly when clustering solution can be used as a better retrieval results presentation compared to ranked list.

Clustering Effectiveness Variation Across Different Values of n

In the experiments we also wanted to study how the change of the number of top-ranked documents influenced the clustering effectiveness. The results of the experiments are reported in Table 5.22 and Table 5.23. For purpose of completeness, we also listed the corresponding MK1-k and MK3 values for different values of n . These data actually corroborate what we have claimed in last section that effectiveness of query-specific clustering is higher than the ranked list in our experiments.

Table 5.23 presents the optimal cluster effectiveness measured by MK1 averaged over different document sets and different numbers of partitions for clustering corresponding to progressively increased numbers of top-ranked documents. The highest effectiveness in each column is in bold font. We can make a couple of observations from analyzing the results. First, the effectiveness of clustering generally declines as the number of top-ranked documents increases for different values of β and for different representation schemes. Second, the pattern we observed for clustering effectiveness also applies to the optimal ranked list effectiveness (MK3). This probably implies that it is not very beneficial for users to browse deep into the ranked list. Third, there is no obvious monotonic variation for increasing values of n for MK1-k and the best values often appear with high values of n.

The best MK1 values for different numbers of top ranked documents averaged over all document sets are shown in Table 5.22. Again the results for MK1 follow the same pattern as we observed in Table 5.23: degradation of best effectiveness for increasing value of n.

Discussion

We present the possible causes of potential unfairness to large number of n regarding optimal cluster effectiveness along with our explanation of why they do not hold in our experiments:

1. **Decreased recall**. Because the E values are calculated based on the number of relevant pages in retrieved documents, when the cluster size does not increase in proportion to the number of top-ranked documents clustered, it is unfair for large values of n. As Tombros, Villa and Rijsbergen [61] stated: For large number of n the number of relevant pages in retrieved documents increases, but the average cluster size does not always increase in proportion for increasing value of n which translates into lower recall. In our experiments, a partitional document clustering algorithm is used which means the average cluster size grows with the total number of top-ranked documents retrieved.

2. **Decreased precision**. For a partitional algorithm, when the number of output partitions is determined beforehand, the size of each cluster will grow with increasing value of n. Coupled with the fact that in the ranked list the density of relevant pages tends to drop along with the increasing value of n (after all the job of ranking algorithm in search engines is to put more promising pages on top), one can expect a depreciation of precision when n grows. However in our experiments we explored different values of m (number of partitions) for clustering, hence when n increases, m can still be leveraged by the clustering algorithm to make a balance. In fact the best E values shown in Table 5.22 still support the same pattern: Effectiveness decreases as n grows.

Based on the above analysis, we can conclude that our experimental settings eliminate the possibility that the decreasing clustering effectiveness with increasing values of n could be caused by unfairness to large number of n.

According to the Nearest Neighbor test discussed by Tombros et al. [61], the clustering tendency of document collections tend to decrease for increasing value of n, which means clustering hypothesis holds better when n is small. Our experimental results support this claim. Our clustering system tends to yield better quality clustering solution when n is relatively small.

	keyfreq								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
n	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
top50	**0.31**	**0.72**	**0.65**	0.25	0.63	**0.47**	0.29	**0.76**	**0.69**
top60	**0.31**	0.73	**0.65**	0.25	0.63	0.49	0.29	0.77	**0.69**
top70	**0.31**	**0.72**	**0.65**	**0.24**	**0.62**	0.49	**0.28**	0.77	0.70
top80	0.37	**0.72**	0.68	0.28	**0.62**	0.52	0.35	0.77	0.72
top90	0.40	0.74	0.69	0.30	0.65	0.53	0.38	0.78	0.73
top100	0.39	0.73	0.69	0.30	0.64	0.53	0.37	0.77	0.73
top110	0.38	0.74	0.70	0.29	0.65	0.53	0.38	0.78	0.74
top120	0.40	0.75	0.70	0.29	0.66	0.54	0.39	0.78	0.74
top130	0.41	0.73	0.70	0.31	0.64	0.54	0.41	0.77	0.74
top140	0.44	0.75	0.71	0.33	0.65	0.55	0.42	0.78	0.75
top150	0.46	0.75	0.71	0.36	0.65	0.56	0.43	0.78	0.75

	fulltxt								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
n	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
top50	**0.31**	0.73	**0.65**	**0.25**	0.63	**0.47**	**0.31**	0.77	**0.69**
top60	**0.31**	**0.71**	**0.65**	**0.25**	**0.60**	0.49	**0.31**	**0.76**	**0.69**
top70	0.36	0.73	**0.65**	0.28	0.62	0.49	0.33	0.78	0.70
top80	0.37	0.73	0.68	0.28	0.62	0.52	0.36	0.78	0.72
top90	0.37	0.74	0.69	0.28	0.64	0.53	0.34	0.78	0.73
top100	0.36	0.74	0.69	0.28	0.65	0.53	0.36	0.78	0.73
top110	0.41	0.73	0.70	0.32	0.64	0.53	0.38	0.77	0.74
top120	0.42	0.75	0.70	0.32	0.65	0.54	0.40	0.79	0.74
top130	0.40	0.74	0.70	0.32	0.64	0.54	0.41	0.78	0.74
top140	0.41	0.75	0.71	0.34	0.65	0.55	0.39	0.80	0.75
top150	0.45	0.75	0.71	0.36	0.64	0.56	0.45	0.80	0.75

	snippet								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
n	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
top50	**0.34**	**0.72**	**0.65**	**0.30**	**0.62**	**0.47**	**0.30**	**0.76**	**0.69**
top60	0.40	0.73	**0.65**	0.31	0.64	0.49	0.36	**0.76**	**0.69**
top70	0.43	0.73	**0.65**	0.36	0.63	0.49	0.41	0.77	0.70
top80	0.40	0.75	0.68	0.34	0.65	0.52	0.37	0.78	0.72
top90	0.44	0.74	0.69	0.36	0.66	0.53	0.41	0.78	0.73
top100	0.43	0.75	0.69	0.34	0.67	0.53	0.40	0.79	0.73
top110	0.47	0.75	0.70	0.37	0.67	0.53	0.44	0.78	0.74
top120	0.47	0.75	0.70	0.38	0.67	0.54	0.46	0.78	0.74
top130	0.48	0.74	0.70	0.39	0.65	0.54	0.46	0.78	0.74
top140	0.50	0.74	0.71	0.39	0.65	0.55	0.50	0.78	0.75
top150	0.50	0.75	0.71	0.40	0.66	0.56	0.49	0.78	0.75

Table 5.22: Evaluation results (best effectiveness for all datasets across different numbers of top ranked documents). Highest effectiveness (the lowest value of E) for each column appears in bold.

	keyfreq								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
n	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
top50	**0.43**	0.89	**0.65**	**0.42**	0.88	**0.47**	**0.40**	0.89	**0.69**
top60	0.44	0.86	**0.65**	0.43	0.85	0.49	0.42	0.86	**0.69**
top70	0.46	**0.85**	**0.65**	0.44	0.84	0.49	0.43	**0.85**	0.70
top80	0.48	0.86	0.68	0.46	0.85	0.52	0.45	0.86	0.72
top90	0.49	0.87	0.69	0.48	0.86	0.53	0.47	0.86	0.73
top100	0.50	0.86	0.69	0.48	0.85	0.53	0.47	0.86	0.73
top110	0.50	0.86	0.70	0.48	0.84	0.53	0.48	0.86	0.74
top120	0.51	0.86	0.70	0.48	0.85	0.54	0.48	0.86	0.74
top130	0.51	**0.85**	0.70	0.48	**0.83**	0.54	0.50	**0.85**	0.74
top140	0.54	**0.85**	0.71	0.51	**0.83**	0.55	0.52	**0.85**	0.75
top150	0.55	**0.85**	0.71	0.53	**0.83**	0.56	0.53	**0.85**	0.75

	fulltxt								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
n	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
top50	**0.42**	0.87	**0.65**	**0.40**	0.86	**0.47**	**0.41**	0.87	**0.69**
top60	0.43	**0.85**	**0.65**	0.41	0.84	0.49	0.42	0.85	**0.69**
top70	0.47	**0.85**	**0.65**	0.45	0.84	0.49	0.45	**0.85**	0.70
top80	0.48	**0.85**	0.68	0.45	0.84	0.52	0.47	0.86	0.72
top90	0.49	0.86	0.69	0.46	0.85	0.53	0.46	**0.85**	0.73
top100	0.49	0.86	0.69	0.46	0.84	0.53	0.47	0.86	0.73
top110	0.51	0.86	0.70	0.48	0.84	0.53	0.49	0.86	0.74
top120	0.51	**0.85**	0.70	0.48	**0.83**	0.54	0.50	**0.85**	0.74
top130	0.52	**0.85**	0.70	0.48	**0.83**	0.54	0.51	**0.85**	0.74
top140	0.51	**0.85**	0.71	0.48	0.84	0.55	0.50	**0.85**	0.75
top150	0.53	**0.85**	0.71	0.50	**0.83**	0.56	0.53	**0.85**	0.75

	snippet								
	$\beta = 1$			$\beta = 2$			$\beta = 0.5$		
n	MK1	MK1-k	MK3	MK1	MK1-k	MK3	MK1	MK1-k	MK3
top50	**0.50**	0.89	**0.65**	**0.51**	0.89	**0.47**	**0.44**	0.89	**0.69**
top60	0.52	0.87	**0.65**	0.52	0.86	0.49	0.48	0.87	**0.69**
top70	0.56	0.86	**0.65**	0.56	0.85	0.49	0.51	0.86	0.70
top80	0.53	0.88	0.68	0.54	0.87	0.52	0.49	0.87	0.72
top90	0.55	0.89	0.69	0.55	0.88	0.53	0.51	0.88	0.73
top100	0.55	0.88	0.69	0.55	0.87	0.53	0.50	0.87	0.73
top110	0.58	0.87	0.70	0.57	0.86	0.53	0.54	0.87	0.74
top120	0.57	0.87	0.70	0.57	0.86	0.54	0.54	0.87	0.74
top130	0.57	0.86	0.70	0.56	**0.84**	0.54	0.54	**0.85**	0.74
top140	0.59	0.86	0.71	0.58	0.85	0.55	0.56	0.86	0.75
top150	0.59	**0.85**	0.71	0.57	**0.84**	0.56	0.57	0.86	0.75

Table 5.23: Evaluation results (effectiveness averaged over 2-20 partitions for all datasets across different numbers of top ranked documents). Highest effectiveness (the lowest value of E) for each column appears in bold.

Partition Numbers

In this set of experiments we aimed to investigate the degree at which numbers of partitions output by clustering process affect the optimal cluster effectiveness across progressively larger numbers of top-ranked documents.

Table 5.24 shows the numbers of partitions that generate the best optimal cluster effectiveness expressed in MK1 for β being 1 for document collection #10. We chose to only show the result for document collection #10 because results of all other document sets share the same trend and pattern as this one. Before the experiments we speculated that the number of partitions producing the best MK1 would probably increase as the number of top-ranked documents rises in order to keep the average size of the clusters down. However surprisingly that is not what we observed from the experiment results. As shown in Table 5.24, the numbers of clusters that yield the best effectiveness are not very sensitive to the number of retrieved top-ranked documents. The numbers of clusters giving the best MK1 do not necessarily grow as the numbers of top-ranked documents increase. Interestingly, the other observation we made is that, for a given number of retrieved top-ranked documents, the numbers of partitions yielding the best effectiveness tend to stay within a continuous range. These observations hold for different settings of β and different page representation schemes.

Table 5.25 shows the average numbers of m values that produce the least MK1 ($\beta = 1$) values for different values of n over entire document collection sets. The last row in this table tells us the average numbers of m values having the best MK1 values over all n values for all document collections.

Because in previous section we have experimentally proved keyfreq outperformed the other two representation schemes (i.e., fulltxt and snippet) in terms of optimal cluster effectiveness, next we focus on analyzing the variation of effectiveness across different numbers of clusters for keyfreq. In order to compare the effectiveness achieved through different m values, we used the relative optimal cluster effectiveness in order not to let different characteristics in document sets distort our results. The procedure to calculate relative effectiveness has been introduced in section 5.2.5.

Table 5.26 shows the results of relative optimal cluster effectiveness for different document sets, for $\beta = 1$, 2 and 0.5, for keyfreq scheme and for top-50 ranked pages

Document Set 10	MK1(β=1)		
	Keyfreq	Fulltxt	Snippet
top50	4,5,6,7,8,9	4,5,6,7,8,9,10	8,9,10,11,12
top60	4,5,6,7,8,9,10	2,3,4,5,6,7,8	3,4,5,6,7
top70	4,5,6,7,8	6,7,8,9,10,11,12	4,5,6
top80	13,14,15,16,17,18,19,20	2,3,4,5,6,7,8	2,3
top90	2,3,4,5	2,3,4,5,6,7	2,3
top100	2,3,4	2,3,4,5,6	4,5,6,7,8
top110	2,3,4,5	2,3,4,5,6,7,8,12,13,14,15,16,17,18,19	2,3
top120	2,3,4,5	2,3,4,5,6,7,8	2,3
top130	2,3,4,5	10,11	4,5,6,7,8
top140	2,3,4,5	9,10,11,12	2,3,4
top150	2,3,4,5	9,10,11,12,13,14	18,19,20

Table 5.24: Numbers of partitions that generate the best optimal cluster effectiveness across progressively larger numbers of top-ranked Web pages from search engine for document collection 10

n	($\beta = 1$)			$\beta = 2$			$\beta = 0.5$		
	Keyfreq	Fulltxt	Snippet	Keyfreq	Fulltxt	Snippet	Keyfreq	Fulltxt	Snippet
top50	6.5	7.4	5.1	5.7	6.1	4.5	6.2	7.6	6.0
top60	7.3	6.5	4.9	4.7	6.0	3.6	5.9	6.5	6.2
top70	4.8	5.7	4.2	4.4	4.5	3.2	6.1	5.9	6.9
top80	5.9	6.5	5.0	5.2	4.3	3.7	6.8	5.8	6.7
top90	6.4	6.1	5.5	4.5	4.5	2.5	6.6	6.5	5.7
top100	6.1	6.2	5.4	3.9	4.4	2.7	6.6	6.2	5.9
top110	4.5	6.6	5.7	3.5	4.5	3.1	5.5	4.6	5.5
top120	4.7	6.4	6.1	3.7	4.5	3.5	6.3	6.0	6.8
top130	5.0	5.4	5.1	3.8	4.4	4.1	6.4	5.4	6.3
top140	6.1	6.6	5.4	3.5	5.2	3.2	7.3	7.7	6.5
top150	5.1	6.5	5.2	3.4	4.6	3.8	6.6	7.0	6.4
Avg.	5.7	6.4	5.2	4.2	4.8	3.4	6.4	6.3	6.3

Table 5.25: Numbers of m values yielding the best optimal cluster effectiveness averaged over all document sets for progressively larger numbers of top-ranked documents and average numbers of m values over all values of n.

returned from the search engine across progressively increasing numbers of partitions output from clustering process. Again the smaller the relative effectiveness value is (1 is the minimum), the better the clustering solution is. Except for the first and last column, each column of Table 5.26 corresponds to a document set whose index number is specified in the second row. The first column labelled "m" indicates the value of m (number of partitions). The last column labelled "Avg." corresponds to relative effectiveness averaged over all the document sets. Rows in Table 5.26 have been sorted in order of ascending values of average relative effectiveness (last column). Document set 8 is not displayed in Table 5.26 because some MK1 values (for top-50 pages) are zero. This posed a problem for us to calculate the relative effectiveness and would distort the final average relative effectiveness. Table E.1 and E.2 are provided in Appendix E for similar data for top-100 and top-150 ranked Web pages.

From Table 5.26 we can make some observations. When we used top-50 ranked pages to cluster, small values of m (specifically when m falls in the range of 5 and 10) tend to work the best for $\beta = 1$ and $\beta = 2$. When β is 0.5 however, we got the comparatively better results for m from 10 to 16. These observations are true even when n is considerably larger (Table E.1 and E.2 in Appendix E). This tells us those ranges of numbers are relatively insensitive for values of n for this particular clustering algorithm we used. The other observation is that extreme values of m (e.g., 2 and 20) tend to produce poor results.

Primarily for the sake of completeness, in Appendix E we provide figures showing distribution of values of m for yielding best MK1 values for different β values for all three page representation schemes across different values of n for all document sets (Figure E.1, E.2, E.3, E.4, E.5 and E.6).

$\beta = 1$											
m	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**9**	**10**	**11**	**Avg.**
9	1.21	1.00	1.00	2.11	1.00	1.16	1.49	1.00	1.00	1.00	1.20
7	1.00	1.00	1.00	1.00	2.78	1.16	1.49	1.43	1.00	1.32	1.32
5	1.00	1.50	1.00	1.00	2.78	1.00	1.49	1.50	1.00	1.32	1.36
6	1.00	1.50	1.00	1.00	2.78	1.16	1.49	1.50	1.00	1.32	1.38
10	1.21	1.00	2.36	2.11	1.00	1.16	1.90	1.00	1.21	1.00	1.40
11	1.21	1.00	2.36	2.11	1.00	1.16	1.90	1.00	1.21	1.00	1.40
4	1.00	1.50	1.00	1.00	3.67	1.00	1.09	1.50	1.00	1.32	1.41
12	1.43	1.00	2.36	2.11	1.00	1.16	1.90	1.00	1.21	1.00	1.42
8	1.21	1.00	1.00	2.11	2.78	1.16	1.49	1.43	1.00	1.00	1.42
13	1.43	1.04	2.36	2.11	1.00	1.16	1.90	1.00	1.21	1.00	1.42
14	1.43	1.04	2.36	2.11	1.00	1.25	1.90	1.00	1.25	1.00	1.44
3	1.20	1.50	1.00	1.00	3.67	1.00	1.00	1.50	1.25	1.32	1.44
15	1.43	1.04	2.36	2.11	1.00	1.25	1.90	1.00	1.25	1.20	1.46
16	1.43	1.04	2.36	2.11	1.00	1.25	2.54	1.00	1.25	1.20	1.52
17	1.43	1.04	7.26	2.11	1.00	1.25	2.54	1.25	1.25	1.20	2.03
18	1.43	1.04	7.26	2.11	1.00	1.47	2.54	1.25	1.25	1.20	2.06
19	1.43	1.04	7.26	2.11	1.00	1.47	2.54	1.25	1.25	1.20	2.06
20	1.43	1.04	7.26	3.24	1.00	1.47	2.54	1.25	1.25	1.08	2.16
2	1.20	1.76	9.70	1.42	4.29	1.16	1.00	1.77	1.25	1.52	2.51
$\beta = 2$											
m	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**9**	**10**	**11**	**Avg.**
9	1.69	1.00	1.00	2.73	1.00	1.58	3.09	1.00	1.00	1.00	1.51
3	1.00	1.21	1.00	1.44	5.76	1.13	1.00	1.07	1.09	1.31	1.60
5	1.15	1.21	1.00	1.44	3.66	1.13	3.09	1.07	1.00	1.31	1.61
7	1.15	1.00	1.00	1.44	3.66	1.58	3.09	1.21	1.00	1.31	1.64
6	1.15	1.21	1.00	1.44	3.66	1.58	3.09	1.07	1.00	1.31	1.65
4	1.15	1.21	1.00	1.44	5.76	1.13	2.01	1.07	1.00	1.31	1.71
8	1.69	1.00	1.00	2.73	3.66	1.58	3.09	1.21	1.00	1.00	1.80
10	1.69	1.00	8.45	2.73	1.00	1.58	3.92	1.00	1.69	1.00	2.41
11	1.69	1.00	8.45	2.73	1.00	1.58	3.92	1.00	1.69	1.00	2.41
12	2.03	1.00	8.45	2.73	1.00	1.58	3.92	1.00	1.69	1.00	2.44
13	2.03	1.25	8.45	2.73	1.00	1.58	3.92	1.00	1.69	1.00	2.46
14	2.03	1.25	8.45	2.73	1.00	1.84	3.92	1.00	2.06	1.00	2.53
15	2.03	1.25	8.45	2.73	1.00	1.84	3.92	1.00	2.06	1.50	2.58
16	2.03	1.25	8.45	2.73	1.00	1.84	4.85	1.00	2.06	1.50	2.67
2	1.00	1.17	13.50	1.00	7.76	1.00	1.00	1.23	1.09	1.61	3.04
17	2.03	1.25	22.73	2.73	1.00	1.84	4.85	1.41	2.06	1.50	4.14
18	2.03	1.25	22.73	2.73	1.00	2.16	4.85	1.41	2.06	1.50	4.17
19	2.03	1.25	22.73	2.73	1.00	2.16	4.85	1.41	2.06	1.50	4.17
20	2.03	1.25	22.73	3.80	1.00	2.16	4.85	1.41	2.06	1.45	4.27
$\beta = 0.5$											
m	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**9**	**10**	**11**	**Avg.**
14	1.08	1.00	1.00	2.47	1.00	1.00	1.13	1.00	1.00	1.16	1.18
15	1.08	1.00	1.00	2.47	1.00	1.00	1.13	1.00	1.00	1.22	1.19
13	1.08	1.00	1.00	2.47	1.00	1.06	1.13	1.00	1.26	1.16	1.21
16	1.08	1.00	1.00	2.47	1.00	1.00	1.75	1.00	1.00	1.22	1.25
10	1.00	1.68	1.00	2.47	1.00	1.06	1.13	1.00	1.26	1.16	1.28
11	1.00	1.68	1.00	2.47	1.00	1.06	1.13	1.00	1.26	1.16	1.28
12	1.08	1.68	1.00	2.47	1.00	1.06	1.13	1.00	1.26	1.16	1.28
9	1.00	1.68	1.52	2.47	1.00	1.06	1.00	1.00	1.21	1.16	1.31
7	1.03	1.68	1.52	1.00	2.33	1.06	1.00	1.59	1.21	1.48	1.39
8	1.00	1.68	1.52	2.47	2.33	1.06	1.00	1.59	1.21	1.16	1.50
17	1.08	1.00	3.70	2.47	1.00	1.00	1.75	1.03	1.00	1.22	1.53
18	1.08	1.00	3.70	2.47	1.00	1.08	1.75	1.03	1.00	1.22	1.53
19	1.08	1.00	3.70	2.47	1.00	1.08	1.75	1.03	1.00	1.22	1.53
5	1.03	3.01	1.52	1.00	2.33	1.05	1.00	1.73	1.21	1.48	1.54
6	1.03	3.01	1.52	1.00	2.33	1.06	1.00	1.73	1.21	1.48	1.54
4	1.03	3.01	1.52	1.00	2.85	1.05	1.00	1.73	1.21	1.48	1.59
3	1.37	3.01	1.52	1.00	2.85	1.05	1.36	1.73	1.50	1.48	1.69
20	1.08	1.00	3.70	4.52	1.00	1.08	1.75	1.03	1.00	1.00	1.72
2	1.37	3.62	11.63	3.94	3.17	1.31	1.36	1.98	1.50	1.64	3.15

Table 5.26: Relative optimal cluster effectiveness for top-50 documents for keyfreq across increasing values of m.

Discussion

Apparently no specific value of m is superior to others and gives the best result at all time. However after analyzing our experimental results, we can draw some general guidelines when it comes to choose a "good" value for m. Based on the results, we found that on the full spectrum of all possible values of m studied in our experiments (from 2 to 20), the clustering solutions produced by keyfreq-based clustering tend to perform better in terms of optimal cluster effectiveness when the values of m fall in the middle of the spectrum than values from the two ends of the spectrum. Moreover for keyfreq-based clustering, MK1 for $\beta = 1$ and $\beta = 2$ seem to be more alike to each other than to $\beta = 0.5$ in terms of values of m to output better results. More specifically our experiments show MK1 for $\beta = 1$ and 2 favor m valued between 5 and 10, and on the other hand, when m ranged from 10 to 16, MK1 for $\beta = 0.5$ gains its best. Considering in practice it is hard to know apriori whether precision or recall is more important to the user or maybe they are equally important, it seems 10 is a good candidate to be used as the default for keyfreq-based clustering. It generally performed well for varying numbers of top-ranked pages and did not give inconsistent performance under different experimental conditions in our experiments.

Performance Comparison Between Vivisimo and Categorizer

Vivisimo [34] was founded by Carnegie Mellon University research computer scientists in 2000. Over the past a few years it has become the leading clustering search engine on the Web. Thus it would be interesting for us to compare the clustering quality between Vivisimo and Categorizer.

Vivisimo does not collect search results from Google. Its Web sources include "MSN", "Lycos", "Looksmart", "Wisenut", "Open Directory" and "Overture". The other difference between Vivisimo and Categorizer is that Vivisimo's clustering result is hierarchical, which means that the top-level clusters may contain sub-clusters. In our experiments, we treated Vivisimo's results as flat partitions (we ignored the lower-level clusters nested under the top-level clusters and only calculated the E effectiveness for the top-level clusters) in order to make a meaningful comparison between Vivisimo and Categorizer since Categorizer generates flat partitional

n	1	2	3	4	5	6	7	8	9	10	11	Ave.
top-50	12	24	5	13	2	3	11	6	10	1	6	8.46
top-100	16	42	12	24	6	15	19	9	13	1	6	14.82
top-150	20	58	16	52	9	22	46	10	15	2	8	23.46

Table 5.27: Number of relevant Web pages per query for Vivisimo.

clustering. Each URL only belongs to one cluster in Categorizer, while Vivisimo produces overlapping clusters which means one document can appear in different clusters. Despite all the differences, it is still interesting to see how Vivisimo performs under the same experimental settings as Categorizer, i.e., same queries, same relevant URL identification method and same experimental metrics.

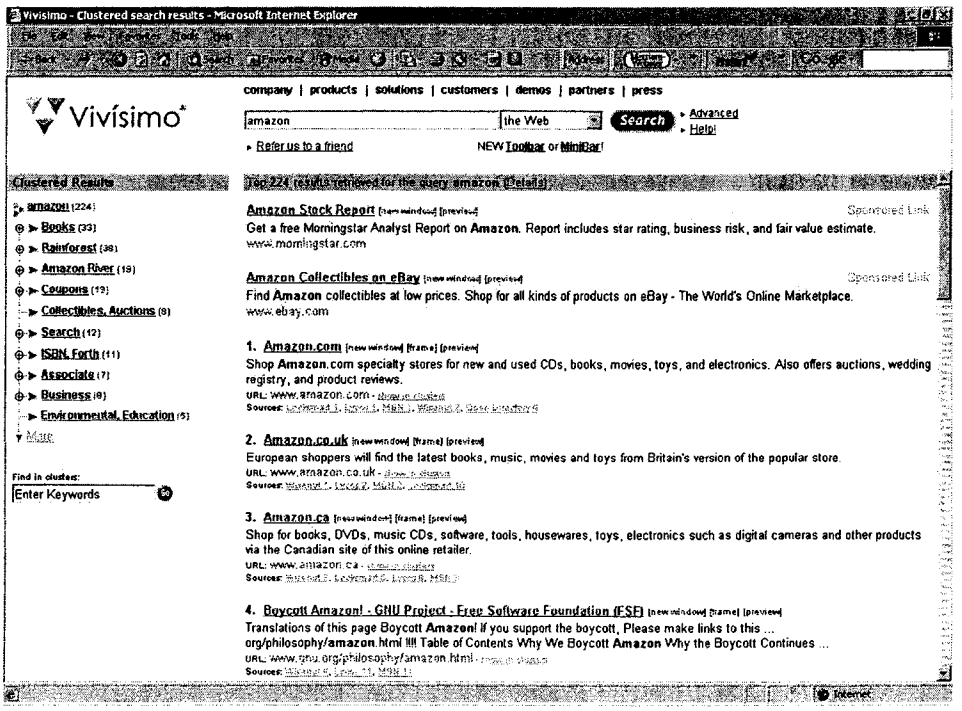

Figure 5.5: Search result from Vivisimo

Figure 5.5 shows the clustering results returned from Vivisimo for query "amazon". Because Vivisimo always presents 10 clusters in its initial search result (more clusters can be found by clicking the node labelled as "more"), we only checked the effectiveness of these 10 clusters and we also used "10" as the number of partitions for Categorizer for comparison.

	n=50					
	MK1 of Vivisimo			MK1 of Categorizer(Keyfreq)		
Query	$\beta = 1$	$\beta = 2$	$\beta = 0.5$	$\beta = 1$	$\beta = 2$	$\beta = 0.5$
1	0.524	0.561	0.479	0.538	0.595	0.464
2	0.429	0.474	0.375	0.385	0.412	0.355
3	0.167	0.074	0.242	0.125	0.186	0.054
4	0.273	0.344	0.184	0.391	0.507	0.205
5	0.667	0.444	0.762	0.2	0.091	0.286
6	0.714	0.687	0.737	0.529	0.545	0.512
7	0.52	0.483	0.552	0.5	0.615	0.286
8	0.4	0.464	0.318	0	0	0
9	0.687	0.597	0.615	0.4	0.4	0.4
10	1	1	1	0.538	0.516	0.559
11	0.5	0.412	0.565	0.556	0.444	0.63
	n=100					
	MK1 of Vivisimo			MK1 of Categorizer(Keyfreq)		
Query	$\beta = 1$	$\beta = 2$	$\beta = 0.5$	$\beta = 1$	$\beta = 2$	$\beta = 0.5$
1	0.636	0.63	0.643	0.417	0.493	0.314
2	0.358	0.372	0.343	0.333	0.365	0.298
3	0.394	0.275	0.479	0.158	0.184	0.13
4	0.277	0.286	0.267	0.263	0.364	0.125
5	0.565	0.39	0.662	0.5	0.286	0.615
6	0.667	0.71	0.608	0.667	0.638	0.691
7	0.579	0.579	0.576	0.467	0.545	0.355
8	0.375	0.419	0.324	0.238	0.167	0.298
9	0.875	0.909	0.8	0.667	0.667	0.667
10	1	1	1	0.655	0.597	0.699
11	0.667	0.524	0.643	0.556	0.556	0.556
	n=150					
	MK1 of Vivisimo			MK1 of Categorizer(Keyfreq)		
Query	$\beta = 1$	$\beta = 2$	$\beta = 0.5$	$\beta = 1$	$\beta = 2$	$\beta = 0.5$
1	0.5	0.444	0.545	0.44	0.521	0.327
2	0.416	0.425	0.406	0.5	0.539	0.453
3	0.481	0.314	0.583	0.265	0.338	0.174
4	0.267	0.28	0.254	0.321	0.371	0.262
5	0.576	0.417	0.667	0.833	0.722	0.881
6	0.81	0.815	0.783	0.676	0.647	0.7
7	0.495	0.498	0.491	0.443	0.522	0.331
8	0.474	0.419	0.519	0.259	0.206	0.306
9	0.818	0.71	0.767	0.762	0.731	0.786
10	1	1	1	0.714	0.662	0.667
11	0.692	0.524	0.75	0.65	0.557	0.711

Table 5.28: Performance comparison between Vivisimo and Categorizer in terms of MK1.

n	MK1 of Vivisimo			MK1 of Categorizer(Keyfreq)		
	$\beta = 1$	$\beta = 2$	$\beta = 0.5$	$\beta = 1$	$\beta = 2$	$\beta = 0.5$
50	3	5	2	8	6	9
100	0	2	1	10	9	10
150	3	8	4	8	3	7

Table 5.29: Summarization of performance comparison between Vivisimo and Categorizer.

n	E(1)	E(2)	E(0.5)
top-50	0.034	0.120	0.009
top-100	0.001	0.009	0.003
top-150	0.120	0.681	0.062

Table 5.30: One-tailed probabilities given by Wilcoxon test for keyfreq

n	E(1)	E(2)	E(0.5)
top-50	0.715	0.862	0.080
top-100	0.207	0.416	0.120
top-150	0.449	0.768	0.183

Table 5.31: One-tailed probabilities given by Wilcoxon test for snippet

We performed 3 sets of experiments for the eleven queries (see Table 5.7) on Vivisimo corresponding to different numbers of returned URLs, i.e., n = 50, 100 and 150. Table 5.27 shows the numbers of relevant URLs for different n values. The last column of the table gives the numbers of relevant URLs averaged over all the queries.

Table 5.28 shows the optimal cluster effectiveness of the Vivisimo clustering results for different numbers of top-ranked documents and for different queries. For comparison purpose, corresponding experimental results of Categorizer are also displayed in the table. Table 5.29 summarized the results in Table 5.28 and shows how many times each system has better optimal cluster effectiveness than the other. From Table 5.29, we can see that in most of the cases Categorizer has better MK1 values than Vivisimo under the same experimental conditions. The difference between Vivisimo and Categorizer was found to be statistically significant for most cases according to Wilcoxon test. Table 5.30 shows the one-tailed probabilities from the Wilcoxon test. Because Vivisimo clusters pages according to the page summarizations returned from search engines, we also compared Vivisimo with Categorizer when snippet is used as the page representation schemes. Table 5.31 shows the one-tailed significance levels given by Wilcoxon test. We can see that for most of the cases, the result is not significant. Again this shows the benefit of using keyphrases in query-specific clustering.

Discussion

Because we do not have access to Vivisimo's proprietary clustering algorithm, our comparisons are completely based on observation of results of certain queries. Our clustering scheme, K-Means bisecting algorithm over keyphrases, have more number of cases where the optimal cluster effectiveness was better. Categorizer significantly outperformed Vivisimo in terms of optimal cluster effectiveness under our experimental conditions. However, We need to perform more comprehensive and thorough research in order to reach to a generalized conclusion. There are some possible explanations as to why Categorizer exhibits better clustering quality in our experiments:

- The performance difference might be caused by different Web search engine sources (Categorizer uses Google and Vivisimo uses search service from MSN

etc). Because both Vivisimo and Categorizer are meta search engines that cluster results returned from other search engines, the different sources can have an impact on the clustering results.

- In our experiments, Vivisimo's clustering results were interpreted as flat partitions (we only examined the top-level clusters). Hence the experimental results are not optimal for Vivisimo because its lower-level clusters may have better E effectiveness. However, doing so is fair for the comparison because Categorizer only has one level of clusters and browsing lower-level clusters in Vivisimo definitely entails more effort from users since they have to first make decisions on the top-level nodes and then expand the promising node to access the contained lower-level nodes.

- The fact that when Categorizer used snippet as the clustering source, it did not significantly outperformed Vivisimo suggested that using keyphrases does help improve the effectiveness of Categorizer. Hence using keyphrase as the clustering source also contributes to the better performance of Categorizer.

Although we can not conclude that Categorizer is definitely a better system than Vivisimo just from the preliminary experimental results comparison, we did find Categorizer was able to generate good quality clustering in terms of optimal cluster effectiveness even when being compared to the current leading commercial clustering search engine. By experimentally studying the performance of Vivisimo, we were also able to provide some insight into how Vivisimo performs in terms of objective performance measure, i.e., optimal cluster effectiveness. Thus it helps us better understand the status of our research in the context of the current clustering quality of the clustering searching engine in the real World.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The research we conducted explored a keyphrase extraction algorithm that uses HTML formatting elements information and its application in query-specific document clustering in the domain of the World Wide Web. We believe the meaningful way to evaluate the quality of extracted keyphrases is to measure its effectiveness in the context of its application. The conclusion of the research is structured according to the objectives we set for the research (see Chapter 1).

6.1.1 Using Keyphrases as Content-based Address

In the first part of the research we studied the quality of keyphrases generated by Extoken in terms of helping in creating "content-based address" for Web documents. Our preliminary experimental results showed that the keyphrases extracted by Extoken were effective when they were being used to create search engine queries to retrieve the original Web documents from the Web. In the meantime, results suggest that there are no significant difference between the two weighting schemes that we proposed in the algorithm: HLOGN and HATF. They both substantially outperformed method that is merely based on lexical statistics information. The fact that both HLOGN and HATF significantly outperformed FREQ scheme demonstrated the effectiveness of using HTML formatting elements in keyphrase extraction. Their superiority over RANDOM scheme also demonstrated the effectiveness of selectively extracting topical terms.

6.1.2 The Effectiveness of Query-specific Clustering

Our main focus of the research was to experimentally study application of keyphrases extraction in the context of post-retrieval Web documents clustering. What distinguishes this research from a lot of previous works is that we investigated the application of partitional clustering algorithm with different document representations in query-specific document clustering in the domain of the Web. The main issues that we aimed to investigate include the following: Firstly, how different Web document representations (i.e., keyphrases, full document and snippet) affect the effectiveness of query-specific Web document clustering; secondly, whether or not clustering solution provided by query-specific clustering can actually outperform the

traditional ranked list presentation; thirdly, how clustering effectiveness varies when different numbers of top-ranked documents are clustered; and lastly for partitional clustering how the number of partitions generated by clustering process influence the clustering effectiveness.

Which page representation scheme is the best?

Our experimental results suggest that there is no statistical significance for the average clustering effectiveness between the three page representation schemes. However, the average clustering effectiveness yielded by keyfreq tend to be more stable (it has the lowest standard deviation for relative effectiveness between the three schemes). In other words, when the three schemes are compared, keyfreq's effectiveness tend to be either the best or the one closest to the best. Furthermore, keyfreq showed higher probability of yielding the best optimal clustering effectiveness across different experimental conditions (n and m) for $\beta = 1$ and 0.5.

The usefulness of using keyphrases in query-specific clustering was also corroborated when we compared the effectiveness between Vivisimo and Categorizer. We also argued that although keyphrase extraction module is not present in popular Web search engines, the current architecture of Web search engines makes it easy for them to incorporate this module. Once the keyphrases are computed for Web pages, the succinctness and topicality make them a more efficient and qualitatively superior Web page source candidate for clustering. The extracted keyphrases can even be used as the substitute for Web page snippet because of their topicality. The implication of these results is that they provided solid experimental evidence for the benefit of the application of keyphrase extraction technology in query-specific document clustering in light of their better clustering quality. Thus our preliminary results can serve as the motivation for researchers to develop better keyphrase extraction algorithms in the field of Web document clustering and searching.

Is clustering solution better than the ranked list provided by general search engine?

Our research provided experimental evidence that query-specific clustering as an alternative presentation for query retrieval results has the potential to be more effective than the ranked list. In our experiments, clustering solutions outperformed

the ranked lists under various experimental parameters (e.g., the number of top-ranked pages and different document collections). For most of the experimental conditions. the results are statistically significant. In the course of performing our experiments, we found that clustering is normally most useful when the searched topic potentially has different orientations. Apparently if all source pages are pertinent to the same exact topic, there is no need to cluster them at all (as the clustering results would be of little use, the amount of time users spend on picking out the relevant cluster to further browse would not be justified).

Through the study of user queries and search log, we observed that users' information needs (reflected by formulated queries) play a vital role as to how much clustering can help. Users can be characterized by the nature of their queries which can be seen as the representation of their information needs. Queries can be generally classified into three types in terms of how they serve the purpose of getting needed information from search engines:

1. **Queries that consist of unambiguous and unique terms which match users' information needs exactly.** For instance, to find the movie schedule of the local city, users probably will spontaneously input "movie guide Calgary" (for users living in Calgary). Users who want to know where is YMCA's Web site mostly likely would type in query "YMCA" to the search engine. This kind of queries are generally handled very well by major search engines like Google and AskJeeves;

2. **Queries that include obscure and general terms or polysemous terms.** For example, "salsa" could refer to a special sauce or a music genre. "Amazon" could mean the popular online bookstore or the famous river in South America. "Apple" is a fruit but also a well-known computer company. The vagueness of query terms can be caused by users' inexperience in query construction, the nature of the question itself and the combination of both. Due to the nature and volume of the Web, polysemy tends to be more common on the Web compared with traditional structured database. This type of queries can be formulated unconsciously (not aware of the innate vagueness) or intentionally (users would like to survey a general topic);

3. **Imprecise and misleading queries.** Users who are not familiar with the jargons or vocabularies in certain topics may not use the right words when they query subjects in those areas. Users lacking of query composition skills sometimes even give queries which do not represent their true information needs.

We observed the similar pattern happening in the domain of the Web as the one found by Allen, Obry and Littman [3]. Generally speaking traditional search engines do a good job handling the first type of queries. The ranked list works satisfactorily in this case and users normally can find relevant pages in the top-ranked pages swiftly. One characteristic of most of this type of queries is that users will almost immediately stop browsing once they find one relevant page that answers their questions or helps them achieve their tasks.

It is the other two types of queries that query-specific clustering can help the most. By organizing result pages into topically-related groups, clustering gives users a chance to identify relevant document sets among unrelated document groups. Hence it enables users to "jump" to the promising groups directly. This is especially true when the query terms are polysemous and clustering provides categories that match the multi-aspects of the topic naturally. Furthermore, by comprehending the topic structure of the query result aided by clustering, users can also refine or modify their original queries in order to achieve better retrieval precision from conventional search engines. Clustering-based query result navigation can also help users quickly gain an overview of the document set generated by the query. Therefore it is very helpful when users want to explore a general topic through search engines. The clustering solutions provided by query-specific clustering help users perceive the intrinsic multi-facet topics embedded in the retrieval results returned by traditional search engines. These "flavors" of queries justify the validity of using clustering and at the same time stresses that clustering solution should be used as a supplementary presentation to the ranked list instead of a complete replacement.

How did different experimental conditions affect clustering effectiveness?

The experimental results in our research work implied that lower values of the number of retrieved top-ranked documents tend to generate better quality clustering

solutions than higher values. However, in order to cover as much retrieved information as possible, we need to have more top-ranked documents. Therefore it is a trade-off. During our experimentation, we found some reasonably big numbers like 100 work pretty well in serving the two requirements. One of the other contributions made by this research is that we performed the experimental evaluation of how different numbers of partitions generated by partitional clustering algorithm influenced the Web document clustering effectiveness. We explored the variation of clustering effectiveness across different numbers of partitions for keyphrase based K-Means via bisections clustering algorithm. Our analysis of the results suggests such a pattern: extreme numbers of partitions seem not to work very well with partitional clustering and on the other hand numbers in the middle of the spectrum tend to give better results.

Categorizer VS. Vivisimo

In our experiments, we also compared our system with the leading clustering search engine Vivisimo. Our preliminary results revealed that using keyphrase-based clustering, Categorizer significantly outperformed Vivisimo for most experimental conditions when the flat structure of Categorizer and the top-level nodes of Vivisimo were compared. The possible reasons why Categorizer outperformed Vivisimo can be summarized as follows:

- Different clustering sources. In our experiments, We used keyphrases extracted from the Web pages while Vivisimo used page summarization returned from search engines. When Categorizer used snippet as clustering source, statistical significance was not achieved most of the times.
- Different clustering algorithms. We used bisecting K-Means partitional clustering algorithm. The hierarchical clustering algorithm used by Vivisimo is not disclosed.
- Different backend search engines. We used Google as our backend search engine and Vivisimo used other search engines (e.g., MSN etc).
- We only compared the top-level clusters of the hierarchical clustering results from Vivisimo with our partitional clustering results.

Although we sill can not claim the our system is definitely better than Vivisimo, our experimental results can be taken as evidence of that partitional clustering based on keyphrases can produce good quality clustering solutions. In the meantime, it can also serve as the motivation for researchers to further study this area.

limitations of Query-specific Clustering

One limitation of query-specific clustering is that clustering result depends on the quality of the backend search engine. As Allen, Obry and Littman concluded [3], if the search engine does not retrieve the relevant documents, query-specific clustering will definitely fail. On the other hand if the ranked list from the search engine is too good, i.e., the top hits exactly meet users' information needs, query-specific clustering won't improve the retrieval either.

6.2 Future Work

During the process of our research, we realized there are still a lot of challenges and problems that need to be addressed in the area of the research we were doing. Furthermore our research work can also be expanded and furthered in some directions.

One potentially beneficial addition to our research evaluation would be evaluation data from the real world gained by comprehensive user study. This applies to both the keyphrase generation and Web document clustering. Document clustering user study in the domain of the World Wide Web has not been thoroughly researched, although the counterpart in the context of traditional IR system has been extensively studied by researchers. The intrinsic difference between the World Wide Web and traditional IR systems and the distinct user behaviors [38] justify the need of such a study. The popularity of the World Wide Web and its impact on people's everyday life make it necessary and practical for the research community to invest more effort to conduct systematic and thorough research work in this area. However the volume of the Web, the heterogeneous and dynamic nature of the Web and its huge variety of user groups make it a very challenging task to perform such a study in an unbiased and meaningful way. Although as we mentioned before, evaluation from users' perspective can be highly subjective, we think it can definitely

contribute to helping us see the whole picture more clearly if we do not solely rely on it. It can also help us find problems with the system that won't be found using objective measurements, e.g., readability of the results as well as the usability of the prototype systems that we developed.

There is still plenty of room for us to improve our keyphrase extraction algorithm. There are more semantic information embedded in HTML that we can use to enhance Extoken (e.g., the alternative text for multimedia such as images and video as well as relative position of text in the document). The design of the algorithm makes it very easy to add more heuristics into the system. A more sophisticated keyphrase extraction algorithm will definitely benefit its application in different areas. Keyphrases extraction can be used in other areas of Web application too. Keyphrases assisted browse and search systems (e.g., using keyphrases as a synopsis of the source document, helping users reformulate queries with topical keyphrases etc.) deserve further study in itself.

The presentation of document clustering solution is vital to the overall usability of an online Web document clustering system. One important aspect of the presentation is the cluster node labelling. It is the cluster label that gives users the direct hint as to which cluster may be of interest. Without proper labelling, a perfect clustering solution can be useless to users since they can not identify the right clusters. However it is very difficult to find the balance between being concise and also informative. In order for users to find the most relevant cluster, we need to provide them with more information, but to serve the very basic purpose of using clustering as an alternative output method to the ranked list, we have to make the representation of clustering succinct. Cluster node labelling as well as other clustering visualization methods should by all means receive more efforts from the research community.

As a future extension to this research, we can investigate more partitional clustering algorithms as well as clustering algorithms having other traits (e.g., clustering algorithms that allow overlapping) to see if they can produce equally good or even better results. So far prior research work has stressed evaluating document clustering from the perspective of information retrieval (i.e., precision and recall). The utility of using clustering to reveal the different aspects of the document set calls for the need to address the performance evaluation based on contextual metrics such

as topicality in addition to relevancy. However it is rather difficult to come up with such a measurement that can be calculated in an objective manner.

The "literal search" method employed by most of the current search engines sometimes worsen the problems. So it is also meaningful to use Natural Language Processing techniques to address the "query ambiguity" problem posed by current search engines from a different perspective. Actually NLP has been adopted by some commercial search engines, e.g., AskJeeves [24]. But due to the limitation of current research level of NLP, it is still not very practical to implement a system using fully-automated NLP technology. The natural language processing ability of AskJeeves still strongly relies on human examination and editing.

The web has been and also will be developing rapidly in future. Because of its volatility and scale, it has presented a huge methodological challenge on researches in this area. Web search engines crawl on the Web without break. Documents popular today may become obsolete in a few months. Searching trends also change in an unpredictable way. Researches in this area should also be updated and study the problems in a sustained way in order to keep up with the speed of the growth of the Web. There probably won't be a day when we can reach a valid conclusion once and for all, but what we can certainly do is make the Web more and more easy to use.

Appendix A

Prototype Systems Implementation Details

The keyphrase extraction engine of *Phrastractor*, which is the implementation of Extoken algorithm, was developed in C++ and the Web interface was implemented using JavaScript and CGI.

Several different programming languages including C++, Perl and JavaScript were used in the implementation of Categorizer. The selection of languages is solely based on that their different characteristics make them good candidates for different modules in the system. Processing Unit mostly employed C++ because of the requirement of high performance. Coordinating Unit used Perl due to its embedded capability of string manipulation and flexibility. We chose to use a combination of C++ , Perl and JavaScript to implement Web User Interface.

Appendix B

Categorizer Efficiency Optimization

Due to the slowness of downloading Web pages, we devised a server side history cache to speed things up for Categorizer. Basically for each recently submitted query we keep the copies of all result Web pages corresponding to the number of top-ranked URLs specified by the user for the query. When the same query is submitted again, the system will detect that there exists a local cached copy of the needed Web pages and they will be used instead of downloading the Web pages from the Internet. Hence the result can be delivered in a shorter amount of time.

Categorizer will download the Web documents if a local copy can not be found in cache database. Because of various reasons (e.g., some of the Web servers have slow response time, some Web Pages are not available at the time of downloading), the download agent in Categorizer gives each downloading session a pre-specified time limit. Once it times out, the agent will try again. The process repeats until the preset number of times is reached. Then the agent will move to the next URL.

Appendix C

Percentage of Paid Links in Meta Search Engines

In this sampling test, search query "canada" is used to calculate the what percentage of links in the first page of search results were paid listings [20]. Because usually users do not change the defaults search settings default settings were used when the search was conducted.

Meta Search	Paid Links	Total Links	Paid %
Dogpile	30	35	86%
qbSearch	66	98	67%
MetaCrawler	13	25	52%
Mamma	6	15	40%
Search.com	10	29	34%
ProFusion	2	14	14%
Ixquick	1	10	10%
Vivisimo	0	20	0%

Table C.1: Percentage of Paid Links[32]

Appendix D

The Most Popular Search Engines

The chart below shows the most popular search sites in the United States, as based on audience reach for January 2003 [19]. Audience reach is the percentage of US home and work internet users estimated to have searched on each site at least once during the month through a web browser or some other "online" means. For January 2003, there were an estimated 134 million active at home and at work internet users in the US.

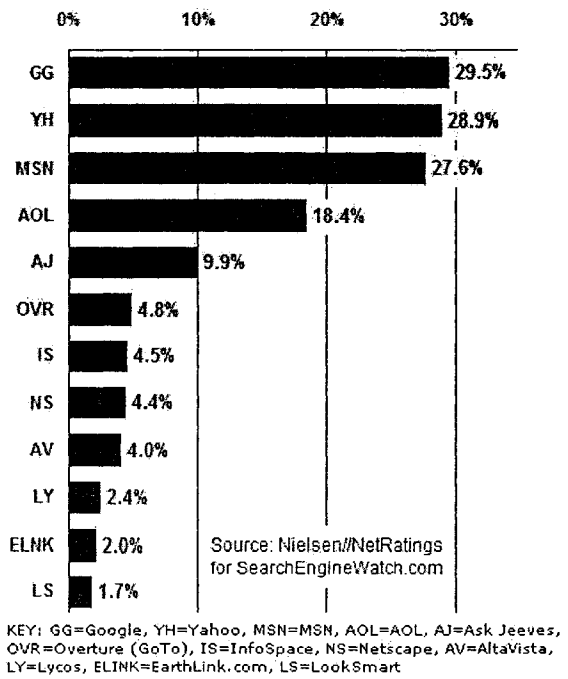

Figure D.1: Most popular search engines ranking in US

Appendix E

Effectiveness variation across increasing values of partition number for document collections

Table E.1 and E.2 present the results of relative optimal cluster effectiveness represented by MK1 across for all document sets across increasing numbers of partitions with an increment of 10 for top100 and top150 ranked documents.

The following figures show the distribution of frequencies of different numbers of partitions output from clustering that produce the best optimal cluster effectiveness for different β values and for three page representation schemes over all document collections. $E(1), E(2)$ and $E(0.5)$ denotes MK1 when β is 1,2 and 0.5 respectively

.

$\beta = 1$												
m	1	2	3	4	5	6	7	8	9	10	11	Avg.
6	1.00	1.35	1.00	1.43	1.50	1.13	1.61	4.03	1.03	1.07	1.00	1.47
5	1.00	1.54	1.00	1.43	1.50	1.13	1.61	4.03	1.03	1.07	1.00	1.49
10	1.12	1.00	2.26	1.43	1.00	1.00	1.61	4.03	1.00	1.15	1.00	1.51
11	1.12	1.00	2.26	1.43	1.00	1.00	2.14	4.03	1.00	1.15	1.00	1.56
7	1.12	1.35	2.26	1.43	1.50	1.02	1.61	4.03	1.03	1.07	1.00	1.58
9	1.12	1.35	2.26	1.43	1.50	1.00	1.61	4.03	1.00	1.15	1.00	1.59
8	1.12	1.35	2.26	1.43	1.50	1.00	1.61	4.03	1.03	1.15	1.00	1.59
14	1.12	1.11	5.41	2.47	1.00	1.09	2.14	1.00	1.00	1.17	1.00	1.68
15	1.12	1.11	5.41	2.47	1.00	1.09	2.14	1.00	1.00	1.17	1.00	1.68
19	1.62	1.11	5.41	2.47	1.00	1.00	2.14	1.00	1.03	1.20	1.00	1.73
16	1.62	1.11	5.41	2.47	1.00	1.09	2.14	1.00	1.03	1.17	1.00	1.73
17	1.62	1.11	5.41	2.47	1.00	1.09	2.14	1.00	1.03	1.17	1.00	1.73
18	1.62	1.11	5.41	2.47	1.00	1.09	2.14	1.00	1.03	1.17	1.00	1.73
20	1.62	1.11	5.41	2.47	1.00	1.00	2.14	1.00	1.03	1.20	1.12	1.74
12	1.12	1.00	5.41	1.43	1.00	1.00	2.14	4.03	1.00	1.15	1.00	1.85
4	1.00	1.54	1.00	1.00	1.75	1.13	1.00	8.97	1.08	1.00	1.00	1.86
13	1.12	1.00	5.41	2.47	1.00	1.00	2.14	4.03	1.00	1.17	1.00	1.94
3	1.25	1.54	1.00	1.00	1.75	1.06	1.15	11.41	1.08	1.00	1.00	2.11
2	1.25	2.07	6.34	1.86	1.85	1.06	1.15	11.41	1.08	1.00	1.29	2.76
$\beta = 2$												
m	1	2	3	4	5	6	7	9	10	11	Avg.	
4	1.13	1.03	1.00	1.00	2.58	1.28	1.34	3.78	1.11	1.00	1.00	1.48
5	1.13	1.03	1.00	2.09	1.91	1.28	2.78	1.84	1.19	1.43	1.00	1.51
3	1.00	1.03	1.00	1.00	2.58	1.00	1.00	4.96	1.11	1.00	1.00	1.52
6	1.13	1.13	1.00	2.09	1.91	1.28	2.78	1.84	1.19	1.43	1.00	1.52
10	1.77	1.00	6.34	2.09	1.00	1.19	2.78	1.84	1.25	1.73	1.25	2.02
7	1.77	1.13	6.34	2.09	1.91	1.28	2.78	1.84	1.19	1.43	1.00	2.07
11	1.77	1.00	6.34	2.09	1.00	1.19	3.66	1.84	1.25	1.73	1.25	2.10
8	1.77	1.13	6.34	2.09	1.91	1.19	2.78	1.84	1.19	1.73	1.25	2.11
9	1.77	1.13	6.34	2.09	1.91	1.19	2.78	1.84	1.25	1.73	1.25	2.12
2	1.00	1.32	8.34	1.09	2.92	1.00	1.00	4.96	1.00	1.00	1.23	2.26
12	1.77	1.00	17.03	2.09	1.00	1.19	3.66	1.84	1.25	1.73	1.25	3.07
14	1.77	1.32	17.03	3.28	1.00	1.40	3.66	1.00	1.25	2.05	1.25	3.18
15	1.77	1.32	17.03	3.28	1.00	1.40	3.66	1.00	1.25	2.05	1.25	3.18
13	1.77	1.00	17.03	3.28	1.00	1.19	3.66	1.84	1.25	2.05	1.25	3.21
19	2.49	1.32	17.03	3.28	1.00	1.37	3.66	1.00	1.36	2.06	1.25	3.26
16	2.49	1.32	17.03	3.28	1.00	1.40	3.66	1.00	1.36	2.05	1.25	3.26
17	2.49	1.32	17.03	3.28	1.00	1.40	3.66	1.00	1.36	2.05	1.25	3.26
18	2.49	1.32	17.03	3.28	1.00	1.40	3.66	1.00	1.36	2.05	1.25	3.26
20	2.49	1.32	17.03	3.28	1.00	1.37	3.66	1.00	1.36	2.06	1.47	3.28
$\beta = 0.5$												
m	1	2	3	4	5	6	7	9	10	11	Avg.	
14	1.00	1.00	1.83	2.00	1.00	1.27	1.34	1.00	1.07	1.00	1.00	1.23
15	1.00	1.00	1.83	2.00	1.00	1.27	1.34	1.00	1.07	1.00	1.00	1.23
19	1.37	1.00	1.83	2.00	1.00	1.00	1.34	1.00	1.00	1.02	1.00	1.23
20	1.37	1.00	1.83	2.00	1.00	1.00	1.34	1.00	1.00	1.02	1.07	1.24
16	1.37	1.00	1.83	2.00	1.00	1.20	1.34	1.00	1.00	1.00	1.00	1.25
17	1.37	1.00	1.83	2.00	1.00	1.20	1.34	1.00	1.00	1.00	1.00	1.25
18	1.37	1.00	1.83	2.00	1.00	1.20	1.34	1.00	1.00	1.00	1.00	1.25
10	1.00	1.58	1.21	1.00	1.00	1.27	1.12	12.42	1.07	1.14	1.00	2.16
11	1.00	1.58	1.21	1.00	1.00	1.27	1.34	12.42	1.07	1.14	1.00	2.18
12	1.00	1.58	1.83	1.00	1.00	1.27	1.34	12.42	1.07	1.00	1.00	2.23
9	1.00	2.54	1.21	1.00	1.35	1.27	1.12	12.42	1.07	1.14	1.00	2.28
8	1.00	2.54	1.21	1.00	1.35	1.23	1.12	12.42	1.16	1.14	1.00	2.29
7	1.00	2.54	1.21	1.00	1.35	1.23	1.12	12.42	1.16	1.11	1.13	2.30
13	1.00	1.58	1.83	2.00	1.00	1.27	1.34	12.42	1.07	1.00	1.00	2.32
6	1.34	2.54	1.00	1.00	1.35	1.47	1.12	12.42	1.16	1.11	1.13	2.33
5	1.34	3.17	1.00	1.00	1.35	1.47	1.12	12.42	1.16	1.11	1.13	2.39
4	1.34	3.17	1.00	1.55	1.49	1.47	1.00	26.38	1.26	1.10	1.13	3.72
3	1.82	3.17	1.00	1.55	1.49	1.45	1.36	31.96	1.26	1.10	1.13	4.30
2	1.82	4.11	5.24	3.58	1.55	1.45	1.36	31.96	1.28	1.10	1.44	4.99

Table E.1: Relative optimal cluster effectiveness for top-100 documents for keyfreq across increasing values of m.

$\beta = 1$												
m	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Avg.**
6	1.11	0.58	1.00	1.02	1.27	1.00	1.00	1.30	1.10	1.05	1.00	1.04
7	1.11	0.58	1.00	1.02	1.27	1.00	1.00	1.30	1.10	1.05	1.00	1.04
8	1.11	0.58	1.00	1.02	1.25	1.00	1.51	1.30	1.10	1.05	1.00	1.08
5	1.11	0.58	1.00	1.02	1.27	1.00	1.00	2.73	1.03	1.00	1.00	1.16
4	1.11	0.59	1.00	1.00	1.32	1.03	1.00	2.73	1.03	1.00	1.00	1.16
3	1.15	0.59	1.00	1.00	1.33	1.10	1.17	2.73	1.03	1.00	1.09	1.20
9	1.00	0.58	3.35	1.02	1.17	1.00	1.51	1.30	1.10	1.05	1.00	1.28
10	1.00	0.50	3.35	1.02	1.17	1.00	1.51	1.30	1.10	1.18	1.00	1.28
11	1.00	0.50	3.35	1.02	1.17	1.00	1.51	1.30	1.10	1.18	1.08	1.29
12	1.00	0.50	3.35	1.02	1.17	1.11	1.51	1.30	1.10	1.18	1.08	1.30
16	1.00	0.50	3.35	1.30	1.12	1.11	2.20	1.00	1.10	1.18	1.08	1.36
13	1.00	0.50	3.35	1.02	1.17	1.11	2.20	1.30	1.10	1.18	1.08	1.36
14	1.00	0.50	3.35	1.30	1.12	1.11	2.20	1.30	1.10	1.18	1.08	1.38
15	1.00	0.50	3.35	1.30	1.12	1.11	2.20	1.30	1.10	1.18	1.08	1.38
19	1.00	0.42	4.61	1.30	1.00	1.11	2.29	1.00	1.00	1.18	1.08	1.45
18	1.00	0.42	4.61	1.30	1.00	1.11	2.20	1.00	1.10	1.18	1.08	1.45
17	1.00	0.50	4.61	1.30	1.00	1.11	2.20	1.00	1.10	1.18	1.08	1.46
20	1.00	0.42	4.61	1.30	1.00	1.11	2.51	1.00	1.00	1.18	1.08	1.47
2	1.15	0.74	5.53	1.58	1.36	1.10	1.17	3.66	1.03	1.00	1.09	1.77

$\beta = 2$												
m	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**9**	**10**	**11**	**Avg.**	
3	1.00	0.40	1.00	1.00	1.44	1.01	1.00	1.82	1.00	1.00	1.00	1.06
4	1.36	0.40	1.00	1.00	1.42	1.00	1.48	1.82	1.17	1.00	1.04	1.15
6	1.36	0.54	1.00	1.48	1.33	1.15	1.48	1.00	1.42	1.36	1.04	1.20
7	1.36	0.54	1.00	1.48	1.33	1.15	1.48	1.00	1.42	1.36	1.04	1.20
5	1.36	0.54	1.00	1.48	1.33	1.15	1.48	1.82	1.17	1.00	1.04	1.22
8	1.36	0.54	1.00	1.48	1.30	1.15	3.00	1.00	1.42	1.36	1.04	1.33
2	1.00	0.54	7.18	1.26	1.53	1.01	1.00	2.59	1.00	1.00	1.00	1.74
9	1.78	0.54	10.24	1.48	1.17	1.15	3.00	1.00	1.42	1.36	1.04	2.20
10	1.78	0.54	10.24	1.48	1.17	1.15	3.00	1.00	1.42	1.74	1.04	2.23
11	1.78	0.54	10.24	1.48	1.17	1.15	3.00	1.00	1.42	1.74	1.34	2.26
12	1.78	0.54	10.24	1.48	1.17	1.41	3.00	1.00	1.42	1.75	1.34	2.29
13	1.78	0.54	10.24	1.48	1.17	1.41	4.21	1.00	1.48	1.75	1.34	2.40
14	1.78	0.54	10.24	2.06	1.12	1.41	4.21	1.00	1.48	1.75	1.34	2.45
15	1.78	0.54	10.24	2.06	1.12	1.41	4.21	1.00	1.48	1.75	1.34	2.45
16	1.78	0.54	10.24	2.06	1.12	1.41	4.21	1.39	1.48	1.97	1.34	2.50
18	1.78	0.51	14.12	2.06	1.00	1.41	4.21	1.39	1.48	1.97	1.34	2.84
17	1.78	0.54	14.12	2.06	1.00	1.41	4.21	1.39	1.48	1.97	1.34	2.85
19	1.78	0.51	14.12	2.06	1.00	1.41	4.36	1.39	1.43	1.97	1.34	2.85
20	1.78	0.51	14.12	2.06	1.00	1.41	4.69	1.39	1.43	1.97	1.34	2.88

$\beta = 0.5$												
m	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**9**	**10**	**11**	**Avg.**	
19	1.00	0.27	1.76	1.00	1.00	1.00	1.51	1.00	1.00	1.00	1.00	1.05
16	1.00	0.45	1.44	1.00	1.10	1.00	1.48	1.00	1.21	1.00	1.00	1.06
18	1.00	0.27	1.76	1.00	1.00	1.00	1.48	1.00	1.21	1.00	1.00	1.07
20	1.00	0.27	1.76	1.00	1.00	1.00	1.69	1.00	1.00	1.00	1.00	1.07
17	1.00	0.45	1.76	1.00	1.00	1.00	1.48	1.00	1.21	1.00	1.00	1.08
12	1.00	0.45	1.44	1.07	1.14	1.00	1.01	3.36	1.21	1.00	1.00	1.24
11	1.00	0.45	1.44	1.07	1.14	1.02	1.01	3.36	1.26	1.00	1.00	1.25
10	1.00	0.45	1.44	1.07	1.14	1.02	1.01	3.36	1.26	1.00	1.12	1.26
14	1.00	0.45	1.44	1.00	1.10	1.00	1.48	3.36	1.21	1.00	1.00	1.28
15	1.00	0.45	1.44	1.00	1.10	1.00	1.48	3.36	1.21	1.00	1.00	1.28
9	1.00	0.62	1.44	1.07	1.14	1.02	1.01	3.36	1.26	1.06	1.12	1.28
13	1.00	0.45	1.44	1.07	1.14	1.00	1.48	3.36	1.21	1.00	1.00	1.29
8	1.70	0.62	1.00	1.07	1.20	1.02	1.01	3.36	1.26	1.06	1.12	1.31
6	1.70	0.62	1.00	1.07	1.21	1.02	1.00	3.36	1.26	1.06	1.12	1.31
7	1.70	0.62	1.00	1.07	1.21	1.02	1.00	3.36	1.26	1.06	1.12	1.31
5	1.70	0.62	1.00	1.07	1.21	1.02	1.00	7.07	1.25	1.07	1.12	1.65
4	1.70	0.69	1.00	1.51	1.24	1.12	1.00	7.07	1.25	1.07	1.12	1.71
3	1.91	0.69	1.00	1.51	1.25	1.19	1.40	7.07	1.28	1.07	1.24	1.78
2	1.91	0.82	4.58	2.46	1.27	1.19	1.40	8.92	1.28	1.07	1.24	2.38

Table E.2: Relative optimal cluster effectiveness for top-150 documents for keyfreq across increasing values of m.

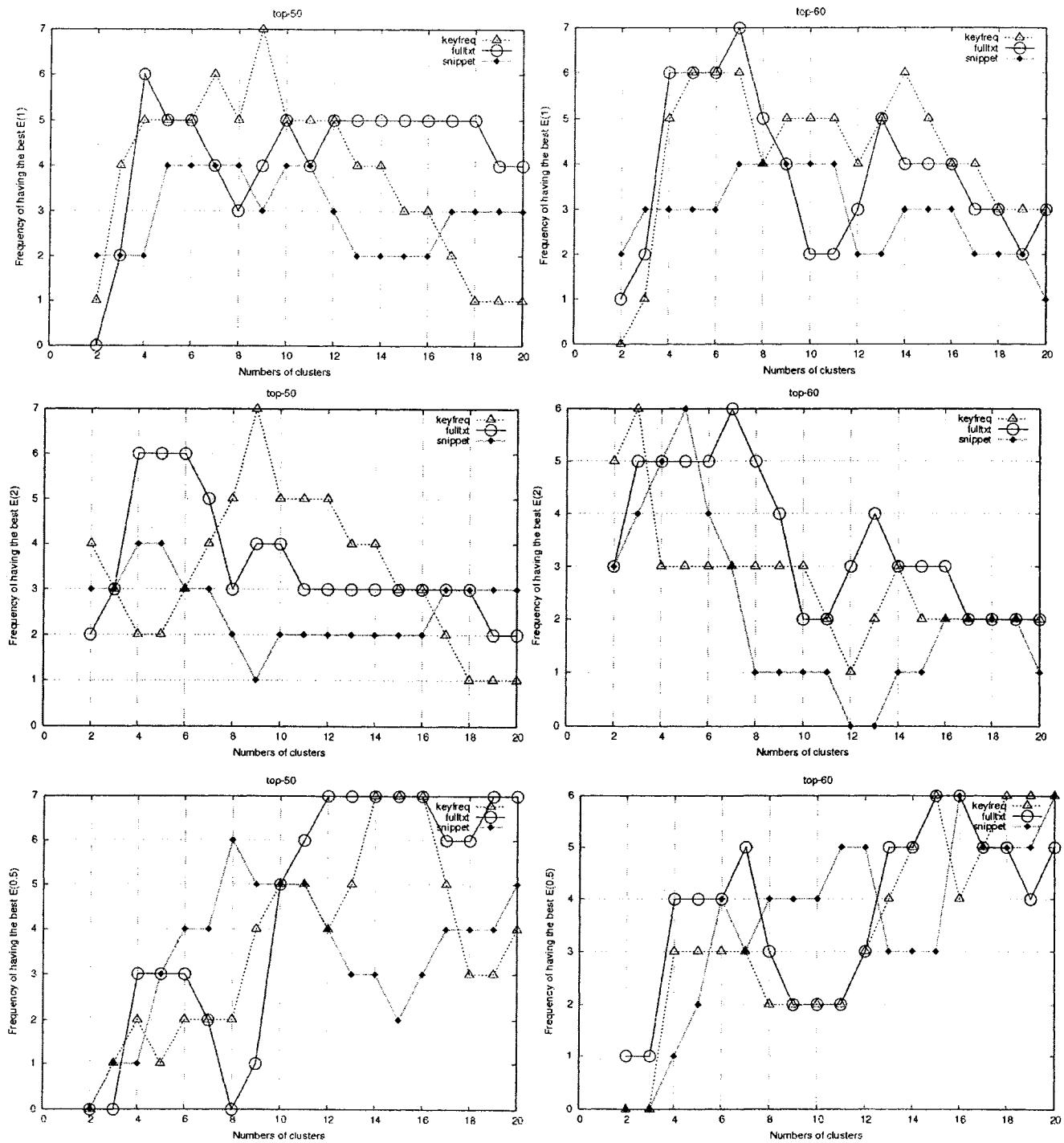

Figure E.1: Frequency of having the best E value.

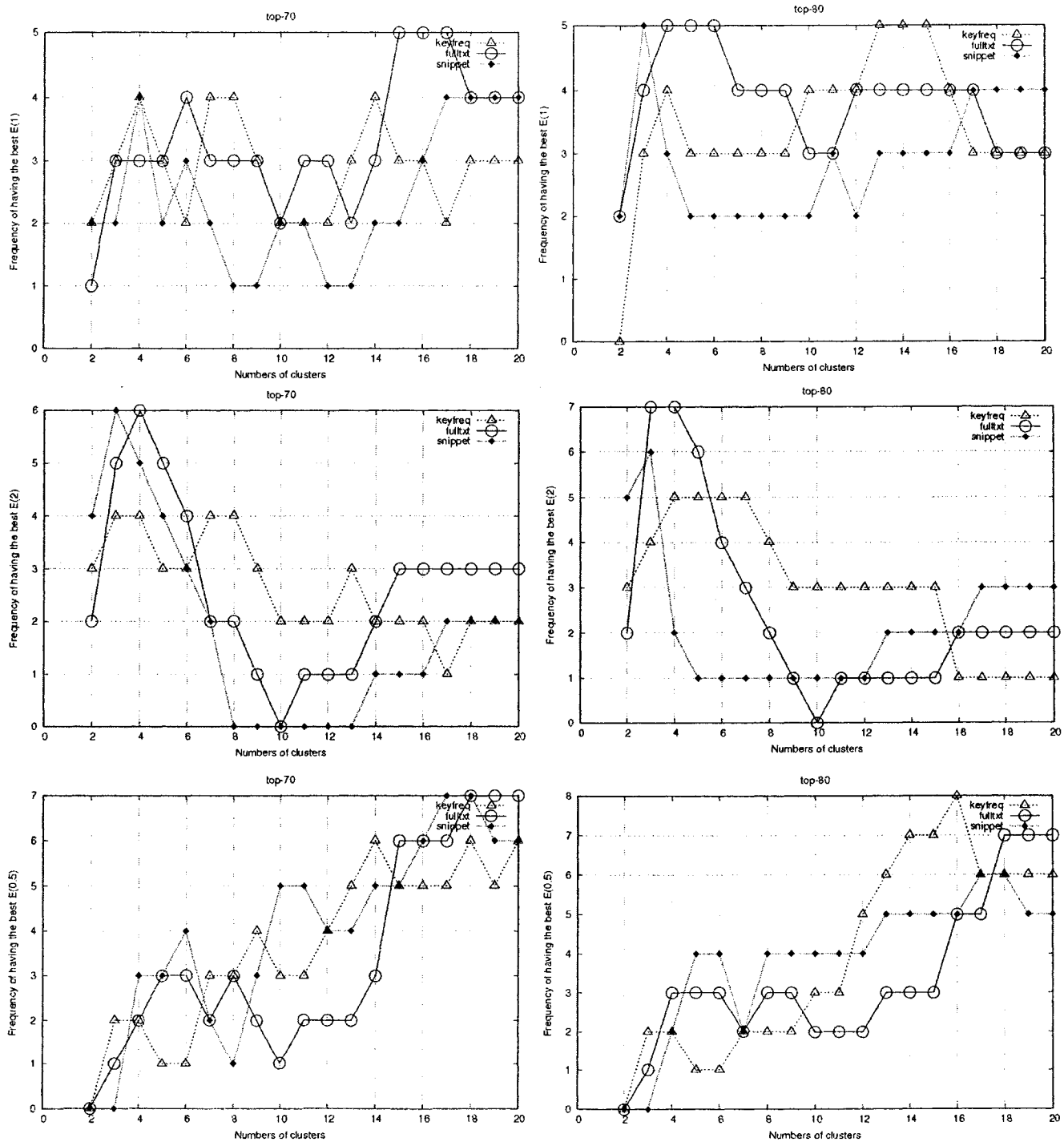

Figure E.2: Frequency of having the best E value.

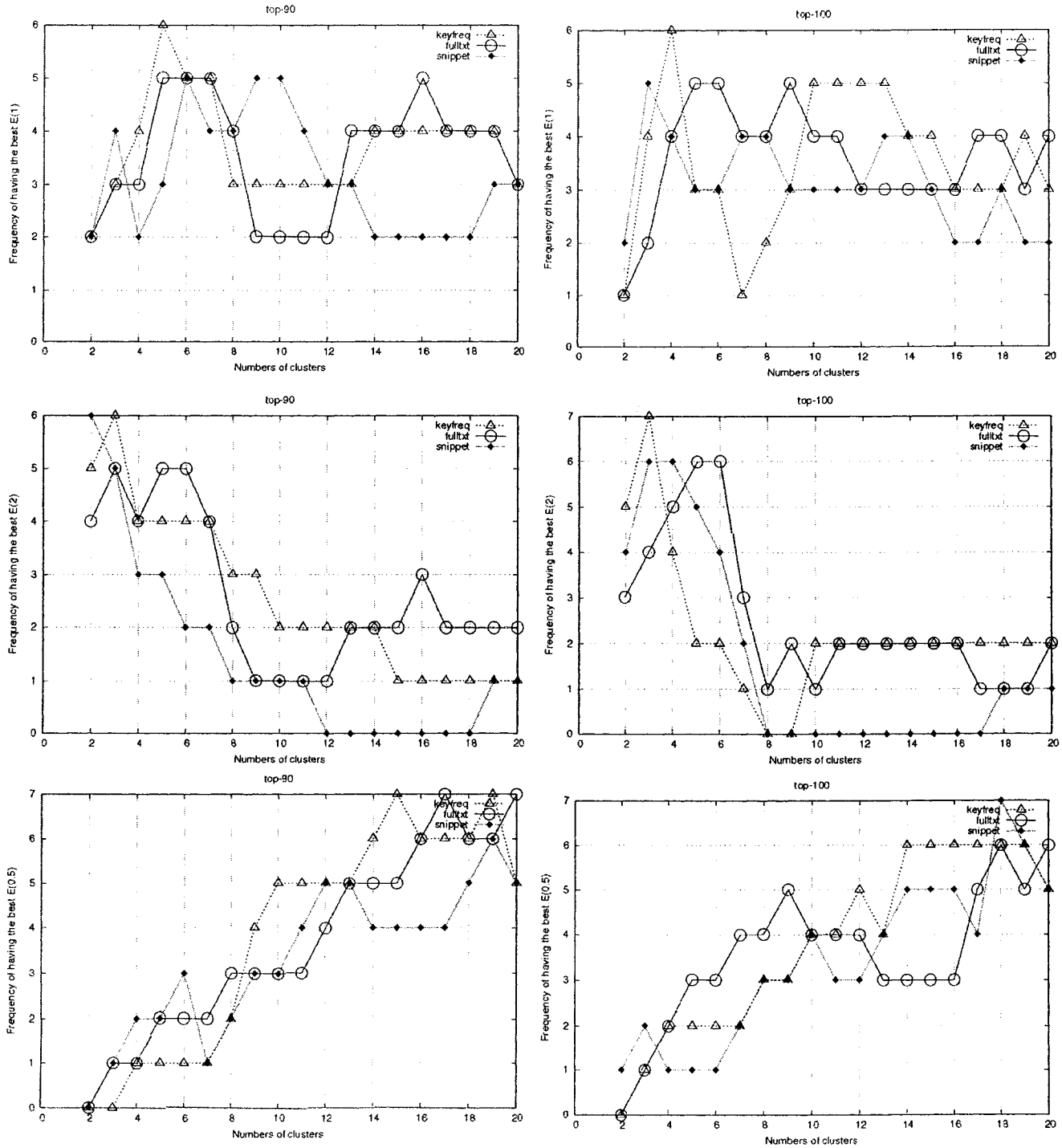

Figure E.3: Frequency of having the best E value.

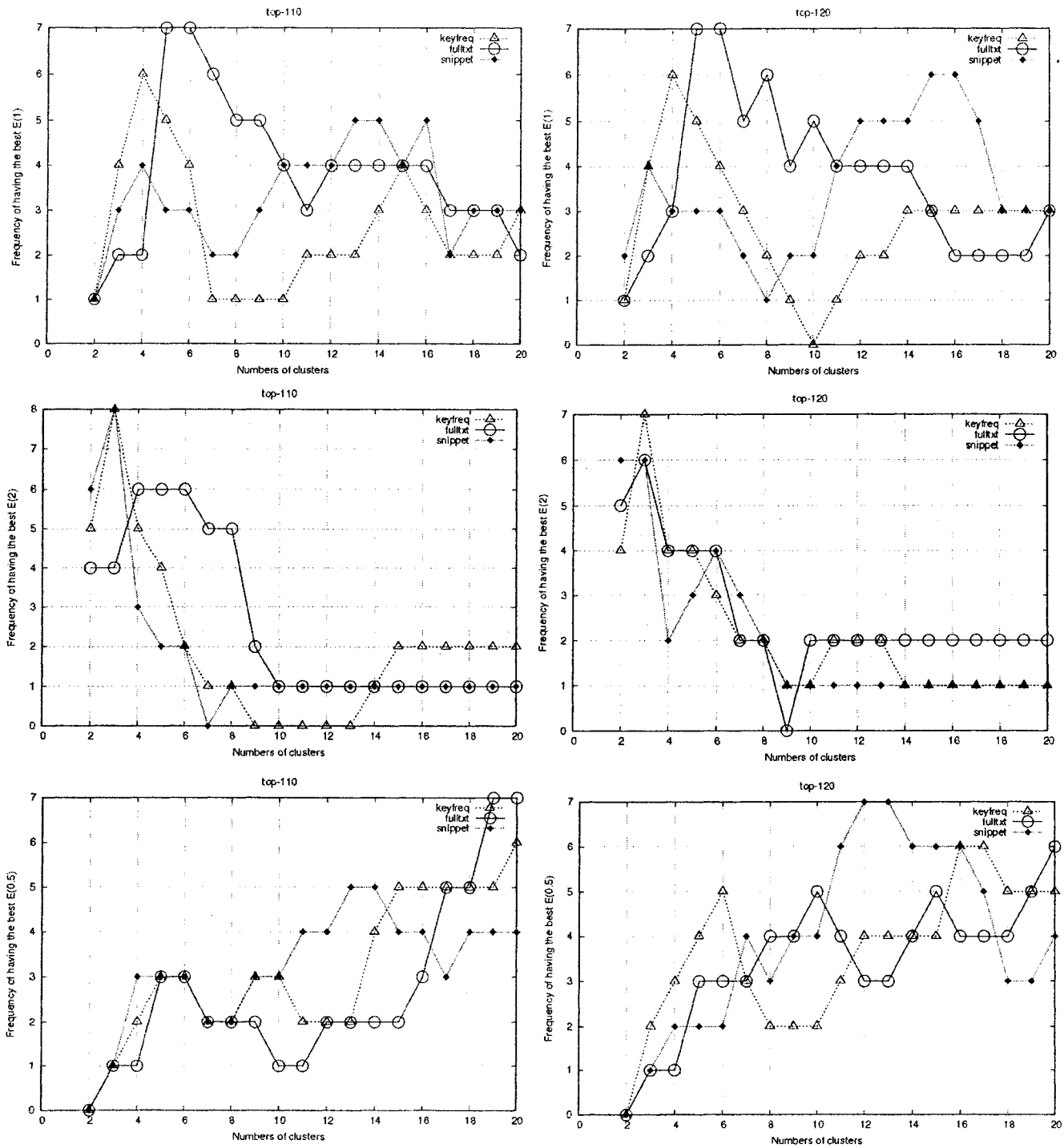

Figure E.4: Frequency of having the best E value.

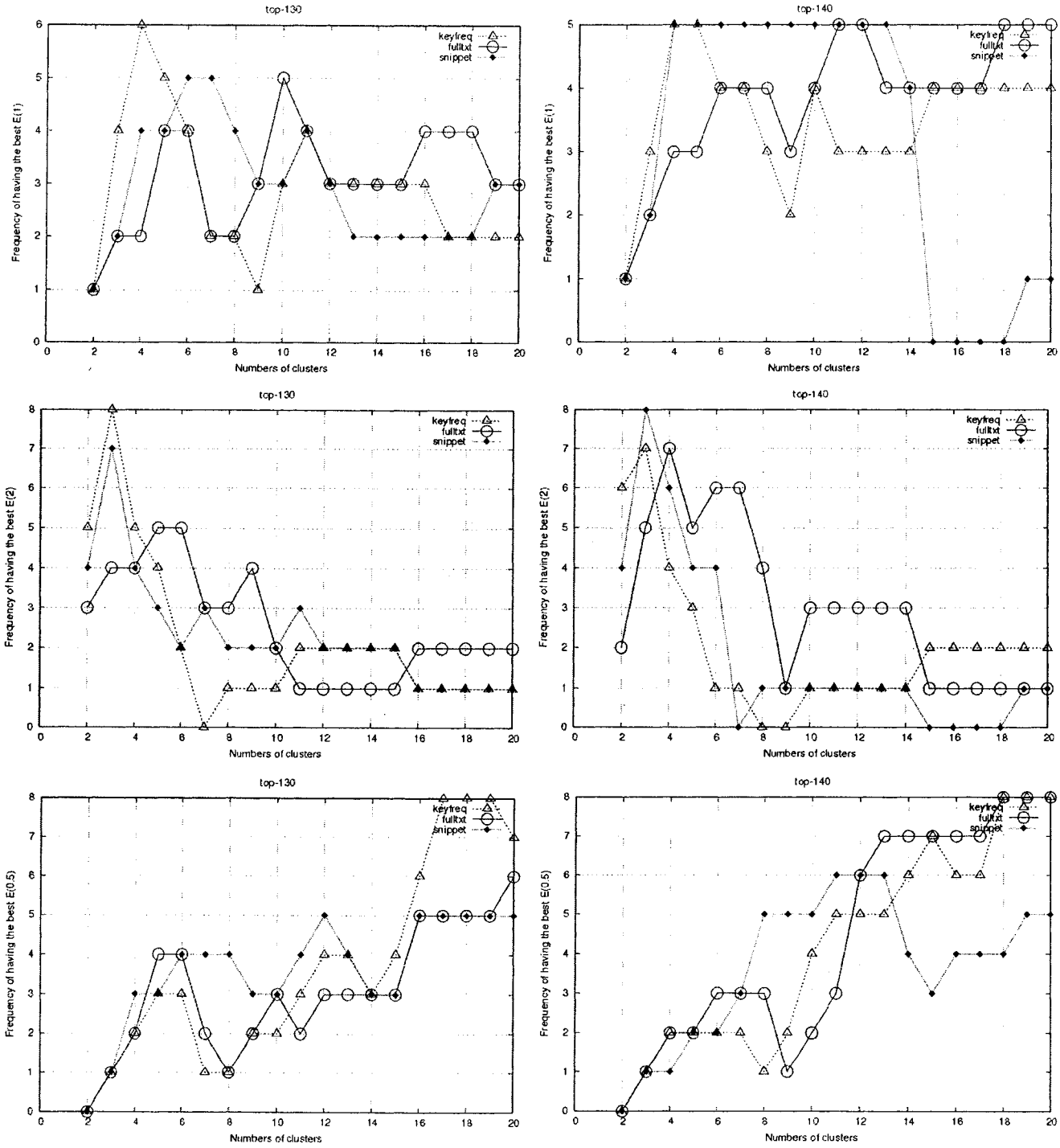

Figure E.5: Frequency of having the best E value.

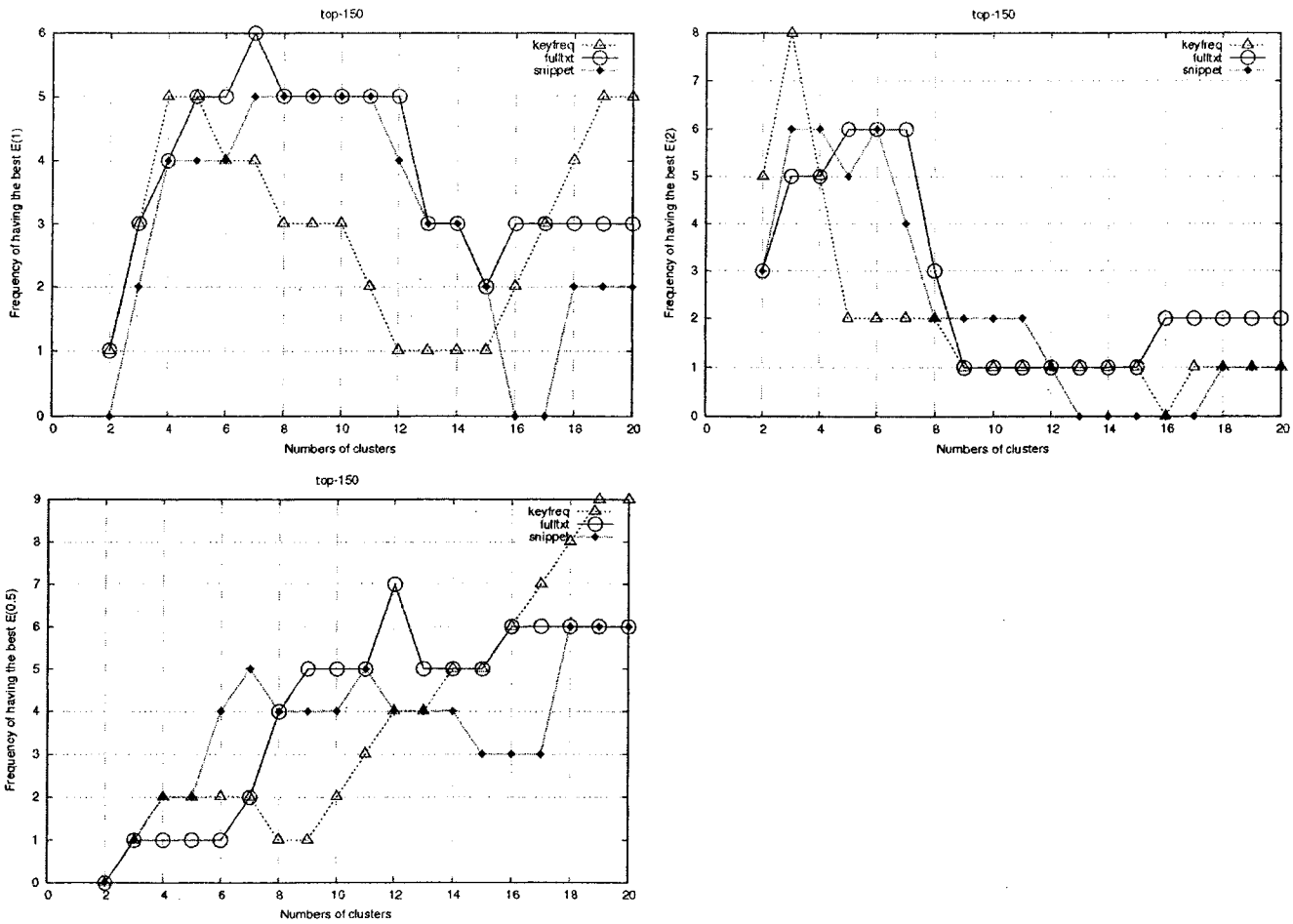

Figure E.6: Frequency of having the best E value.

Bibliography

[1] Griffiths A., Luckhurst H.C., and P. Willett. Using inter-document similarity information in document retrieval systems. *Journal of the American Society of Information Science*, 37:3–11, 1986.

[2] K. Aas and L. Eikvil. Text categorisation: A survey. June 1999.

[3] R. Allen, P. Obry, and M. Littman. An interface for navigating clustered document sets returned by queries. In *the ACM Conference on Organizational Computing Systems*, pages 166–71, 1993.

[4] Mandar Mitra Chris Buckley, Amit Singhal and Gerard Salton. New retrieval approaches using smart: Trec 4. In *The Fourth Text REtrieval Conference(TREC-4)*, pages 25–48, Gaithersburg, Maryland, 1996. NIST Special Publication 500-236.

[5] R. Cooley, J. Srivastava, and B. Mobasher. Web mining: Information and pattern discovery on the world wide web. In *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997.

[6] R.M. Cormack. A review of classification. *Journal of the Royal Statistical Society*, Series A(134):321–353, 1971.

[7] W. B. Croft. Experiments with representations in a document retrieval system. *Information Technology: Research and Development*, 2(1):1–21, 1983.

[8] W.B. Croft. *Organizing and searching large files of document descriptions. Ph.D. Thesis.* Churchill College, University of Cambridge, 1978.

[9] Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[10] Richard C. Dubes and Anil K. Jain. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[11] A. El-Hamdouchi and P. Willet. Comparison of hierarchic agglomerative clustering methods for document retrieval. In *The Computer Journal*, volume 32, pages 220–227, 1989.

[12] Curt Franklin. How internet search engines work. *http://computer.howstuffworks.com/search-engine.htm/printable.*

[13] Matthew Gray. Web growth summary. *http://www.mit.edu/people/mkgray/net/web-growth-summary.html*, 1996.

[14] A. Griffiths, L.A. Robinson, and P. Willett. Hierarchic agglomerative clustering methods for automatic document classification. *Journal of Documentation*, 40:175–205, 1984.

[15] Brad Grimes. Fooling google. *PC Magazine*, May 2003.

[16] Chris Hearne. How can information retrieval be evaluated. *http://www.scism.sbu.ac.uk/inmandw/ir/irtopics/h2.doc.*

[17] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 76–84, Zürich, CH, 1996.

[18] http://searchenginewatch.com/facts/article.php/2156041. What people search for - most popular keywords.

[19] http://searchenginewatch.com/reports/article.php/2156451. Nielsen netratings search engine ratings.

[20] http://searchenginewatch.com/sereport/article.php/2163821. Meta search or meta ads?

[21] http://search.msn.com/. Msn search.

[22] http://trec.nist.gov/. Text retrieval conference (trec) home page.

[23] http://trec.nist.gov/data/web_topics.html. Text retrieval conference web topics.

[24] http://www.ask.com/. Ask jeeves.

[25] http://www.cnn.com/2000/TECH/computing/07/26/deepweb.ap/. Study says web is 500 times larger than major search engines now show, 2000.

[26] http://www.dogpile.com/.

[27] http://www.google.com/. Google.

[28] http://www.google.com/apis/. Google web apis.

[29] http://www.google.com/help/interpret.html. How to interpret your search results.

[30] http://www.kartoo.com/.

[31] http://www.metacrawler.com/.

[32] http://www.searchenginewatch.com/. Search engine watch.

[33] http://www.surfwax.com/.

[34] http://www.vivisimo.com/.

[35] http://www.yahoo.com/. Yahoo! web directory.

[36] http://yahooligans.yahoo.com/.

[37] Wen-Chen Hu, Yining Chen, Mark S.Schmalz, and Gerhard X. Ritter. An overview of the world wide web search technologies. In *Proceedings of 5 th World Multi-conference on System, Cybernetics and Informatics*, Orlando, Florida, July 2001.

[38] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36(2):207–227, 2000.

[39] N. Jardine and C. J. van Rijsbergen. The use of hierarchical clustering in information retrieval. In *Information Storage and Retreival*, volume 7, pages 217–240, 1971.

[40] Petr Koubsky. 1.5 million pages added to web each day, says research company. *http://www.internetnews.com/bus-news/article.php/37891*, 1998.

[41] Yasemin Kural, Steve Robertson, and Susan Jones. Deciphering cluster representations. In *Information Processing and Management*, volume 37, pages 593 – 601, 2001.

[42] Yasemin Kural and Susan Jones Steve Robertson. Clustering information retrieval search outputs. In *21st BCS IRSG Colloquium on IR*. Glasgow, 1999.

[43] Sofus A. Macskassy, Arunava Banerjee, Brian D. Davison, and Haym Hirsh. Human performance on clustering web pages: A preliminary study. In *Knowledge Discovery and Data Mining*, pages 264–268, 1998.

[44] Joe D. Martin and Robert Holte. Searching for content-based addresses on the world-wide web. In *Proceedings of the 3rd ACM International Conference on Digital Libraries*, pages 299–300, Pittsburgh, PA, USA, June 1998. ACM.

[45] Fairouz Medjahed. Combining web browsing and data mining to identify interesting web sites (algeria). *ICTP Workshop on Web Enabling: Technologies & Authoring Tools*, November 1999.

[46] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to wordnet: An on-line lexical database. *Journal off Lexicograph*, 3(4):235–244, 1990.

[47] Dunja Mladenić and Marko Grobelnik. Word sequences as features in text-learning. In *the Seventh Electrotechnical and Computer Sc. Conference ERK'98*, Ljubljana, Slovenia: IEEE section, 1998.

[48] Greg R. Notess. Search engine statistics: Freshness showdown. *http://www.searchengineshowdown.com/stats/freshness.shtml*, 2003.

[49] Edward T. O'Neill, Brian F. Lavoie, and Rick Bennett. Trends in the evolution of the public web. *D-Lib Magazine*, 9(4), April 2002.

[50] Peter Pirolli, Patricia Schank, Marti Hearst, and Christine Diehl. Scatter/gather browsing communicates the topic structure of a very large text collection. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–220. ACM Press, 1996.

[51] M. F. Porter. An algorithm for suffix stripping. Program 14(3):130–137, July 1980.

[52] Kanagasabai Rajaraman and Hong Pan. Document clustering using 3-tuples. In *PRICAI Workshop on Text and Web Mining*, pages 88–95, 2000.

[53] Marc L. Resnick and Rebeca Lergier. Things you might not know about how real people search. *http://www.searchtools.com/analysis/how-people-search.html*, 2002.

[54] C.J.van Rijsbergen. *Information Retrieval.* Buttersworth, London, second edition, 1979.

[55] W. Rudin. *Functional Analysis.* McGraw-Hill, New York, 1973.

[56] G. Salton. *Automatic Information Organization and Retrieval.* McGraw-Hill, New York, 1968.

[57] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, 1989.

[58] Hinrich Schutze and Jan O. Pedersen. A co-occurrence based thesaurus and two applications to informaiton retrieval. *Information Processing & Management*, 33(3):307–318, 1997.

[59] Chris Sherman and Gary Price. *The Invisible Web: Uncovering Information Sources Search Engines Can't See.* Independent Publishers Group, 2001.

[60] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD Workshop on Text Mining*, 2000.

[61] Anastasios Tombros, Robert Villa, and C.J. Van Rijsbergen. The effectiveness of query-specific hierarchic clustering in information retrieval. *Information Processing & Management*, 38(4):559–582, July 2002.

[62] Peter D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

[63] C.J. Van Rijsbergen. Further experiments with hiearchic clustering: An evaluation of some experiments with the cranfield 1400 collection. In *Information Processing & Management*, 11, pages 171–182, 1975.

[64] P. Willett. Query specific automatic document classification. *International Forum on Information and Documentation*, 10(2):28–32, 1985.

[65] P. Willett. Recent trends in hierarchic document clustering: A critical review. *Information Processing & Management*, 24(5):577–597, 1988.

[66] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. KEA: Practical automatic keyphrase extraction. In *ACM DL*, pages 254–255, 1999.

[67] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. pages 46–54, 1998.

[68] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. Fast and intuitive clustering of web documents. In *Knowledge Discovery and Data Mining*, pages 287–290, 1997.

[69] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001.

[70] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *the eleventh international conference on Information and knowledge management*, pages 515–524. ACM Press, 2002.