Anchor Point Indexing in Web Document Retrieval*

Ben Kao Joseph Lee Chi-Yuen Ng David Cheung

Department of Computer Science and Information Systems,
The University of Hong Kong, Hong Kong.
{*kao, jkwlee, cyng, dcheung*}*@csis.hku.hk*

Abstract

Traditional WWW search engines, such as *Alta Vista*, index and recommend individual Web pages to assist users in locating relevant documents. As the Web grows, however, the number of matching pages increases at a tremendous rate. Users are often overwhelmed by the large answer set recommended by the search engines. Also, if a matching document is a hypertext, the document structure is destroyed and the individual pages that compose the document are returned instead. The logical starting point of the hyper-document is thus hidden among the large basket of matching pages. Users need to spend a lot of effort browsing through the pages to locate the starting point, a very time consuming process. This paper studies the *anchor point indexing problem*. The set of anchor points of a given user query is a *small* set of *key* pages from which the larger set of documents that are relevant to the query can be easily reached. The use of anchor points help solve the problems of huge answer set and low precision suffered by most search engines by considering the hyper-link structures of the relevant documents, and by providing a summary view of the result set.

Keywords: World-Wide-Web, hypertext, information retrieval, anchor points, indexing.

1 Introduction

The "Information Superhighway" (Internet) enables a computer user to be connected to virtually endless numbers of sites on the network. The World-Wide-Web (WWW) uses the Internet to transmit hypermedia documents between computer users located around the world. Large amounts of interesting and valuable information has been made available on the Web for retrieval. In order to fully utilize the power of the WWW as a gigantic information source, it is essential to develop intelligent software systems on top of the Web to assist users in retrieving relevant documents.

Search engines are the most popular tools that people use to locate information on the Web. A search engine works by traversing the Web via the hyper-links that connect the Web pages, performing text analysis on the pages it has encountered, and indexing the pages based on the keywords they contain [18]. A user seeking information from the Web would formulate his information goal in terms of a few keywords composing a query. A search engine, on receiving a query, would match the query against its document index. All of the pages that match the user

*A summary of the first four sections of this paper appears in the proceedings of COMPSAC '98 [19].

query will be selected into an *answer set* and be ranked according to how *relevant* the pages are with respect to the query. Relevancy here is usually based on the number of matching keywords that a page contains, with the "positions" of the matches taken into account. (For example, a matching keyword that appears in the title of a page is considered more relevant than an occurrence in the text body.)

Although search engines have been proven in practical use as indispensable tools for Web information retrieval, they suffer from a number of drawbacks [7, 11]. The simple keyword matching criteria very often do not provide queries with the expressive power to distinguish the target documents from the millions of pages available on the Web. This results in large answer sets and thus low precision. Users are often overloaded by the myriad of pages returned, seriously weakening the usability and the effectiveness of the search engines. Also, if the target document is a hypertext that consists of a number of Web pages being connected by hyper-links, the document structure is destroyed. Individual pages that compose the document are scrambled and are returned out of order. The logical starting page of the hyper-document is thus hidden among the large basket of matching pages. The purpose of this paper is to study the low precision problem of traditional search engines and suggest solutions to relieve users from being overloaded by the large numbers of recommended pages. In particular, we introduce the concept of *anchor points* which consider the hyper-link structures of the Web pages when processing queries to achieve the following goals:

- to improve the ranking of matching documents,
- to identify good starting points (anchors) that allow efficient and orderly accesses to pages that belong to the same logical hypertext documents, and
- to reduce the size of the answer set by recommending only those pages that are representative of the ones which match the query and which belong to the same logical clusters.

The rest of this paper is organized as follows. In Section 2 we discuss some common deficiencies of traditional search engines. In particular, we study the large-size, low-precision answer set problem, and the aftermath of disregarding the hyper-link structures of documents when making a recommendation. In Section 3 we mention some related works. In Section 4 we propose the idea of recommending *anchor points* instead of individual matching pages. Loosely speaking, given a query, the anchor points constitute a *small set* of *key pages* from which the set of matching pages can be accessed easily and in a logical order. The idea is that by presenting to the users a restricted set of "representative" pages, users are able to perform a fast first-level screening of the pages to single out a selected set of good candidates before examining each one of them. Also, if some of the matching pages belong to the same logical hypertext document, the anchor point that connects to them provide a good logical starting point for browsing. We propose a model for matching a query to a set of anchor points. To demonstrate the effectiveness of our approach, we implemented a prototype system for indexing anchor points and evaluated its performance through experiments. In Section 5 we describe our experiments and show the results. In Section 6 we discuss the various design and implementation issues of our prototype system. Finally, we conclude our study in Section 7.

Goal	Query	Search Engine	no. of hits	no. of relevant pages in the first 30 hits	Rank of 1st relevant hit	no. of logical clusters in the first 30 hits
Find the site of Microsoft Windows	microsoft windows	Alta Vista	1,675,241	3	10th	4
Find RIFF specification	riff specification	Lycos	unknown	1	12th	26
Find NBA scoreboard	nba score	Excite	71,118	27	3rd	2
Find general information about cricket	cricket	Infoseek	40,990	1	22nd	20

Table 1: Example queries and results.

2 Shortcomings of Search Engines

In this section we identify and discuss four sources of ineffectiveness of traditional search engines. These include:

- large answer set,
- low precision,
- unable to preserve the hypertext structures of matching hyper-documents, and
- ineffective for general-concept queries.

Large answer set. Most search engines boast of their services mentioning their extensive coverage of the Web. *Alta Vista*, for example, maintains a 200-plus gigabytes of index data covering more than 50 million sites. Empowered by high performance workstations, large storage servers, and sophisticated indexing techniques, these search engines handle user queries with reasonable response times. The quality of the responses, however, are sometimes questionable. The large numbers of pages indexed plus relatively loose matching criteria often result in large answer sets (sets of matching pages). Users are overloaded with the vast amounts of information returned from the search engines. Even though search engines rank the pages in the answer set by guessing how relevant they are with respect to the queries, the ranking systems are far from perfect given the limited expressive power of the keyword-based query interfaces. Browsing through the numerous pages returned is a tiring and time consuming process. Users do not usually have the patience to toil through more than the first thirty hits returned by a search engine. Table 1 illustrates this problem by showing the results obtained from querying four popular search engines with some sample queries.

In each row of the table, we show a search goal (*Goal*), the keyword-based query submitted (*Query*), the search engine that processed the query (*Search Engine*), the number of pages returned (*# of hits*), the number of *relevant* pages that satisfied our goal in the first thirty hits

(*# of relevant pages in the first 30 hits*)[1], the rank of the highest ranked relevant page (*Rank of 1st relevant hit*), and the number of logical hyper-text documents that contained the first thirty hits (*# of logical clusters in the first 30 hits*). The numbers shown in the last column need some further explanation. We observed that it is not unusual that a number of the pages returned are in fact parts of a logical cluster or of a hypertext document. For example, if one submits the query "nba scoreboard november 19 1997" to *Alta Vista*, one would get a whole bunch of matching pages from a sports site, one for each basketball game that occurred on a November day in some year. (The numbers 19 and 1997 are logically ignored by the engines.) One would find that all these pages share the same prefix in their URL's, and that they belong to a single logical cluster of a big hyper-document. The last column of Table 1 refers to the number of clusters that the first thirty hits can be grouped into.

From the table we see that the answer sets are huge. If it takes a person 5 seconds to decide whether a page is relevant or not, screening through 100,000 recommendations takes about 6 man-days. Of course, one would argue that the screening would stop as soon as one good recommendation is found. Still, as suggested by Table 1, the first relevant page may not be found until a couple dozens pages have been examined, many more if one is unlucky. Also, the first relevant page may not be the best page that can be found in the answer set. More screening is required if one would like to compare relevant hits looking for a better match.

Besides illustrating the large answer set problem, Table 1 also gives us a hint on how to avoid overwhelming the users with the numerous recommendations. The last column of the table suggests that the large number of pages can be grouped into a small number of logical clusters. Now, if a search engine could be smart enough to identify the clusters and recommend to the users only one *representative* page per cluster, the users would be able to screen through the suggested list much more efficiently. As an example, any one of the pages that is returned due to the query "nba scoreboard november 19 1997" would lead the user to the sports site's Web structure containing all the game summaries. If these are not the pages the user is looking for, all of them could be skipped by one inspection of a single page.

Interestingly, a "representative page" needs not be one that is highly ranked by a traditional search engine. For example, one could imagine that the game summary pages can all be reached by an *index* page. If the target of the user's goal is some of the game summary pages (e.g., all those games played in November), the index page might as well be the best starting point (an anchor) that the user can have. However, the index page may not contain the keywords with high enough occurrence frequencies for it to score high under the ranking systems of traditional search engines (e.g., the keyword "November" may occur only once in the index page, among others). Ranking anchor points is thus an interesting and challenging problem. We will discuss how anchor points could be found given a user query in Section 4.

[1] A page which simply contains the keywords as specified in the query but does not satisfy the goal is not considered a relevant hit.

Low precision. In addition to "information overload", low precision of the answer sets is sometimes another concern of the effectiveness of search engines. Previous studies have conducted experiments showing that relevant pages are often interspersed with irrelevant ones in the ranked query outputs [12]. The implication is that users cannot afford to examine only the first few, or any small subset, of the answer set. Table 1 illustrates this problem by showing the number of pages among the first 30 hits that are relevant to a search goal. We see that, for some queries, the numbers are less than honorable.

The problem of low precision has been documented in a number of previous studies [7, 11]. We will mention some of the references in Section 3. While there are many factors that lead to low precision results, we remark that under the simple model of matching Web pages against user queries based solely on word statistics, the current approach taken by most search engines might already be representing the best effort. We believe that working on a better user interface that assists the users to better express their search goals should be a more fruitful option in precision improvement. One example system developed by The University of Arizona using the idea of concept space [4, 5] has demonstrated that user queries can be semi-automatically enhanced to improve the precision of the answer sets. (*Excite* also takes a similar approach.)

In this paper we do not attempt to *solve* the low precision problem. However, we identify and study one important source of the problem and discuss how the concept of anchor points could help tackle it.

The biggest attractive feature of the Web is that it provides an on-line source of information being structured as a huge network of Web pages, each contains a certain piece of interesting knowledge. Users are free to access and navigate the Web via hyper-links connecting related information units. Hypertext is especially useful for "on-line presentation of large amounts of loosely structured information such as on-line documentation or computer-aided learning" [16]. It is therefore a useful concept that Web document authors like to adopt. As an example, the *Usenet Hypertext Frequently Asked Questions Archive* stores FAQs of different subjects in hierarchical hypertext structures. A FAQ is usually broken up into a number of parts, each part containing a number of questions and their answers. Figure 1 shows a logical hypertext structure of a FAQ document.

In the figure, the FAQ is rooted at the part index page, which contains hyper-links to three second-level index pages: part I, part II, and part III. These index pages, in turn, point to a number of "leaf" pages containing information about certain subjects (A, B, and C in the figure). Besides the top-down hyper-links (solid arrows), there are cross references (dashed arrows) among the pages.

To the author of the FAQ, the whole structure constitutes a single document, and the part index represents the logical starting point of the document. Unfortunately, traditional search engines ignore the structural information embedded in the hyper-links and index the pages separately. If a user queries an engine with the keywords A, B, and C, none of the pages matches all the keywords. These pages would be ranked very lowly by the engines even though

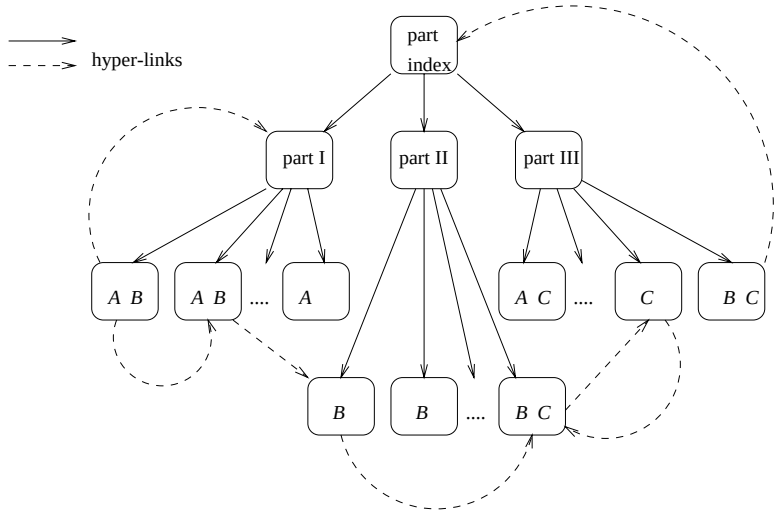


Figure 1: A typical hypertext structure of a FAQ document.

the hyper-document as a whole might provide all one needs to know about the subjects. On the other hand, if the author puts everything in one single Web page, the document would match the user query very well. This reveals a drawback of the ranking mechanism of traditional search engines, namely, it works against documents that are presented in hypertext structures.

To deal with this ranking misjudgment, a search engine should recognize the fact that logically the pages belong to a hyper-document (or a cluster) and that the document is relevant to the subjects that occur in the pages (A, B, C in our example). That is to say, the search engine should be able to (1) match queries against hypertext structures and (2) return an *entry point* through which the constituent pages can be easily accessed. As we have alluded to earlier, an anchor point is a representative page of a cluster of pages that provides a good starting point and that allows efficient and orderly accesses to pages that belong to the same logical cluster, the idea of anchor points meets our second requirement. To satisfy the first requirement, we need to associate a page with not only the keywords it contains, but also the keywords that occur in the cluster the page belongs to. (With our FAQ example, the search engine should associate the keywords A, B, C with the part index page because those keywords occur in the cluster and the pages containing them can be easily accessed through the part index.) In Section 4, we will describe how this type of keyword associations are derived and how the associations are used to efficiently locate anchor points given a user query.

Destroying the hypertext structures of matching hyper-documents. Another inadequacy of traditional search engines again results from not preserving the hypertext structures of matching hyper-documents when making recommendations. Even if an engine could match a query with the pages of a hyper-document, the hypertext structure is flattened and the individual pages are returned out of order. As an example, we submitted the query "C80 faq" to *Alta Vista* looking for information about a DSP chip named C80. *Alta Vista* successfully located a forum with hundreds of postings about the chip. The postings were organized as a hypertext with an index page pointing to the numerous postings, ordered by date. Each posting was

contained in a separate Web page. Although the pages returned by the search engine matched the query goal, the postings listed in the answer set were totally out of order, and the index page was not listed in the first 100 hits returned. A user thus needs to reconstruct the logical reading path from the disorganized answer set. A better approach would be to recognize that the pages belong to the same cluster and to return a logical starting point (such as the index page). The concept of anchor point again applies here. We will demonstrate how anchor points help ameliorate the problem in Section 5.

General concept queries. Finally, while traditional search engines perform quite well for "specific" and "precise" queries (e.g., "find me the solution for solving the Rubik's cube), they are not particularly effective in serving general concept queries. As an example, someone may want to know about the sport cricket. Ideally, a site such as *CricInfo* that is dedicated to the sport would be a perfect match. Unfortunately, the query "cricket" is too general for traditional search engines with their simple keyword-matching systems. The result is that any Web page that contains the keyword "cricket" matches the query, be it the start page of *CricInfo*, a page written by a fifth grader on his favorite sports, or a page reporting the scores of an international cricket tournament.

As the Web develops, we see more and more information sources that are dedicated to specific topics of interests. There are sites for *tennis*, sites for *salmon*, and even sites for *Big Foot*. If a user is looking for some general information about a topic, chances are that a site exists on the Web that is specialized on that topic. Recommending Web sites instead of individual Web pages becomes more meaningful to this type of general concept queries.

Currently, the best way to look for specialized information sources is to use a directory service, such as *Yahoo!*. The down sides of this approach are that subject categorization is done by hand and that the sites need to be suggested.

As an alternative, one could imagine that the Web pages of a specialized site circle around a major subject, and thus could be considered as parts of a very big cluster or a hypertext document rooted at the site's home page. Also, as we have discussed, an anchor point is a representative page of a cluster and that it is associated with the keywords found in the cluster. It is reasonable to argue that a specialized site's home page is a good anchor point that matches the general concept query (on the site's specialized subject). The concept of anchor points is thus useful in recommending Web sites and answering general concept queries.

3 Related Works

The goal of our study is to improve the effectiveness of information retrieval on the WWW. In particular, we focus on improving the ranking system of search engines dealing with hypertext documents, cutting down on the size of answer sets, supporting general concept queries, and identifying good starting points for efficient and orderly accesses to hypertext documents. The

inadequacies of traditional search engines with respect to some of the above issues are mentioned in [7, 11].

Some research studies take another approach to matching users to information on the Web. Instead of indexing the whole Web like traditional search engines do (a *server-based* approach), these studies work on the design of intelligent *Client-based* Web tools that "learn" about a user's interests and guide the user in traversing the Web, zeroing on the target documents. Here, we briefly mention two such systems. Interested readers are referred to [6, 3, 1, 15, 2, 14] for more details.

De Bra and Post developed a client-based search engine. The system allows a user to specify a search goal in three different ways: (1) by keywords — documents are ranked according to the occurrences of the user-supplied keywords; (2) by regular expression — documents are ranked according to the number of matches of the user-supplied patterns; and (3) by external filters — a user can provide a program that assigns scores to documents. The search engine explores the Web using a *fish search algorithm* with relevant documents ranked according to the scoring system the user specified.

Letizia [15] is an intelligent agent that works with a conventional Web browser. It tracks the user's browsing behavior and tries to infer the user's goals. While the user is reading a page, Letizia conducts a resource-limited search based on the goals it deduced. Relevant pages found during the search will be recommended to the user upon request. The goal of Letizia is to automatically perform some of the Web exploration on behalf of the user to anticipate future page accesses.

In [14], an intelligent system is designed which tracks users' browsing behavior to deduce sets of keywords (called term vectors). These term vectors are used to describe the information that the users are interested. The paper proposes an architecture of an intelligent system that integrates various tools to analyze the users' accessing behavior and automatically brings in relevant documents for the users. The system consists of two learning agents, one for discovering users' topics of interest, and another for discovering the topics covered by information sources. The system then matches users to Web sites based on the topics accordingly.

The expressive power of conventional search engine query interfaces is relatively weak when restricted to keyword-based search. The deficiencies are documented in [7]. One of the many problems is that a concept under interest may be described by many terms. For example, someone looking for information about "AIDS" would miss documents that mention only "HIV". One approach to this problem is to construct a *concept space graph* representing the important terms and their weighted relationships. This concept space graph is then used to augment keyword queries by complementing user supplied keywords with their strongly associated terms [4]. Concept space construction involves *frequency analysis* and *cluster analysis* [5]. However, these analyses are very computationally expensive and thus are not suitable for a highly dynamic Web environment.

4 Anchor Points

In Section 2 we explained how recommending anchor points can improve the effectiveness of search engines, particularly in dealing with hypertext documents. To reiterate, given a user query and subsequently a set of matching Web pages which form a number of clusters, we propose that the system should recommend, instead of all the matching pages, a set of anchors (possibly one for each cluster) such that

1. an anchor is a representative page of a cluster (i.e., by inspecting the anchor, a user can easily deduce what the pages in the corresponding cluster are about); and
2. an anchor provides efficient and orderly accesses to the pages in the corresponding cluster.

In our discussion so far, we have only presented the idea of anchor points fairly informally. For example, we have not defined what we mean by a cluster, what we mean by a page being a representative of a cluster, or what we mean by efficient accesses to a cluster of pages. The reason is that there are, in fact, many different ways one could interpret the terms. (For example, efficient accesses of pages may be defined in terms of the number of network requests, the number of mouse clicks, or the amount of memory cache needed.) In order to avoid restricting ourselves to one interpretation, we focused our previous discussion on the general concept and the motivation of the problem.

In this section we present one formal definition of anchor points. We show how to process queries in anchor point recommendation. We will demonstrate the effectiveness of our approach in solving the commonly occurring problems in traditional search engines (Section 2) via experiments in Section 5.

4.1 Definitions

We define the *distance* from a Web page A to a Web page B, denoted by $D(A, B)$, to be the minimum number of hyper-links that need to be traversed to reach page B starting from page A. If page B is unreachable from A, we have $D(A, B) = \infty$. We define the *k-neighborhood* of a Web page A, denoted by $N_k(A)$, to be the set of all the pages that are within distance k or less from A. Figure 2 illustrates these terms. For example, the distance from page A to page J is 3 ($A \rightarrow E \rightarrow D \rightarrow J$) while the distance from page J to A is 4 ($J \rightarrow E \rightarrow D \rightarrow C \rightarrow A$). The 2-neighborhood of page A is the set of all the pages encircled by the dotted lines. With our notations, we have $D(A, J) = 3$, $D(J, A) = 4$, and $N_2(A) = \{A, B, C, D, E, F, G, H\}$.

We assume a *scoring function* f exists such that given a keyword a and a Web page X, $f(X, a)$ measures the extent that page X matches the keyword a. We call the value of this function the *score* of page X with respect to a. There are many choices for such a function. One simple example would be:

$$f(X, a) = \begin{cases} 0 & \text{if } X \text{ does not contain } a \\ 1 & \text{if } X \text{ contains } a \end{cases}$$

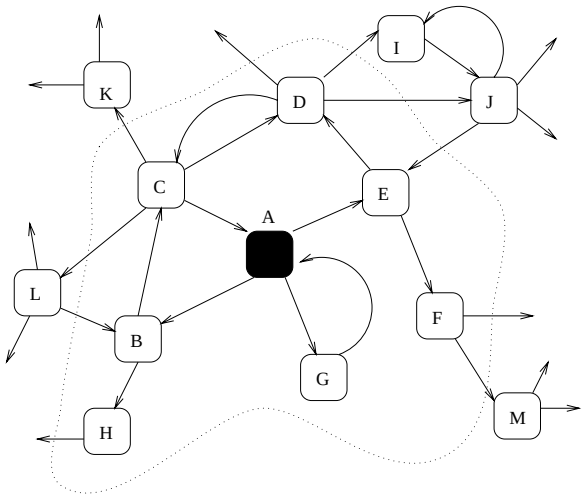


Figure 2: Distance and neighborhood.

Alternatively, $f(X, a)$ could be the normalized occurrence frequency of a in X, as is done in *TFIDF* [13]. A third example would be the probabilistic model as suggested in [17] in which the authors proposed a function that estimates the *probability* that a document X is relevant to a keyword a.

Recall that our goal is to recommend a page to a user starting from which he is likely to find relevant pages via *simple navigation*. Here, let us assume that by simple navigation, we mean that the user does not need to traverse more than k links away from the suggested starting point. In order to measure how well potentially a page X can lead a user to pages that match a keyword a, we define a *potential function* $P_k(X, a)$ as follows:

$$P_k(X, a) = \sum_{Y \in N_k(X)} f(Y, a) \times \alpha^{D(X,Y)} \tag{1}$$

where α is a constant parameter between 0 and 1.[2] In words, the potential function of a page X with respect to a keyword a gives the sum of all the scores of the pages in X's k-neighborhood, with each page's contribution to the score scaled down exponentially with respect to its distance from page X. The constant α controls the pace of the scale-down (the smaller the value, the faster the pace).

Depending on the semantics of the function f, different quantitative interpretations can be associated with the potential function. For example, if $f(Y, a)$ represents the *amount* of information that a page Y contains about the keyword a, and α is the probability that a user follows a hyper-link, then $\alpha^{D(X,Y)}$ gives the probability that page Y is visited if a user starts at page X, and $P_k(X, a)$ gives the *expected* amount of information that a user would learn about keyword a if he starts from page X (assuming that the user does not take more than k hops away from X). As another example, if $f(Y, a)$ measures the *probability* that page Y is relevant to keyword a, then $f(Y, a)\alpha^{D(X,Y)}$ is the probability that a user starting from page X would

[2] We take $0^0 = 1$.

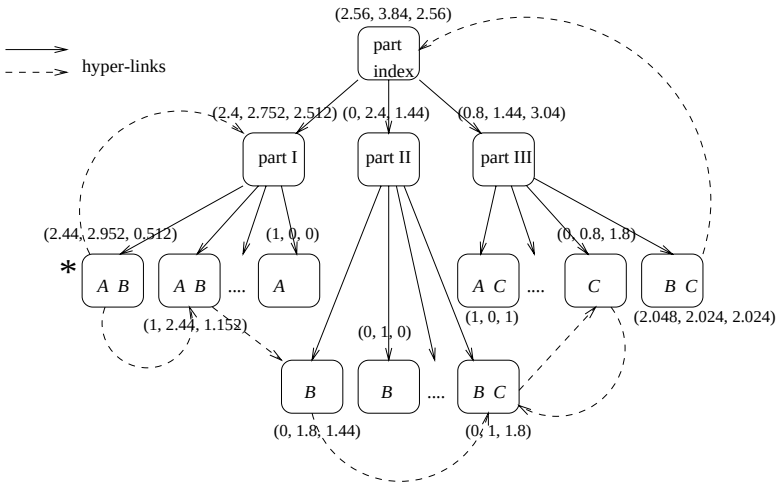


Figure 3: A FAQ document and the potential of each individual page.

get a relevant page in Y about the keyword a. Hence, $P_k(X, a)$ is equal to the *expected* number of pages that are relevant to a that a user would visit given that he starts at page X. In any case, intuitively, $P_k(X, a)$ is an indicator of how much information about a one can get starting from page X. For the purpose of discussion, we will use the second interpretation of f (i.e., $f(Y, a)$ measures the probability that page Y is relevant to a) for the rest of this section.

4.2 Queries

Given a query Q and a page X, our approach to evaluate whether X is a good anchor point for Q is by estimating the number of relevant documents (w.r.t. Q) that are within distance k from X and that a user will visit if he starts from X. We call this estimate the *potential* of X with respect to Q (Potential(X, Q)). Anchor points are ranked based on their potentials. In this subsection we show how anchor point potentials are computed.

As we have discussed in Section 4.1, if $f(Y, a)$ is a measure of the probability that page Y is relevant to a keyword a, then $P_k(X, a)$ is the expected number of relevant pages (w.r.t. a) in X's k-neighborhood that a user would visit if he starts from X. So, for a single-keyword query $Q = a$, Potential(X, Q) is simply $P_k(X, a)$, as is given in Equation 1.

As an example, the FAQ document we discussed in Section 2 is redrawn in Figure 3. If $\alpha = 0.8$, $k = 3$, and the scoring function $f(X, a)$ simply returns 1 if X contains a; 0 otherwise, then the tuple beside each page in the figure shows the values of Potential with respect to the keywords A, B, C respectively. For example, we have

$$\begin{aligned} Potential(\text{“part I”}, A) &= P_k(\text{“part I”}, A) = 2.4, \\ Potential(\text{“part I”}, B) &= P_k(\text{“part I”}, B) = 2.752, \\ Potential(\text{“part I”}, C) &= P_k(\text{“part I”}, C) = 0.512. \end{aligned}$$

In this example, the anchor points for keywords A, B, and C are “part index”, “part index”,

and "part III" respectively.

For a multiple-keyword query Q, Potential(X, Q) can also be estimated based on $P_k(X, a_i)$ where a_i's are the keywords in Q. Here, we distinguish two cases: conjunctive queries and disjunctive queries.

To simplify our discussion, let us define $Pr[Q]$ to be that, given a page in X's k-neighborhood that a user visits if he starts from X, the probability that the page is relevant to Q. Since the expected number of pages $(n_k(X))$ in X's k-neighborhood that a user would visit if he starts from page X is simply:

$$n_k(X) = \sum_{Y \in N_k(X)} \alpha^{D(X,Y)},$$

we have $Pr[a] = P_k(X, a)/n_k(X)$ for any keyword a.

Assuming that the occurrences of keywords in a document are independent from each other, we have, given a conjunctive query $Q = a_1 \wedge a_2 \wedge ... \wedge a_m$,

$$\begin{aligned} Pr[Q] &= Pr[a_1 \wedge a_2 \wedge ... \wedge a_m] \\ &= Pr[a_1] \cdot Pr[a_2] ... \cdot Pr[a_m] \\ &= \frac{P_k(X, a_1)}{n_k(X)} \cdot \frac{P_k(X, a_2)}{n_k(X)} ... \cdot \frac{P_k(X, a_m)}{n_k(X)} \\ &= \frac{(\prod_{i=1}^{m} P_k(X, a_i))}{(n_k(X))^m}. \end{aligned}$$

Thus,

$$Potential(X, Q) = Pr[Q] \cdot n_k(X) = (\prod_{i=1}^{m} P_k(X, a_i))/(n_k(X))^{m-1}. \quad (2)$$

We remark that in practice the independent assumption may not hold. However, we would like to point out that the Potential function as defined above is used only to rank anchor points. It is not used to compute an accurate estimate of the number of relevant pages. In fact, this independent assumption is used in other information retrieval techniques, such as *Gloss* [8, 9, 10]. A *Gloss* server is one that maintains certain statistics about a number of *information sources* such as document libraries. Given a user query (conjunctive or disjunctive), the *Gloss* server estimates, based on the statistics, which information source is most likely to contain the largest number of matching documents. Basically, the server remembers, for each information source and each keyword a, the number of documents in the information source that contain a. *Gloss* uses this keyword statistics and the independent assumption to estimate the expected number of documents that each information source contains that match a query. It is shown that, in practice, such estimation is extremely effective in *ranking* information sources on how likely they contain relevant documents to queries.

The potential of a page with respect to a disjunctive query can be similarly estimated. Given a disjunctive query $Q = a_1 \vee a_2 \vee ... \vee a_m$, by the principal of inclusion and exclusion, we have,

$$Pr[Q] = Pr[a_1 \vee a_2 \vee ... \vee a_m]$$

$$= \sum_i Pr[a_i] - \sum_{i_1 \neq i_2} Pr[a_{i_1} \wedge a_{i_2}] + ... + (-1)^{m-1} \sum_{i_1 \neq i_2 \neq ... \neq i_m} Pr[a_{i_1} \wedge ... \wedge a_{i_m}]$$

$$= \sum_i \frac{P_k(X, a_i)}{n_k(X)} - \sum_{i_1 \neq i_2} \frac{P_k(X, a_{i_1})}{n_k(X)} \cdot \frac{P_k(X, a_{i_2})}{n_k(X)} + ... + (-1)^{m-1} \sum_{i_1 \neq i_2 \neq ... \neq i_m} (\prod_{j=1}^{m} \frac{P_k(X, a_{i_j})}{n_k(X)}).$$

Thus,

$$Potential(X, Q) = Pr[Q] \cdot n_k(X) \tag{3}$$

can be estimated by the $P_k(X, a_i)$'s.

As an example, if we apply the above formula to the FAQ hypertext (Figure 3), the anchor points for the queries "$A \wedge B$" and "$A \vee C$" are the page marked with an asterisk and "part index" respectively.

5 Experiments

In order to demonstrate the effectiveness of our model, we have built a prototype system and have conducted a series of experiments. We took snapshots of Web pages from four information sources. These include a Sports site, a Newspaper site, a site that contains FAQs' and a site that maintains a collection of NetNews articles. These sites were chosen because they had very different hypertext structures and contents. Most importantly, these structures are commonly found in major information sources on the Web. For example, Newspaper organizes its pages into different sections (e.g., World News, Sports, Weather, etc.) with cross-references among them. On the other hand, NetNews articles form many long lists of follow-up postings with few references between the lists. By studying the effectiveness of our model over a diversified set of information sources, we will be able to learn about the robustness of the model and how it can be improved.

Our prototype system for indexing and recommending anchor points is implemented based on the model discussed in the previous sections. The Web pages collected from the snapshots were parsed to obtain the word statistics and the hypertext structures of the information sources. Based on this information, the various data structures needed to compute anchor point potentials were constructed. These data structures include the scoring function $f(X, a)$, the k-neighborhood of each page, and the single-keyword Potential function $P_k(X, a)$. We will discuss how these data structures are maintained and constructed in Section 6.

Volunteers were invited to test the performance of the prototype system by submitting queries that covered a broad range of topics. These queries include single-keyword queries, multiple-keyword (conjunctive and disjunctive) queries and general-concept queries. For each query, the inquirer would traverse the Web pages in the snapshots manually and would locate and rank the pages that he/she would consider the best "starting pages." We then submitted the same set of queries to our prototype and compared its recommendations against the inquirer choices. The goals of these experiments are to find out:

- the *precision* of the recommendations. That is how good the recommended anchor points are in leading a user to the requested information.
- the *conciseness* of the recommendations. That is how well the recommended anchor points summarize the relevant pages.
- the appropriate values for k and α.
- the *stability* of anchor points. That is how they shift over time as the web pages in the information sources get updated.

In the rest of this section we briefly summarize some general observations from our experiments. We first show a few examples of the queries submitted to and the recommendations returned from the prototype. We then discuss how the values of k and α are chosen. Finally, we submitted the same set of queries to different snapshots (taken at different times) of the information sources. We show that anchor points are usually fairly "stable" despite frequent updates of individual Web pages.

Sports and Newspaper

The Web sites Sports and Newspaper exhibit very similar hypertext structures. Both of them organize their documents into several major sections (e.g., local news, international news, weather, finance, etc.) Each section consists of a section index page which contains headlines (with pointers to the corresponding article pages) and pointers to subsection index pages. Another common feature of both Sports and Newspaper is that most of the pages contain a menu bar with references to the section index pages. Hence, starting from almost any page in the hypertext structure, a user can reach a section index in just one hop.

We describe the hypertext structure of Sports and Newspaper as *shallow* and *broad*. We found that starting from the home page, one can reach a major portion of the pages in the information source within 4 hops. As an example, the 4-neighborhood of the Sports's home page represents 54.3% of all the pages in Sports (66.3% for Newspaper). The hypertext structure is thus very short or *shallow*. Also, most of the pages contain many hyperlinks to other pages. For example, in our experiment, the NBA index page in Sports contains 49 links to other pages. A page thus has a very extensive or *broad* reach to others.

With a shallow and broad hypertext structure, pages are fairly *close* to each other. Also, the k-neighborhoods of any two pages overlap to a large extent even if k is moderately large (e.g., when $k = 4$). Recall that a page X is considered an anchor point of a query Q if one can find much information about Q in the k-neighborhood of X and that the information is *close* to X. With a shallow and broad hypertext structure, we thus need a fine measure of k-neighborhood (i.e., a small k) and a fine measure of closeness (i.e., a small α) in order to better distinguish the pages' anchor point rankings. Therefore, in the experiment on Sports and Newspaper, we picked $k = 2$ and $\alpha = 0.2$.

Figure 4 shows an example query `<<nba & scoreboard>>` and the potential values of the

```
Query submitted: nba scoreboard

Rank    Page                                          Potential
----    -------------                                 ---------

1       NBA Index page                                 8,529.7
2       NBA Today's scoreboard                         5,587.9
3       Index page to the editorial section of NBA     3,143.8
4       Sports homepage (leading to NBA Index page)    2,512.0
5       NBA scoreboard on a particular date            1,983.4
```

Figure 4: Sports: Ranking of query result, $k = 2$, $\alpha = 0.2$.

```
Query submitted: weather

Rank    Page                                                Potential
----    -------------                                       ---------

1       US cities weather forecasts index                   1,056.9
2       today's climate in various states in the US           828.3
3       world cities (outside US) weather forecasts index     701.8
4       general weather knowledge FAQ                         691.1
5       another index page on 5-day weather forecasts         682.8
```

Figure 5: Newspaper: Ranking of query result, $k = 2$, $\alpha = 0.2$.

top 5 recommendations in Sports evaluated with respect to the query. The goal of the query is to locate information about the scores of NBA matches. We note that the highest-ranked recommendations represent very good anchor points to the query. For example, the first-ranked recommendation is the NBA section index page, from which a user can find a link directing the user to the NBA scoreboard section, a link to the highlights of today's NBA match boxscores, a link to score statistics, besides others. The second-ranked recommendation is a page containing today's scoreboard. This page also contains a calendar on which each date is associated with a link to a boxscore page of matches played on that date. The third-ranked recommendation is an index page to the NBA editorial section. This page also contains much information about NBA matches and scores, as well as a link to the scoreboard page.

As another example, Figure 5 shows the prototype's recommendations in Newspaper to the general concept query `<<weather>>`. The top five recommendations are excellent anchor points to the query. For example, the first-ranked page lists the weather forecasts to all the cities in the US. It contains links to information about the weather of different cities, links to different sections of "weather", such as world weather, statistics, etc. The 4th recommendation is also an interesting one. It contains answers to many common weather questions (e.g., storm formation, global temperature change, El Niño and La Niña, etc.)

We submitted many different types of queries in the experiment to test the effectiveness of our prototype. In most cases, we found that the suggested pages are precise and relevant to our queries. Most importantly, the recommended pages represent logical starting points from

```
Query submitted: pointer arithmetic

Rank  Page
----  -------------
1     index page of FAQs in comp.lang.c.moderated
2     index page of four C FAQs
3     index page of the directory containing C FAQs
4     subgroups and FAQs in comp.lang.c
5     another index page of the directory containing C FAQs
6     comp.lang.c answers to frequently asked questions (FAQ List)
7     comp.lang.c answers (abridged) to frequently asked questions
8     index page of FAQs under the 'Programming' area
9     index page of three Java-related FAQs
10    index page of C (Abridged) FAQs
```

Figure 6: FAQ: Another example of ranking result, $k = 3$, $\alpha = 0.8$.

which relevant information can be orderly retrieved in their k-neighborhoods. For general-concept queries, our model also works well. We submitted to the prototype many general-concept terms such as `<<tennis>>`, `<<soccer>>`, `<<music>>`, `<<computer>>`, etc. Even though there are hundreds of pages matching each query, our prototype successfully returned some important index pages as highly recommended anchor points. For example, the tennis section index page was the first-ranked anchor point for the query `<<tennis>>`. As another example, for the query `<<music>>`, the first-ranked recommendation was the music entertainment section page in which there are links to pop music highlights, links to songs and CD reviews, links to music programs, etc.

FAQ

The FAQ site indexes the FAQ documents by a tree-like index. The top-level (root) index contains links to a number of *area index pages*. Each area index page contains links to various FAQ documents on different subject areas. Many FAQ documents are split into multiple sections, in which case, the FAQ documents are themselves hypertext documents (see Figure 1).

In the experiment, most of the FAQs (other than the index pages) do not contain links to other documents (except possibly back to the section index page or to the other section pages of the same FAQ). The hypertext structure is thus *narrow*, meaning that cross-references are rare; and *shallow*, meaning that a user can reach most of the pages within a few hops from the root index. With a narrow hypertext structure, users are more willing to explore pages that are located a bit farther from a given starting page. This is because the size of the k-neighborhood of a page is small enough for a user to manage and navigate without getting lost. Hence in our experiment with FAQ, we set $k = 3$. Figure 6 shows an example query `<<pointer & arithmetic>>` and the anchor point ranking computed by our prototype.

There are three FAQ documents about the C programming language that were identified by the inquirer as the most relevant documents to the query. The first 5 recommendations by our

prototype are index pages containing the links to all these FAQs. It is interesting to note that while the user picked these index pages as good starting points for locating matching pages, these index pages themselves contain neither keywords in the query. A traditional search engine would thus exclude them in its recommendation.

In the FAQ experiment, we collected a total of 115 distinct user queries. Among them, 50 contained one keyword, 53 contained two keywords, and 12 contained three keywords. To compare our prototype's performance against that of a traditional search engine, we repeated our experiment with the same set of queries but set $k = 0$ in our prototype. This is equivalent to not considering the neighborhoods of pages in calculating potentials. That is, only the scoring function ($TFIDF$) is used in the ranking.

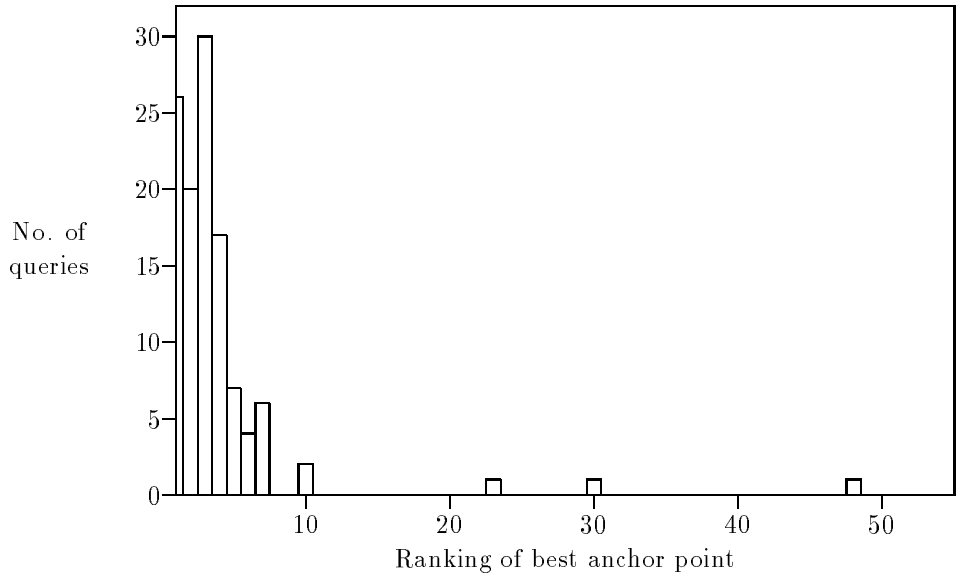
We checked how the user-identified best anchor points ranked in the two cases (i.e., when $k = 3$ and when $k = 0$). Figure 7 shows the result. Each histogram shows the number of queries whose best anchor points are recommended at a particular rank. For example, with $k = 3$, the best anchor points were ranked top in 26 queries and second in 20 queries; with $k = 0$, however, the best anchor points were ranked top in only 11 queries and second in 5 queries. From the figure, we see that with $k = 3$, almost all best anchor points were returned within the top 7 recommendations. The quality of the recommendations is thus much better when anchor point potentials ($k = 3$) are considered over the simple $TFIDF$ scores ($k = 0$).

Besides ranking the best anchor points higher, our prototype was also more successful in locating them. Figure 8 illustrates this performance advantage by displaying the cumulative percentage of queries whose best anchor points were found in the first n ranks of the result sets, where n is represented by the x-axis of the figure. We see that, for $k = 0$, the best anchor points were identified in only about half of the queries (55 out of 115). The reason is that in FAQ, an index is often created to link up a number of related FAQs. Such index page is usually very small, and contains only the name of the subject and the names of the referenced FAQs. While such an index page is a good anchor point for many queries related to the subject, it often does not contain the queries' keywords. (The query `<<pointer & arithmetic>>` is an example.)

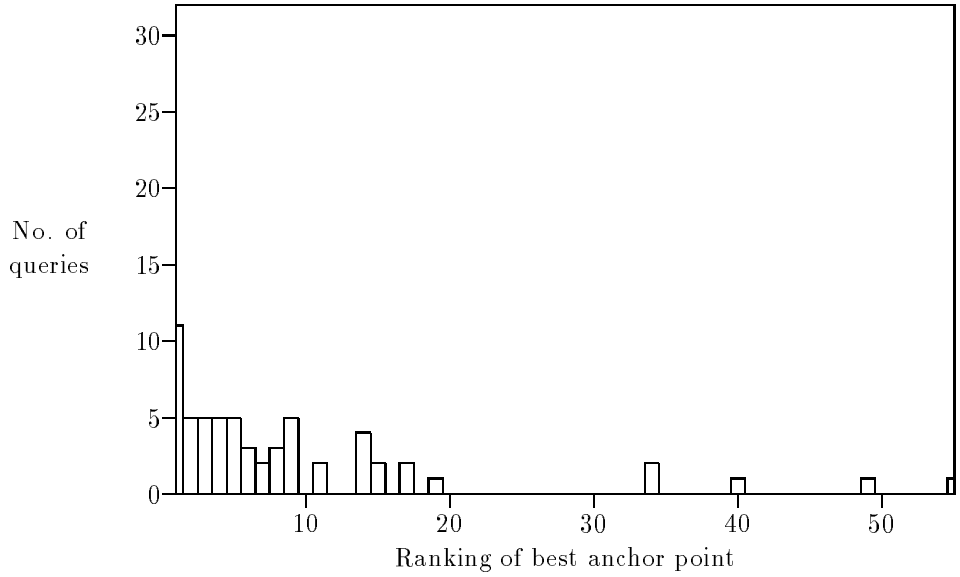
NetNews

Newsgroups are very important sources of information on the Internet. A newsgroup article can be posted as a follow-up of another, thus newsgroup articles form logical threads of discussion. Articles belonging to the same thread tend to use similar vocabulary. Given a query there could be a large number of matching articles that belong to the same thread of discussion. Recommending an article that starts a thread of discussion as a logical starting point is much better than suggesting all the relevant articles to the user.

Newsgroup articles are not hyperlinked documents. When applying the anchor point technique, the "hyperlink" relationship is replaced by the "follow-up" relationship. For example, if an article B is a follow-up of an article A, we say that B is in the 1-neighborhood of A. If an



(a) $k = 3$



(b) k=0

Figure 7: No. of queries against ranking of best anchor point. (a) $k = 3$; (b) $k = 0$.

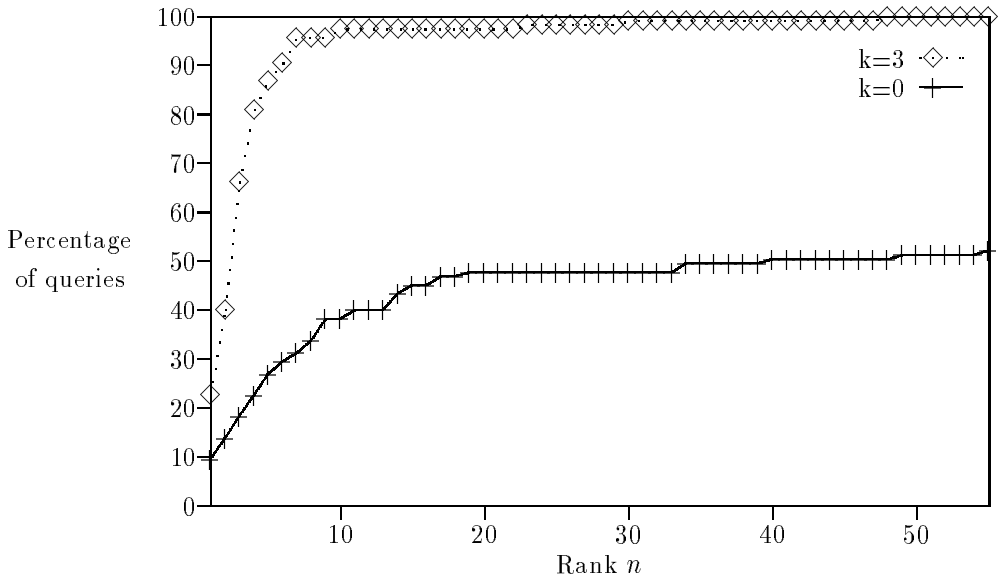


Figure 8: Cumulative percentage of queries whose best anchor points were found in the first n ranks of the result.

article C is a follow-up of B, C is in the 1-neighborhood of B and is in the 2-neighborhood of A. Suppose that they are all relevant to a query, returning only A instead of all three articles to the user gives a more concise recommendation. With a user interface that groups news articles by threads (e.g., the news reader comes with the Netscape Communicator), a user can easily find all the relevant articles within the thread. In the experiment we indexed some 4,300 articles in `alt.sports.soccer.european` over the period from June 20 to July 22, 1998. We describe the NetNews structure as *narrow* and *deep* because there are not many cross references between threads, and threads are usually long. Following our discussion on Sports and FAQ, we picked a large value for α (0.8), and set $k = 3$.

Our experiment on NetNews again showed promising results. For most queries, a relevant article that started a thread of discussion ranked higher than its follow-up postings. Taking it as the anchor point thus dramatically reduced the size of the result sets when compared with those of traditional search engines. On the other hand, the prototype did not blindly return just the *root* articles (i.e., those that are not follow-ups of any others) in answering queries. If the focus of a thread of discussion shifted, the prototype would identify the article where the shift of interest occurred.

5.1 Stability of anchor points

The Web represents a very dynamic source of information. Web pages are added and deleted constantly. This poses a difficult problem to traditional search engines in keeping their recommendations up-to-date. It is not uncommon that a search engine recommends obsolete links.

Query: world weather forecast

		Query Result	
Rank	First set	3 days after	1 week after
1	index page to world weather highlights	index page to world weather highlights	index page to world weather highlights
2	index page to topics on weather and earth science	index page to ongoing research projects in the world	index page to ongoing research projects in the world
3	index page to ongoing research projects in the world	world weather forecasting articles	world weather forecasting articles
4	world weather forecasting articles	index page to topics on weather and earth science	weather section index page
5	index page to weather forecasts for cities outside the U.S.	weather section index page	index page to topics on weather and earth science

Figure 9: Newspaper: Comparison of query results from snapshots taken on different dates, $k = 2$, $\alpha = 0.2$.

Also, the search engines have to be constantly on the look for changes and updates to maintain its inverted index timely.

As we have discussed, anchor points (especially those for general concepts) tend to be indexes. For example, in Sports, the best anchor point of <<NBA>> is the NBA index page. We note that anchor points are relatively stable for two reasons: First, indexes usually have long life-spans. Second, although the content of anchor points change (we have different headlines every day for an interesting world), their ranks against queries (which take into account the aggregate contents of their k-neighborhoods) stay relatively stable.

To verify the stability of anchor points, we collected different snapshots of each information source over a period of time, applied the same set of queries on the snapshots, and compared the results. Figure 9 shows an example on Newspaper.

From the experiment, we see that even though there are minor re-orderings of the recommendations, the set of anchor points stays relatively unchanged. Anchor points are therefore fairly stable in the sense that they remain the best starting points for retrieving relevant information even though the content of an information source is constantly updated. The stability property allows the system to use a smaller *refresh rate*. That is, the potential function needs not be recomputed too often. As we will see in the next section, maintaining the potential function

Site	No. of documents	Occupancy of adjacency matrix	Avg. no. of links per page
Sports	9,116	0.54%	49.2
Newspaper	5,788	0.47%	27.2
FAQ	1,845	0.18%	3.3
NetNews	4,345	0.08%	3.4

Table 2: Occupancy of some adjacency matrices.

requires a higher computational cost than maintaining an inverted index used by traditional search engines. However, the lower recomputation rate can offset the higher CPU demand in anchor point computation. We will explore other techniques in reducing the computational cost in the next section.

6 Data Structures and Algorithms

As we have discussed in Section 4, computing the potential of a page with respect to a query requires the system to maintain certain information about the network of Web pages. This includes the scoring function ($f(X, a)$), the distance function ($D(X, Y)$), and the k-neighborhood of a page ($N_k(X)$). From this data, one can derive the potential function ($P_k(X, a)$) according to Equation 1 (page 10), and hence the anchor point rankings using Equations 2 and 3 (page 12). In this section, we discuss the data structures for storing the various information and the algorithms for their derivations.

Scoring Function

Given a page X and a keyword a, the scoring function, $f(X, a)$, returns a score which indicates how relevant page X is with respect to a. In our prototype, we calculate this score as the normalized word count of a in X according to the *TFIDF* formula [17]:

$$f(X, a) = \text{freq}_{a \in X} \times \log_2 \frac{n}{\text{docfreq}_a} + 1,$$

where $\text{freq}_{a \in X}$ is the occurrence frequency of keyword a in page X, n is the total number of pages indexed, and docfreq_a is the document frequency of keyword a. We represent the scoring function using an inverted index [20], just like most traditional search engines do.

Adjacency and k-reachability

The simplest way to store the adjacency relationship is to use a square bit-matrix. An entry (i, j) in the matrix is set to 1 if page j contains a hyperlink pointing to page i. This representation, however, is not very space efficient. We found that the adjacency matrices are very sparse for most information sources. Table 2 shows some typical adjacency matrix occupancies.

We notice that the occupancies are very small. Even for Sports and Newspaper whose

Site	No. of documents	Occupancy of k-reachability matrix	
		$k = 2$	$k = 3$
Sports	9,116	5.4%	27.8%
Newspaper	5,788	8.2%	35.4%
FAQ	1,845	4.58%	4.50%
NetNews	4,345	0.05%	0.09%

Table 3: Occupancy of some k-reachability matrices.

hypertext structures are broad, an average page is only adjacent to about 0.5% of all the pages located in the same source. The occupancy is much lower for narrow hypertext structures (such as FAQ and NetNews). To save space, our prototype represents the adjacency relationship using *inverted adjacency lists*. That is, for each page i, we maintain a linked list of pages which contain hyperlinks *pointing to* page i.

From the inverted adjacency lists, we derive the k-reachability relationship between pages. The goal is to find out, for a page A that is inside the k-neighborhood of another page B, how many hops it takes to go from B to A. Again, the k-reachability relationship can be represented by a matrix. The entry (i, j) gives the distance from page j to page i if the distance is smaller than or equal to k. Table 3 shows the occupancies of some k-reachability matrices for $k = 2$ and $k = 3$. As we have discussed in Section 5, Sports and Newspaper have a broad hypertext structure. For them, we use a small value of k (= 2). On the other hand, FAQ and NetNews have a very narrow hypertext structure, which allows a larger k (= 3) be used. In any case, Table 3 shows that with the appropriate value of k, the k-reachability matrices are sparse. We thus represent the matrix using linked lists. In particular, for each page A, we keep k linked lists, $L_i(A)$ $(1 \leq i \leq k)$, where $L_i(A)$ represents the set of pages whose distance *to* A is i, i.e., $L_i(A) = \{\text{page } B \mid D(B, A) = i\}$. Figure 10 shows a simple algorithm for deriving the inverted k-reachability lists and the size of each page's k-neighborhood from the inverted adjacency lists. The algorithm executes in iterations. During the i-th iteration, all $L_i(.)'s$ are generated. Essentially, given a page p, a page r is added to the set $L_i(p)$ if r contains a link to a page q whose distance to p is $i - 1$ and r cannot reach p in less than i hops (lines $6 - 9$).

Potential Function

The potential function $P_k(X, a)$ returns the potential value of a page X with respect to a keyword a. Similar to the scoring function, we represent the potential function using inverted lists. That is, for each keyword a, we maintain a list of pages which have non-zero potentials with respect to a, and their potential values. Figure 11 shows an algorithm for deriving the potential function.

The algorithm derives the inverted list of the potential function one keyword at a time. For each keyword a, the algorithm first retrieves the set of matching pages (S) using the inverted lists of the scoring function (line 3). The scores of the matching pages are then propagated to

Given $L_1()'s$, the inverted adjacency lists, derive the inverted k-reachability lists and the size of each page's k-neighborhood, n_k.

```
1    procedure k-reachability
2        initialize all n_k's to 0
3        for i = 2 to k do
4            for each page p do
5                L_i(p) = ∅
6                for each page q ∈ L_{i-1}(p) do
7                    for each page r ∈ L_1(q) do
8                        if r ∉ ⋃_{j=1}^{i} L_j(p)
9                            add r to L_i(p)
10                           n_k(r) = n_k(r) + 1
```

Figure 10: Deriving inverted k-reachability lists.

Given the scoring function and the inverted k-reachability lists, derive the potential function.

```
1    procedure potential_function
2        for each keyword a do
3        let S = {page p | f(p,a) > 0}
4        for each page p ∈ S do
5            P_k(p,a) = f(p,a)
6            for i = 1 to k do
7                for each page q ∈ L_i(p) do
8                    P_k(q,a) = P_k(q,a) + α^i × f(p,a)
```

Figure 11: Deriving the potential function.

those pages that can reach the matching pages within k hops (lines $5-8$).

Query processing

Given a query Q that is composed of the keywords $\{a_1, ..., a_m\}$, to calculate the potentials of the pages with respect to Q, we first retrieve the inverted lists of the potential function corresponding to the keywords. This gives us S_i $(1 \leq i \leq m)$ where $S_i = \{\text{page } p \mid P_k(p, a_i) > 0\}$. If Q is a conjunctive query, we find the intersection of the S_i's, and compute $Potential(X, Q)$ for only those pages $X \in \bigcap_{i=1}^{m} S_i$ using Equation 2. On the other hand, if Q is disjunctive, we find the set union and evaluate the pages' potentials according to Equation 3. The pages are then sorted in decreasing values of their Potentials.

6.1 Storage Requirement

Comparing with traditional search engines, recommending anchor points requires the system to maintain more data. This includes the k-reachability lists and more importantly the potential function. In this section we discuss the additional storage requirement and how we could reduce it.

As we have discussed previously, our system uses a small k value (2) for information sources (Sports and Newspaper) whose hypertext structures are broad (containing many links). From Table 3 (page 22), we see that for an average page in Sports or Newspaper, the size of its 2-neighborhood is less than 500 pages[3]. The storage overhead for maintaining the k-reachability lists is acceptable even for these broad hypertext structures. For example, our prototype uses 11 MBytes to store the 2-reachability lists of Newspaper whose documents require 110 MBytes of storage. The overhead is therefore only 10% of the document database. This storage overhead for sources with narrow hypertext structures is much smaller. For example, in NetNews, the storage overhead for maintaining the 3-reachability lists is only 1.9% of the document database size.

The storage requirement for the potential function, however, can be much higher than that of the scoring function maintained by traditional search engines. This is because while we keep an entry of $f(X, a)$ only if page X contains the keyword a, in the anchor point system, we need to keep an entry of $P_k(X, a)$ if keyword a occurs in *any pages* within the k-neighborhood of page X. As an example, in the Newspaper snapshot, there are 5,788 pages containing 36,021 keywords. A full potential function (or scoring function) would have as many as $5{,}788 \times 36{,}021 = 208$ million entries. Among these, about 137 million entries (65.8%) of the potential matrix are non-zero and have to be kept. Comparing with only 1.14 million non-zero entries for the scoring function, maintaining the potential function requires 120 times the storage of that is needed to maintain the scoring function alone.

[3]Estimated by multiplying the number of documents in the source by the occupancy of the 2-reachability matrix. For Sports, it is $9{,}116 \times 5.4\% = 492$; For Newspaper, it is $5{,}788 \times 8.2\% = 474$.

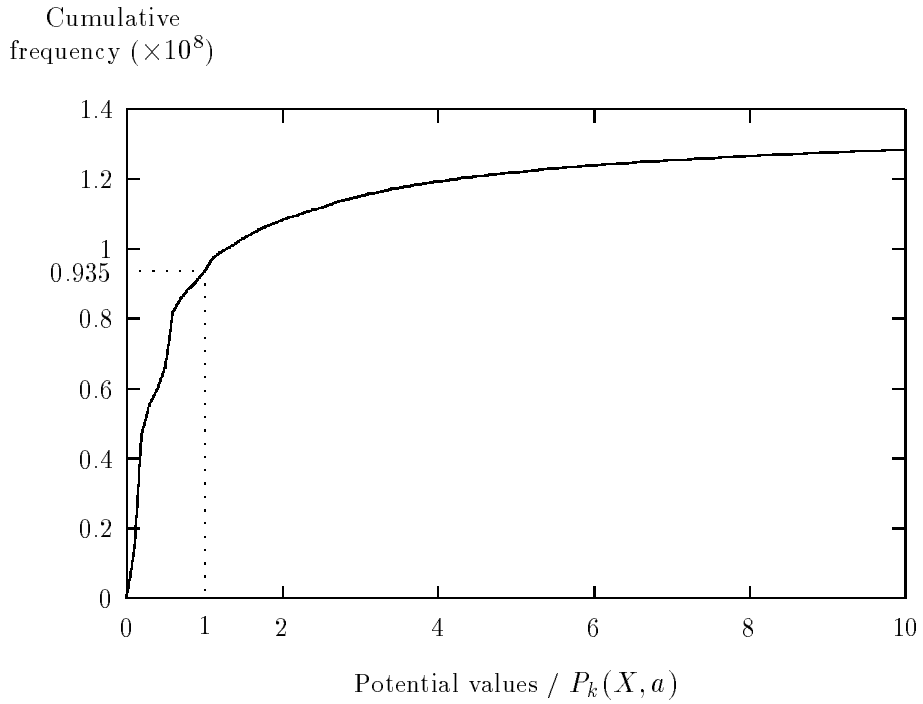


Figure 12: Cumulative frequency of non-zero potential function entries, $P_k(X, a)$ in (0,10].

One way to reduce the storage requirement is to throw away those *less significant* entries of the potential function. We observe that most of the entries contain, in fact, very small values. Again, taking Newspaper as an example, Figure 12 shows the cumulative frequency versus the potential values for those *non-zero* potential values in the range (0,10].

From the figure, we see that among the 137 million non-zero entries, 93.5 million of them (68.1%) are below 1.0, and 128 million of them (93.5%) are below 10.0. The remaining 6.5% of the potential function entries (9 million) have values range from 10 to a maximum potential of about 8,000.0. In summary, the values of the potential function are extremely skewed. We note that, given a query, a page which has a small potential value with respect to a keyword in the query is not likely to rank high in the recommendation. Approximating the *small* potential values by 0 (i.e., not keeping them) may not severely affect the quality of the recommendation. Throwing away *small*-value entries in the potential function can thus significantly reduce the storage demand. To apply this idea, however, we need to consider two questions. First, how shall we decide whether a potential function entry is small and therefore should be removed? Second, how will the removal of small entries affect the quality of the recommendation?

Instead of setting an absolute threshold value below which small potential function entries are purged, we take a relative approach. For each keyword a, we inspect its inverted list on the potential function and locate the maximum entry in the list. A potential function entry $P_k(X, a)$ is purged if its value is smaller than a cutoff percentage (c%) of the maximum entry's value. We thus have different cutoff values for different keywords. Note that the higher the cutoff threshold percentage is, the more entries are purged. Table 4 shows how much the storage requirement is reduced for various cutoff threshold c in Newspaper. We see that a large number of entries contain very small values and can thus be removed. For example, more than 90% of the non-zero entries are purged by setting a cutoff threshold of 10%. The storage requirement

c	Number of entries remained	Storage required compared with full table	% of non-zero entries purged
1%	82,982,804	39.8%	39.4%
2%	64,702,712	31.0%	52.8%
5%	26,142,052	12.5%	80.9%
10%	11,130,952	5.3%	91.9%
20%	3,335,046	1.6%	97.6%
30%	985,299	0.5%	99.3%

Table 4: Storage reduction versus cutoff threshold.

```
Query: world weather forecast

Rank   Recommendations before cut        After cut
----   ------------------------          ---------

1      index page to world weather       index page to world weather
       highlights                        highlights

2      index page to topics              index page to topics
       on weather and earth              on weather and earth
       science                           science

3      index page to ongoing             index page to ongoing
       weather research projects         weather research projects
       in the world                      in the world

4      world weather forecasting         world weather forecasting
       articles                          articles

5      index page to weather             index page to all sections
       forecasts for cities              in Newspaper
       outside the U.S.
```

Figure 13: Query results after removal of low potentials, $c = 10\%$, $k = 2$, $\alpha = 0.2$.

of the potential function can thus be significantly reduced.

Removing small entries from the potential function, however, reduces the amount of *knowledge* of the system. It is thus interesting to see how that would affect the system's performance in answering queries. We repeated some of the user queries after removing small potential values with a cutoff threshold of 10%. We found that in most cases, the cut did not affect the high-ranked recommendations at all. Figure 13 gives an illustrative example showing the pre-cut and the post-cut recommendations (top 5 only) when the query `<<world & weather & forecast>>` was submitted to `Newspaper`. From the example, we see that the top 4 recommendations stay the same. The 5th-ranked recommendation is changed from a weather forecast page to an index page where one could locate the weather section. The quality of the answer set remains very good.

We remark that removing small potential entries has a very mild effect on single-keyword

queries. This is because only low-ranked recommendations are removed from the answer sets. For a multiple-keyword conjunctive query, however, a page may be kicked out of the answer set if it has a low potential with respect to a keyword in the query even though it may has extremely high potentials with respect to the others. This kind of omission may not be too bad if the user gives equal weight to all the query keywords. For a disjunctive query, the page will not lose much in its ranking because it is supported mainly by those keywords for which it has high potentials (see Equation 3).

7 Conclusion

We identified four sources of ineffectiveness of traditional search engines and introduced the concept and use of anchor points. Given a user query, the set of anchor points is a set of key pages from which the larger set of documents that are relevant to the query can be easily reached. The use of anchor points help solve the problems of huge answer set and low precision suffered by most search engines. The major improvement is achieved by considering the hyper-link structures of the relevant documents, and by providing a summary view of the result set. We have implemented a prototype based on the concept of anchor point. Comparisons were made to traditional search engines. We found that our approach gave higher ranks to pages (such as indices) that provided better starting points for accessing relevant pages. On the other hand, traditional search engines tend to ignore the logical structure of hyper-documents, and relevant pages are distributed unpredictably in the answer set.

We have addressed some implementation issues of an anchor point indexing system. In particular, we discussed how the various information and data for computing anchor points are efficiently maintained. We also showed that anchor points are fairly stable. Frequent update of the anchor point index is thus unnecessary. This offsets the CPU requirement of handling a larger data structure (the potential function) than traditional search engines require (the scoring function).

References

[1] Armstrong et. al. A Learning Apprentice for the World-Wide Web. *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*, AAAI Press, 6-12, 1995.

[2] M. Balabanovic, Y. Shoham. Learning Information Retrieval Agents: Experiments with Automated Web Browsing. *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Resources*, Stanford, CA, March 1995.

[3] De Bra, R. D. J. Post. Information Retrieval in the World-Wide Web: Making Client-based Searching Feasible. *Proceedings of the First International World-Wide Web conference*, Geneva, 1994.

[4] H. Chen. Knowledge-based Document Retrieval: Framework and Design. *Journal of Information Science* 18:293-314, 1992.

[5] H. Chen, K. J. Lynch. Automatic Construction of Networks of Concepts Characterizing Document Databases. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5): 885-902, 1992.

[6] O. Etzioni, D. S. Weld. Intelligent Agents on the Internet: Fact, Fiction, and Forecast. *IEEE Expert* 10(4): 44-49, August, 1995.

[7] S. Feldman. Just the Answers, Please: Choosing a Web Search Service. *The Magazine for Database Professionals*, May, 1997.

[8] L. Gravano et. al. The Efficacy of GlOSS for the Text Database Discovery Problem. *ACM SIGMOD'94*, 1994.

[9] L. Gravano et. al. Precision and Recall of GlOSS Estimators for Database Discovery. *PDIS'94*, 1994.

[10] L. Gravano et. al. Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. *VLDB'95*, May 1995.

[11] B. Grossan. Search Engines: What They Are? How They Work? *http://webreference.com/content/search/features.html.*

[12] Gudivada V.N. Information Retrieval on the World Wide Web, *IEEE Internet Computing*, Vol. 1, No. 5, 1997, pp.58-68.

[13] D.L. Lee et. al. Document ranking and the Vector-Space Model. *IEEE Software*, Vol. 14, No. 2, Mar/Apr 1997, 67-75.

[14] J. Lee et. al. Intelligent Agents for Matching Information Providers and Consumers on the World-Wide-Web. *Hawaii International conference on System Sciences*, 1997.

[15] H. Lieberman. Letizia: An Agent that Assists Web Browsing. *International Joint conference on Artificial Intelligence*, 1995.

[16] J. Nielsen. The Art of Navigating Through Hypertext. *Communications of the ACM*, 33(3): 297-310, 1990.

[17] G. Salton. Automatic text processing: the transformation, analysis, and retrieval of information by computer. Mass: Add-Wesley, 1989.

[18] The Web Robots FAQ. *http://info.webcrawler.com/mak/projects/robots/faq.html*

[19] B. Kao, J. Lee, D. Cheung, C.Y. Ng. Recommending Anchor Points in Structure-Preserving Hypertext Document Retrieval. *22nd International Computer Software and Application Conference*, 1998

[20] W. B. Frakes and R. Baeza-Yates. Information Retrieval – Data Structures and Algorithms. Prentice Hall, 1992.