Cha-Cha:
A System for Organizing Intranet Search Results

Michael Chen* Marti Hearst† Jason Hong* James Lin*

*Computer Science Department
445 Soda Hall
University of California, Berkeley
Berkeley, CA 94720-1776
{mikechen,jasonh,jimlin}@cs.berkeley.edu
http://www.cs.berkeley.edu/~mikechen

†School of Information Management & Systems
102 South Hall
University of California, Berkeley
Berkeley, CA 94720-4600
hearst@sims.berkeley.edu
http://www.sims.berkeley.edu/~hearst

Abstract

Although search over World Wide Web pages has recently received much academic and commercial attention, surprisingly little research has been done on how to search the web pages within large, diverse *intranets*. Intranets contain the information associated with the internal workings of an organization.

A standard search engine retrieves web pages that fall within a widely diverse range of information contexts, but presents these results uniformly, in a ranked list. As an alternative, the Cha-Cha system organizes web search results in such a way as to reflect the underlying structure of the intranet. In our approach, an "outline" or "table of contents" is created by first recording the shortest paths in hyperlinks from root pages to every page within the web intranet. After the user issues a query, these shortest paths are dynamically combined to form a hierarchical outline of the context in which the search results occur. The system is designed to be helpful for users with a wide range of computer skills. Preliminary user study and survey results suggest that some users find the resulting structure more helpful than the standard retrieval results display for intranet search.

1 INTRODUCTION

Although search over World Wide Web pages has recently received much academic and commercial attention, surprisingly little research has been done on how to search the web pages within large, diverse *intranets*. Intranets contain the information associated with the internal workings of an organization.

Most web site search engines present search results as a ranked list of titles and metadata such as URLs and file size. The context in which the pages exist, and their relationships to one another, cannot be discerned from such a display. Figure 1 shows an example of a list view returned as a result of a query on "earthquake" using a commercial search engine within our university's web site. From this view, it is difficult to tell what the relationships are among the different search hits.

As a remedy, many authors have called for an organization to be imposed on the Web. Directory services such as Yahoo organize web pages according to pre-defined topics. Although such topics are often intuitive, this approach does not scale well because the web pages are assigned to categories manually. Furthermore, most web directories only cover organizations' home pages; standard search engines must be used if the user wants to find information within an intranet.

By contrast, we are interested in organizing the hits returned as the result of queries *within* large, heterogeneous intranets, such as those found at universities, corporations, and other large institutions. These sites are often quite complex, consisting of a wide variety of document genres, including home pages, product information, policy statements, research reports, current events, and news.

Grouping of retrieved documents may be especially important when the user has issued a very short or vague query (user queries tend to consist of only a few words [19, 7]). Short queries often return a heterogeneous set of documents that cover a wide

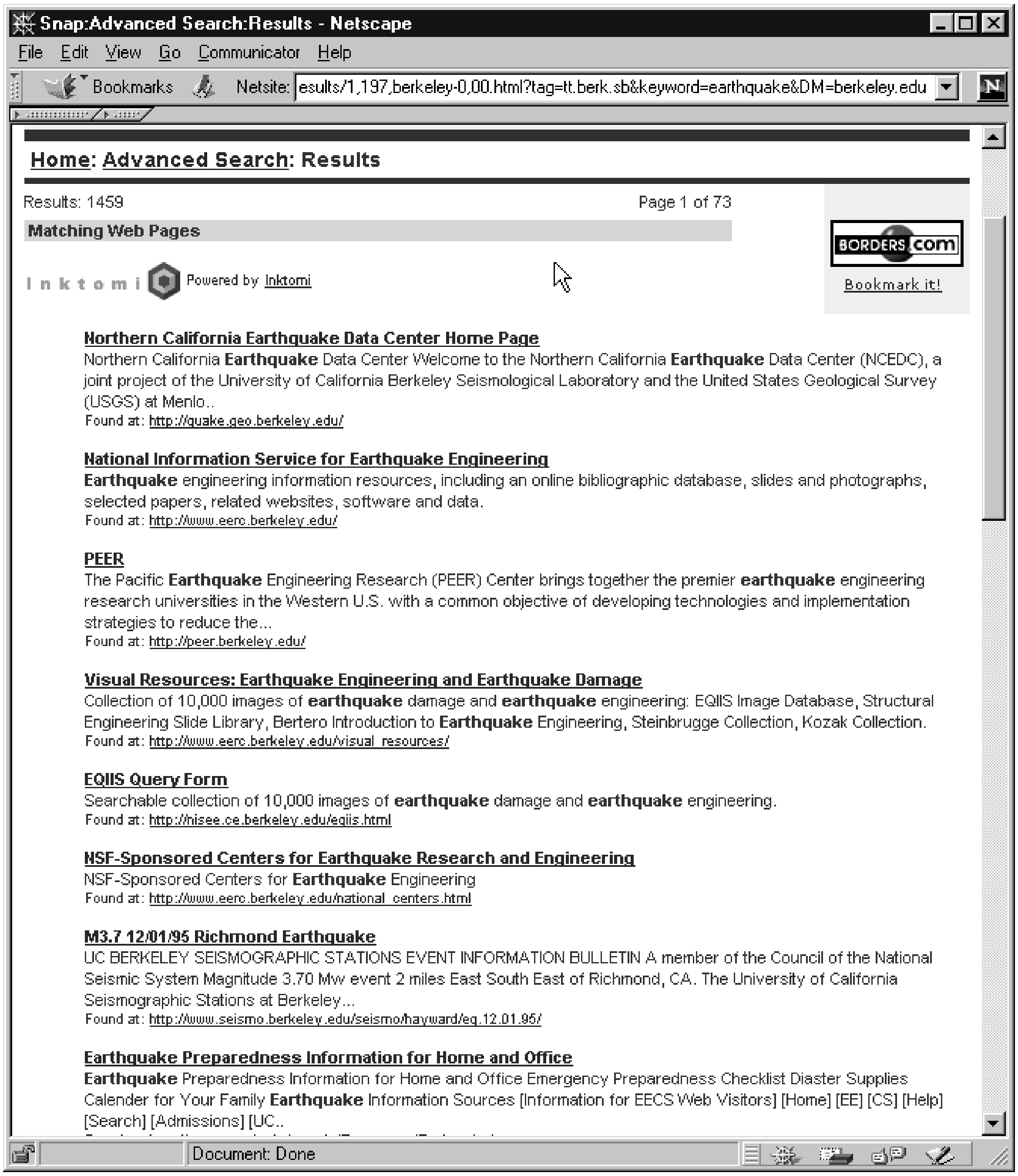

Figure 1: A list view of a commercial search engine on the query "earthquake" within the UC Berkeley intranet.

range of topics. In this situation, retrieval results should suggest the kinds of information available and guide the user to appropriate starting points (such as servers within an intranet).

We have developed a system called Cha-Cha whose goal is to provide fast, well-organized search results for queries within web intranets.[1] Cha-Cha imposes an organization on web site search results by recording the shortest paths, in terms of hyperlinks, from server root pages to every web page within the intranet. After the user issues a query, these shortest paths are dynamically combined to form a hierarchical outline of the context in which the search results occur. This outline structure shows the home pages of the servers on which the search hits occur, as well as the titles of the hyperlinks between the home pages and the search hit.

Figure 2 shows an example on the query "earthquake" within our university's web pages. Note that the interface tightly couples the specification of search with navigation of available hyperlinks.

The top levels of the hierarchy typically identify the servers that the search hits reside on. In this figure, the top levels visible are those associated with a national earthquake center (located on campus), an educational unit on earthquakes (located with a science education site, also on campus), and earthquake research within the campus civil engineering department. The organization pulls out the home pages associated with the search hits, providing not only context in which to embed the hits, but also conveniently indicating the higher level starting points for each subcollection of documents.

We designed the system to make the interface useable by all members of the community, even those with slow computers, low bandwidth connections, and old versions of web browsers. For this reason we use standard HTML and we keep the number of graphics low [26]. We also wanted the interface to look as familiar as possible while still providing added functionality, given research results showing that users do not like to switch to unfamiliar interfaces [21]. We have found that the outline metaphor is familiar enough that users understand the interface rapidly.

The functional goals of the system are to (1) help users better understand the scope of their search results, (2) find useful information more quickly than with a standard view, and (3) learn about the structure of the web site. A major assumption behind this research is that the shortest path of links from a root page to a target web page is meaningful to users. Our subjective experience with the use of this display, and especially in comparison with interfaces that show no context, is that the shortest path information is indeed a useful, coherent way to group the search results. This result is less surprising when taking into account the fact that most hyperlinks are created deliberately by human authors of web pages, who often organize pages into topics.

The remainder of this paper discusses related work, details about the system implementation, and user assessments of the interface.

2 RELATED WORK

This work is closely related to that of SuperBook [9], which demonstrated that showing hit search results in the context of the chapters and sections of the manual from which they are drawn can improve users' information access experiences. The AMIT system [30] indexed a web site covering a specific topic (sailing) in a similar manner, providing a Superbook-like focus-plus-context environment to place search results in context. To our knowledge this system has not been evaluated nor extended to a large heterogeneous web site. The WebTOC system [25] imposes static hierarchical table of contents over a single web site, but focuses on showing the number of pages within a subdirectory and does not integrate the results of search into the outline view. The examples explored for both AMIT and WebTOC were well-structured single web site prototypes organized around a topic (sailing and museum exhibits, respectively).

Cha-Cha demonstrates the application of the idea across a very large, heterogeneous web site that is an order of magnitude larger than those used by these other systems, and is used operationally by thousands of users. Many difficult system-level problems had to be solved to make this approach scale, both because of the larger size of the dataset and the greater heterogeneity in the search results. Cha-Cha also resolves graph-merging issues that these other systems do not address.

A major problem with WebTOC, along with other attempts to provide categorical access to information (such as Yahoo), is that they do not couple navigation with *ad hoc* search (see [1, 12] for more discussion of this point). A navigate-only interface to a large text collection forces the user to contend with the entire contents of the site at once.

[1]The name Cha-Cha stands for Contextualized Hierarchical information Access by Chen, Hearst, and Associates. The system can be found at http://cha-cha.berkeley.edu

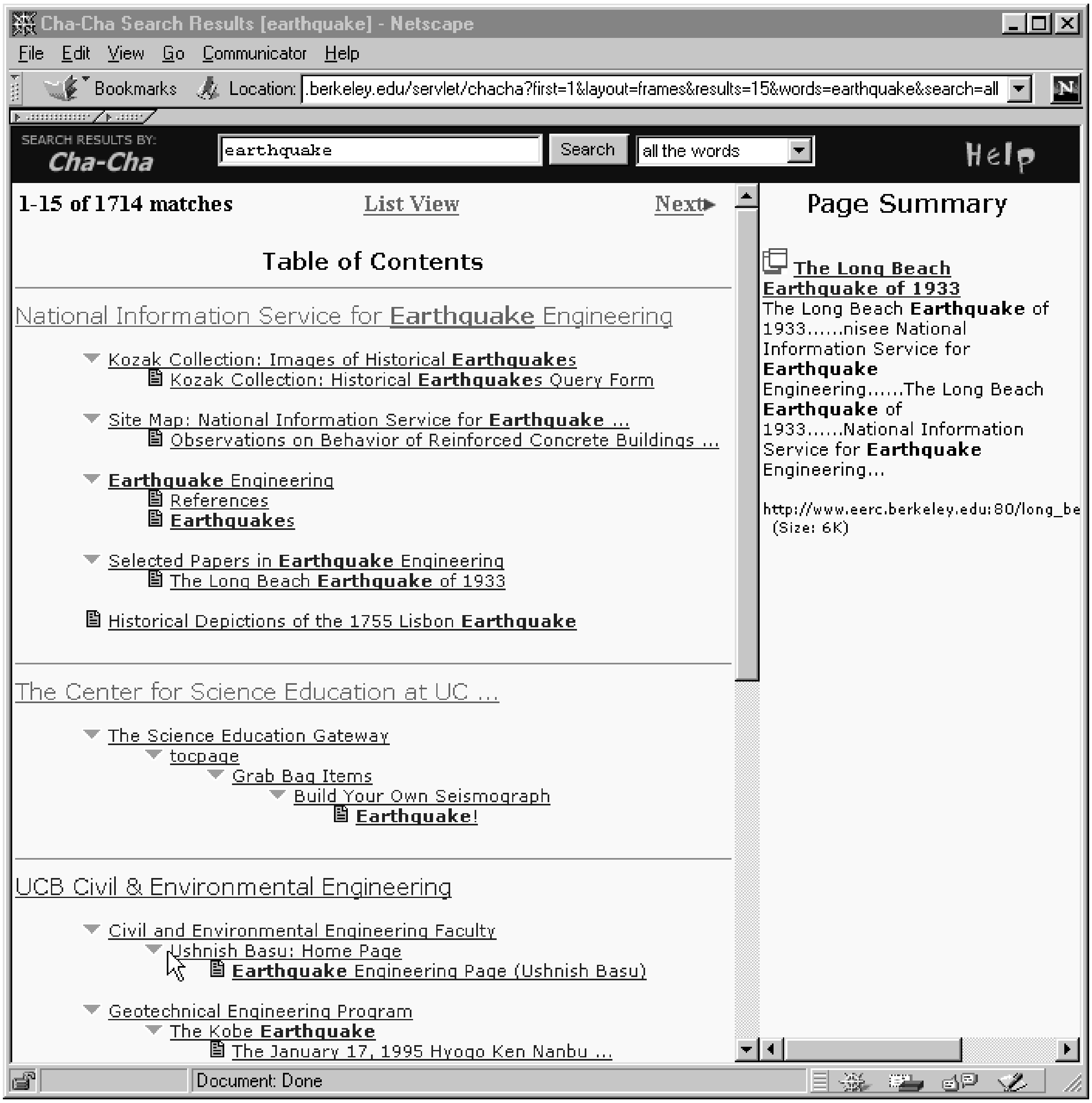

Figure 2: The outline view of the current implementation of Cha-Cha search on the query “earthquake” .

Many web sites show site maps which suffer from the same limitation. Users can browse or navigate the structure of the site, but cannot search within or across that structure.

The WebGlimpse system [22] provides crawling and indexing of an intranet (both local and external links). It also allows system administrators to define link "neighborhoods" in one of two ways: all pages within k links of a given page, or all pages within a file system subdirectory. (We explored this approach in Cha-Cha but found hyperlink path context to be more intuitive than subdirectory paths.) During search, neighborhood information in WebGlimpse is used only to restrict the set of pages that a search is conducted on; it is not used to show the context of the search results. By contrast, Cha-Cha allows search over many different neighborhoods simultaneously, showing in the search results a summary of which neighborhoods the search hits fall into.

The Connectivity Server [3] determines all inlink and outlink information for a given URL across the entire Web. This can be a useful component for a system that ranks pages according to "popularity" as suggested by inlink information (as in Kleinberg et al.'s "authority pages" [15] or the Google system [4]). The authors envision this as a method for viewing the Internet "neighborhood" of a given web page, but do not use the link information to provide context for search results. Furthermore, because results are shown in terms of their relationship to the entire Web, context within a given intranet is not provided.

The WebCutter system [20] (now called Mapuccino) allows the user to issue a query on a particular web site. The system crawls the site in real time, checking each encountered page for relevance to the query. When a relevant page is found, the weights on that page's outlinks are increased. The subset of the web site that has been crawled is depicted graphically in a nodes-and-links view. Other researchers have also investigated spreading activation among hypertext links as a way to guide an information retrieval system [10, 24].

The emphasis of these systems is on the dynamic crawling of the web sites. Unfortunately, this kind of display does not provide the user with information about what the *contents* of the pages are, but rather only shows their link structure. This is in contrast with Cha-Cha which retrieves all search hits at once, whether or not they are close to a starting point. Cha-Cha also emphasizes the display of the contents in a readable form, as previous research indicates that users find this more helpful than nodes-and-links views [14, 28].

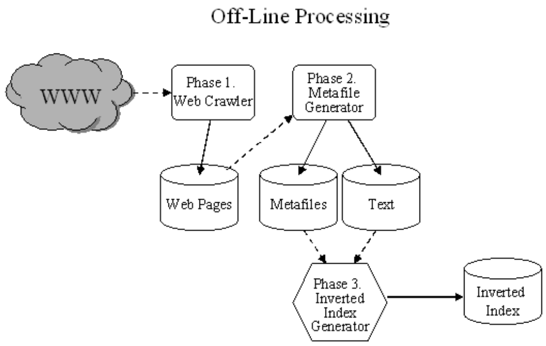

Figure 3: The architecture for the off-line indexing components of Cha-Cha.

3 SYSTEM IMPLEMENTATION

The Cha-Cha system has two main parts: an off-line indexing component in which the web site is crawled and the metadata and indexes are generated, and an on-line query processing component in which the system receives and responds to user requests. Figures 3 and 4 illustrate the two main components of the architecture; all code is written in Java unless otherwise noted.

The indexing component has three main phases: the web crawler, the metafile generator, and the indexer (see Figure 3). The web crawler stores a mirror of the intranet's web pages on its local file system. The metafile generator processes these files to precompute shortest path information and other meta-information, storing the metadata on the local file system. The indexer converts the metadata and the text into an inverted index to be used by the search engine backend, the Cheshire II system [18]. Each of these components is discussed in detail below, followed by a discussion of the query processing component.

3.1 The Web Crawler

In order to ensure thorough web coverage and to create the necessary metadata, we have written a custom web crawler. To uniquely identify each site, we record the MD5 hash[2] of the root page of each

[2] http://www.ietf.org/rfc/rfc1321.txt

host name found. If the root page of a new host has the same hash as a previous host, then the new host name is mapped to the previously known name in all URLs subsequently encountered [8]. This technique eliminates duplicates caused by host name aliases, DNS round-robin, virtual hosts, and unique servers that mount the same web pages at the root level. A global breadth-first search algorithm is used to order the URLs [5], and a page is retried for up to three times if errors are encountered during retrieval. Dynamically generated pages are not crawled.

The web crawler is given a list of URLs from which to start (e.g., the home page at www.berkeley.edu). The crawler is restricted to following links that fall only within a set of domains (e.g., all of *.berkeley.edu), while obeying the robots exclusion standard.[3] The crawler mirrors the full text of the web pages onto disk. This is needed to allow for extraction of sentences for page summaries.

3.2 The Metafile Generator

After the web pages are recorded, the metafile generator extracts hyperlink relationships. It begins at the root page of the main server. The HTML of the current page is parsed and all the outlinks to pages that have not yet been processed are placed on a queue. The system stores information about the page in a disk-based storage system.[4] This information includes a count of the shortest distance found so far from the root to the page, along with the corresponding shortest path(s). Then the next page is taken off the top of the queue and the process repeats until the queue is empty. If later in this process a page is encountered again, and was reached via a shorter path than before, the database entry for the page is updated to reflect the shorter path. Pages are allowed to contain multiple shortest paths (of equal length). Simple algorithms are used to assign categories such as personal home page and department home page, which can later be used as a search criteria. The metafile generator also records the title, domain, URL, page length, date last modified (if available), inlinks and outlinks.

In the initial implementation of the system, the root node was the home page of our institution, and all shortest paths were generated relative to this root page. However, this approach can produce misleading results, because sometimes the shortest path within a local subdomain is more meaningful than the shortest path computed globally or the entire intranet. (A related idea is discussed by Terveen and Hill [29].) To remedy this problem, we have implemented a variation of the algorithm that combines *local* and *global* shortest path information. This allows the shortest path information to be organized more modularly.

To make use of both local and global information, a two-pass scheme is used for metafile generation. First, shortest paths are computed within each logical domain, starting from the root page associated within the local logical domain. Then, global shortest paths are computed in the same way, beginning with the home page of the entire organization. Whenever a page is found using the global pass that has not yet been encountered in any of the local passes, this page and its shortest paths are added to the metafile database. This second pass finds "orphan" pages that might have been missed by the local passes, but gives priority to shortest paths found locally.

The two-phase approach trades file reading time for simplicity, as files have to be read and parsed twice. This inefficiency can be removed by keeping track of both global and local paths simultaneously for each page. The time to generate the metafiles for a 200K collection is currently less than 5 hours, which is adequate for our institution.

3.3 The Indexer

The search engine backend is the Cheshire II system [18], (written in C). This system creates an inverted index [2] based on the full text of the pages and attribute information found in the metadata files. Cheshire II works with SGML markup. After reading in a Document Type Definition (DTD) [27] describing the format of the metafiles, the system creates indexes that provide efficient access to search terms embedded within the full text, titles, and other forms of metadata.

3.4 Query Processing and the User Interface

Because studies show that many users are reluctant to switch from familiar to unfamiliar interfaces and systems, one of our principle design goals to was to create an interface as similar as possible to the status quo, while providing added functionality.

Users access the system via a web browser that communicates with the Cha-Cha frontend, which is

[3]http://info.webcrawler.com/mak/projects/robots/norobots.html

[4]Written in C, see http://www.sleepycat.com.

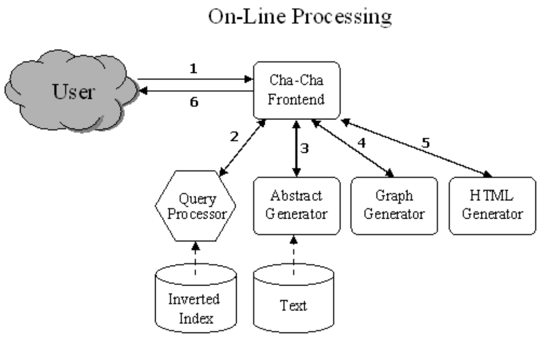

Figure 4: The architecture for the on-line (interactive) query processing components of Cha-Cha.

a Java servlet hosted on an Apache server.[5] (See Figure 4, step 1.) The Cha-Cha frontend formats the user query and sends it to the Cheshire II backend (step 2). The frontend requests the metadata for the first k hits, where currently k is set to 25. The output of queries are ranked using a probabilistic algorithm [6]. The two-tier design will allow us in future to use load-balancing techniques across several servers so the system can scale well.

The user can place the interface into one of two modes: list mode or outline mode. If list mode is selected, the system creates an HTML page containing the titles, summaries, page size, date, and URL for the first k hits (step 3). The page also shows the total number of hits for the query. If there were more than k hits for the query, hyperlinks are shown at the top and bottom of the page indicating that more pages of search hits are available. Titles are hyperlinked to the actual pages to which they refer.

If outline mode is selected, the system builds up a hierarchy from the shortest paths associated with each of the k hits (step 4; how this is done is described below). The outline layout is placed into two frames; the first acts as a "table of contents" or a "category hierarchy." This outline view is generated by recursively traversing the hierarchy data structure and generating HTML (step 5). Small icons are associated with the titles of search hits. These icons, if clicked, bring up a display of the document summary in the righthand frame. The titles within the table of contents (search hits as well as their contextualizing information) are hyperlinked to the actual web pages to which they refer. Finally, the HTML pages are displayed in the user's client browser (step 6).

[5]http://www.apache.org

3.5 The Page Summaries

There is strong evidence that highlighting query terms in the context in which they occur in the page helps user determine why the document was retrieved and how it might be relevant to their information need [17, 23].

The keyword-in-context (KWIC) summaries consist of a sentence extracted from the beginning of the document followed by up to three sentences containing search terms. The search terms themselves are shown in boldface. Figure 5 shows an example on the query "contact lens". If the query consists of q different terms, sentences containing all q are favored over those with fewer, failing these, sentences containing $q-1$ are favored, etc. In the case of ties, sentences from the beginning of the document are favored over those found later. To help retain coherence of the excerpts, selected sentences are always shown in order of their occurrence in the original document, independent of how many search terms they contain.

3.6 The Graph Merging Algorithm

After all of the shortest paths have been found, they are used to build a hierarchy, starting from the root, in which common paths are merged together. This merging is partly responsible for the grouping and compression of search results.

As noted above, there is often more than one shortest path to any particular search hit. This means that a graph built from all of the shortest paths is a directed acyclic graph (DAG). We choose to show search hits only once within the hierarchy in order to reduce the amount of extra information presented to users, and because we suspect that seeing the same page in two locations in the hierarchy will be more confusing than helpful. We also assume that it is better to group many related hits together within one subhierarchy, if possible, rather than scattering the same set of hits across many different subhierarchies. This assumption is based on the results of our empirical work which showed that documents relevant to a query tend to fall into the same one or two clusters when text clustering is employed [13]. In general, grouping related documents together seems to facilitate rapid discarding of non-relevant groups [11].

Although we do not use clustering for computing similarity here, we assume that a human author groups pages together via hyperlinks because they are similar to one another in some sense. Thus hyperlink structure can provide a kind of supervised

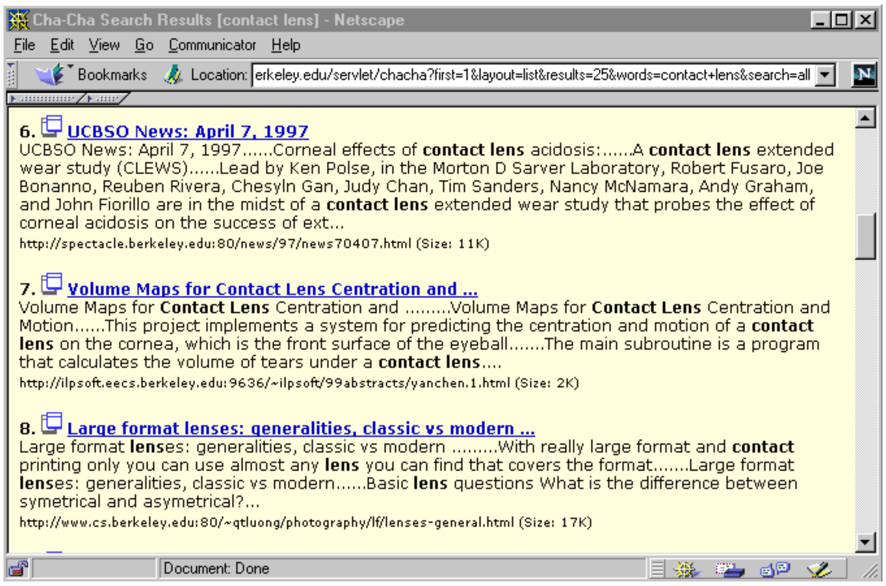

Figure 5: Examples of keyword-in-context summaries.

similarity metric that document clustering attempts to uncover in an unsupervised manner. The hyperlink structure has the added advantage of human-understandable labels (in the form of the page titles) and a uniform granularity of detail, both of which are lacking in clustering algorithms [11].

Based on these assumptions, the layout has the following goals (assuming that search hits are leaves in the final hierarchy).

(i) Group (recursively) as many pages together within a subhierarchy as possible; avoid (recursively) branches that terminate in only one hit (leaf).

(ii) Remove as many internal nodes as possible while still retaining at least one path to every leaf.

(iii) Remove as many edges as possible while still retaining at least one path to every leaf.

Items (ii) and (iii) are based on the assumption that the fewer levels of the hierarchy shown the better, since pilot studies suggest users prefer to have less extra information rather than more.

These desiderata require a non-standard graph algorithm[6] whose goal is to find the smallest subset of nodes that covers all the nodes one level below.[7] To do this correctly, every possible subset of nodes at depth D should be considered to determine the minimal subset which covers all the nodes at depth $D+1$. However, this would require 2^k checks if there are k nodes at depth D. Instead, a heuristic approach is used.

The main idea is to order nodes at each depth according to how many children they have and *eliminate* those nodes that do not uniquely cover nodes at the depth below them.

First, a top-down pass determines the depth of each node and the number of children it links to one level below. Next a bottom-up pass works from the deepest nodes (the leaves) up to the root. In other words, say the current deepest level is $D+1$. The nodes at level D are sorted in ascending order according to how many active children they link to at depth $D+1$. A node is *active* if it has not been eliminated in a previous step.

Every non-leaf node at level D is a *candidate* for elimination from the final graph. Those candidates with the least number of children are examined for potential to be eliminated first, because of goal (i). For each candidate C, if C links to one or more ac-

[6] Minimum spanning tree is inappropriate because internal nodes, as well as edges, are to be eliminated. A depth-first traversal in which the counts of the leaves are propagated up does not work because the graph is a DAG: If node N has two internal node children $N1$ and $N2$ and they both point to leaf L, N will be assigned an erroneous count of two children in such a traversal.

[7] To obtain a truly optimal result the algorithm should optimize elimination of nodes and edges at all levels of the hierarchy, not just between one level and the next. There is at least one case in which the heuristic approach using only one level at a time yields a suboptimal result, but in practice this kind of situation seems to occur only rarely.

tive nodes at depth $D + 1$ that are not covered by any of the other active candidates, then C cannot be eliminated. Otherwise C is removed from the active nodes list for depth D. After a level is complete, there are no active nodes at depth D that cover only nodes that are also covered by another active node at D. This procedure continues recursively up to the root. It works despite the DAG structure because the edges used correspond only to shortest paths at every level. The running time is order $(k\ log\ k) + 2k$ for each level, instead of 2^k. Figure 6 shows the pseudocode for the main loops of the algorithm.

```
mergeShortestPaths(paths) {
  init();
  NodesAtDepth[] =
     findShortestPathDepths(paths);
  pruneNodes(NodesAtDepth);
  return(buildPathTreeFromNodes());
}

pruneNodes(NodesAtDepth[]) {
  Vector tobeCov, candidates;
  for(int d = MaxDepth-1; d>0; d--) {
     tobeCov = NodesAtDepth[d];
     candidates = NodesAtDepth[d-1];
     NodesAtDepth[d-1] =
      getCovering(tobeCov, candidates);
  }
}
```

Figure 6: Pseudocode for the main loops of the graph merging algorithm.

After nodes have been eliminated, the hierarchy must be built up while still attempting to retain the rank ordering from the search engine. This is accomplished as follows. The leaf nodes (search hits) are sorted in ascending order according to their rank in the search results (a rank of 1 means the best-ranked hit). Beginning with an empty tree and the first hit, a path is found from that hit through the active nodes at each level above it to the root. The path must travel through edges from the original set of shortest paths. The parent with the largest number of still active children is chosen, in order to help achieve goal (i). When the root is reached, a new path has been created; this path becomes the beginning of the output tree. This procedure is repeated with the rest of the search hits in ranked order, with an added check: when selecting a parent, if one of the parent choices is already in the final tree, chose that over other parent choices (to help achieve goal (ii)).

The tree is converted to an HTML hierarchy by traversing it in a depth-first manner. The choice of which sibling to traverse next is determined by the order in which the siblings were entered into the tree. Thus the rank ordering is preserved as much as possible while still grouping search hits together within subhierarchies.

3.7 System Status

Cha-Cha has been available from our institution's home page since August 1998, receiving on average approximately 3000 queries per weekday during the school year. During a one week period, for 17831 queries, the outline view was used 16006 times for 14330 unique IP addresses. (Since the outline view is the default setting we cannot assume that people deliberately choose this view.) Our target response time was 3 seconds per query on average. We achieved this goal; for 17831 queries run during one week in August, the average time required by the system was 3.02 seconds running on a Sparc Ultra II. The search engine backend took 2 seconds on average while the Java frontend took 1 second. More recently we have obtained a Sun Enterprise 450. For 65440 queries during the month of July, 1999, the average time was 2.4 seconds/query. The current index covers more than 200,000 web pages and several groups within our institution are using Cha-Cha to search over their site's pages. All that is needed to enable this facility is the customization of a short HTML form.

4 USER ASSESSMENTS

As mentioned above, the functional goals of the system are to help users better understand the scope of their search results, find useful information more quickly than with a standard view, and learn about the structure of the web site. These are all difficult to evaluate empirically [16]. Below we describe a pilot study that attempts to assess some of these factors, a follow-up user study, and the results of a survey intended to uncover user preference information.

4.1 A Pilot Study

To assess the relative merits of the system we conducted a pilot study followed by a full study. The

	Question	Yes	No	NA
1.	I often find the outline view helpful.	56	31	75
2.	I often find the outline view confusing.	37	49	76
3.	The outline view sometimes helps me find information that a standard search engine does not.	57	30	75
4.	The outline view introduces unnecessary clutter.	37	51	78
5.	The outline view would be better if it showed less information.	33	51	78
6.	I usually prefer the list view with the summaries over the outline view.	42	39	81
7.	I usually prefer the Cha-Cha outline view to standard search engine results listings.	35	44	83

Table 1: Responses of 162 survey participants on questions about Cha-Cha in particular.

pilot study used an earlier version of the system on a smaller data set (about 10,000 pages). Seven people participated and four versions of the interface were compared. The participants had not used Cha-Cha prior to the study, and had no training on the interfaces.

The results showed that participants were able to understand a version of the outline view and found it easy to use. When participants were timed on eight question-answering tasks, the average time for outline view was 72.4 seconds/query while for the list view it was 99.7 seconds/query. Due to the small nature of the study, these results are only suggestive.

4.2 Follow-up Study

We conducted a more extensive user study on a later version of the system with a larger number of pages (about 100,000), but with inferior path structure, because the local/global distinction had not yet been implemented. This study involved 18 participants, 9 males and 9 females, from a wide range of undergraduate majors (one participant was a graduate student). The study compared only the outline view and the enhanced KWIC list view. This version of the list view contains more information than a standard web search engine, showing an abstract containing keywords in the context in which they occur in the web page, rather than just the first one or two sentences of the web page.

Study participants scored the two views on the question "How often would you use this interface if it were available" on a scale from 1 (never) to 7 (often). The outline view received an average score of 4.6 while the list view received a score of 4.9 (differences were not significant). However, when participants were timed on eight question-answering tasks, the average time for outline view was 109 seconds/query while for the list view it was 120 seconds/query (again, differences were not significant).

4.3 Survey Results

To attempt to measure preference information, a survey was recently placed on the home page for our institution, inviting the user community to express opinions about the usability of Cha-Cha and a commercial search engine, both of which provide search over the UC Berkeley web pages.

Because those people who choose to answer the survey are somewhat self-selected, the results should be considered to come from a biased sample. Nevertheless, the results should provide useful ballpark estimates on perceived usability and user preference. After several weeks, 162 responses were gathered. 96 responses were from UCB undergraduates (apparently; this was the default choice), 23 were from UCB Staff, 10 were UCB graduate students, with the remaining 33 from other categories.

Respondents were asked if they had to choose between Cha-Cha and the commercial search engine, which would they prefer. 38 respondents preferred Cha-Cha, 21 preferred the commercial engine, 61 marked no opinion, and 42 made no choice at all. Thus, of those who expressed an opinion of one system over the other, 64% preferred Cha-Cha. Bearing in mind that most users are reluctant to adopt to new interfaces, this is an encouraging statement in favor of this approach.

Table 1 shows the questions pertinent only to Cha-Cha and the number of responses. For these questions, respondents could mark Yes, No, or mark neither choice (NA). Over half the respondents find the outline view helpful and claim to be able to find information using it that they can't otherwise find.[8]

[8]The last question seems to be worded in a confusing manner, because in many cases respondents who were positive about other aspects of Cha-Cha responded No to this question.

If only those respondents who expressed an opinion are taken into account, then about two thirds find the outline view helpful ($56/87 = .64$).

Anecdotally, users that tell us they like Cha-Cha tell us it is often useful for especially hard-to-find information. In these circumstances they often end up looking at the hyperlink one or two levels above a hit and explore the web site in that general area.

We think there are three main reasons between the discrepancies in the preference results we see between the follow-up user study and the survey. First, the local/global improvements to the path generation discussed in Section 3.2 were made after the user study but prior to the survey. We think these changes improve the meaningfulness of the hierarchies in many cases. Second, we suspect that users grow to prefer the outline view as they become more familiar with the system. The benefits of a new interface cannot always be assimilated immediately, and may be especially difficult to appreciate in a timed test like that of our follow-up study. Survey users used the interface to achieve their own ends at their own pace. Third, some users may prefer Cha-Cha over the commercial search engine because Cha-Cha's list view is richer than that of the commercial search engine, and because the commercial engine displays advertisements.

5 CONCLUSIONS AND FUTURE WORK

We have described the motivation, architecture, algorithms, and user assessment results for a search engine interface that organizes search results over large intranets into coherent structures.

We are encouraged by the user study results, the initial survey results, and informal reactions by users of the system. We argue that if we are providing an interface that is preferred by a substantial subset of the user population, then we are providing a useful service. During the course of the pilot study and the fuller study, we have learned about improvements participants would like to see in the system design that might make the outline view substantially more effective than standard views. In future, we plan to investigate how to incorporate semantic information into the interface. We also plan to index a wide variety of organizations' intranets.

Acknowledgements We thank Ray Larson for adding code to Cheshire II to make it suit our needs, Keith Bostic and Margo Seltzer of SleepyCat Software Inc for allowing use of their database software package, Eric Brewer for arranging access to Inktomi for an earlier version of the system, Kevin Heard and Bryan Lewis for technical assistance, and Hal Varian for general support. Additionally, we thank Sun Microsystems for a generous equipment donation.

References

[1] M. Agosti, G. Gradenigo, and P.G. Marchetti. A hypertext environment for interacting with large textual databases. *Information Processing & Management*, 28(3):371–387, 1992.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Company, 1999.

[3] Krishna Bharat, Andrei Broder, Monika Henzinger, Puneet Kumar, and Suresh Venkatasubramanian. The connectivity server: fast access to linkage information on the web. In *Proceedings of the Seventh International World Wide Web Conference (WWW 7)*, 1998.

[4] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, 1998.

[5] Junghoo Cho, Hector Garcia-Molina, and Larry Page. Efficient web crawling through URL ordering. In *Proceedings of the Seventh International World Wide Web Conference (WWW 7)*, 1998.

[6] William S. Cooper, Fredric C. Gey, and Aitoa Chen. Probabilistic retrieval in the TIPSTER collections: An application of staged logistic regression. In Donna Harman, editor, *Proceedings of the Second Text Retrieval Conference TREC-2*, pages 57–66. National Institute of Standards and Technology Special Publication 500-215, 1994.

[7] W. Bruce Croft, Robert Cook, and Dean Wilder. Providing government information on the internet: Experiences with THOMAS. In *Proceedings of Digital Libraries '95*, pages 19–24, Austin, TX, June 1995.

[8] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. Mogul. Rate of change and other metrics: a live study of the world-wide web. In *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)*, Dec 1997.

[9] Dennis E. Egan, Joel R. Remde, Thomas K. Landauer, Carol C. Lochbaum, and Louis M. Gomez. Behavioral evaluation and analysis of a hypertext browser. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 205–210, May 1989.

[10] H. P. Frei and D. Stieger. The use of semantic links in hypertext information retrieval. *Information Processing & Management*, 31(1):1–13, 1994.

[11] Marti A. Hearst. The use of categories and clusters in organizing retrieval results. In Tomek Strzalkowski, editor, *Natural Language Information Retrieval*, pages 333–374. Kluwer Academic Publishers, 1999.

[12] Marti A. Hearst and Chandu Karadi. Cat-a-cone: An interactive interface for specifying searches and viewing retrieval results using a large category hierarchy. In *Proceedings of the 20th Annual International ACM/SIGIR Conference*, Philadelphia, PA, 1997.

[13] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19th Annual International ACM/SIGIR Conference*, pages 76–84, Zurich, Switzerland, 1996.

[14] Adrienee J. Kleiboemer, Manette B. Lazear, and Jan O. Pedersen. Tailoring a retrieval system for naive users. In *Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval (SDAIR)*, Las Vegas, NV, 1996.

[15] Jon Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[16] Eric Lagergren and Paul Over. Comparing interactive information retrieval systems across sites: The trec-6 interactive track matrix experiment. In *Proceedings of the 21st Annual International ACM/SIGIR Conference*, pages 164–172, 1998.

[17] Thomas K. Landauer, Dennis E. Egan, Joel R. Remde, Michael Lesk, Carol C. Lochbaum, and Daniel Ketchum. Enhancing the usability of text through computer delivery and formative evaluation: the superbook project. In C. McKnight, A. Dillon, and J. Richardson, editors, *Hypertext: A Psychological Perspective*, pages 71–136. Ellis Horwood, 1993.

[18] Ray R. Larson, Ralph Moon, Jerome McDonough, Lucy Kuntz, and Paul O'Leary. Cheshire ii: Design a next-generation online catalog. *Journal of the American Society for Information Science*, 47(7):555–567, 1996.

[19] X. Allan Lu and Robert B. Keefer. Query expansion/reduction and its impact on retrieval effectiveness. In Donna Harman, editor, *Proceedings of the Third Text Retrieval Conference TREC-3*, pages 231–239. National Institute of Standards and Technology Special Publication 500-225, 1995.

[20] Y. S. Maarek, M. Jacovi, M. Shtalhaim, S. Ur, D. Zernik, and I. Z. Ben Shaul. WebCutter: A system for dynamic and tailorable site mapping. In *Proceedings of the Sixth International World Wide Web Conference*, pages 713–722, 1997.

[21] Robert R. Mackie and C. Dennis Wylie. Factors influencing acceptance of computer-based innovations. In Martin Helander, editor, *Handbook of Human-Computer Interaction*, pages 1081–1106. Springer Verlag, 1988.

[22] Udi Manber, Mike Smith, and Burra Gopal. WebGlimpse – combining browsing and searching. In *Proceedings of 1997 Usenix Technical Conference*, 1997.

[23] Gary Marchionini. *Information Seeking in Electronic Environments*. Cambridge University Press, 1995.

[24] Filippo Menczer and Richard K. Belew. Adaptive information agents in distributed textual environments. In *Proceedings of the 2nd International Conference on Autonomous Agents (AGENTS-98)*, pages 157–164, May 1998.

[25] A. Nation. Visualizing websites using a hierarchical table of contents browser: Webtoc. In *Proceedings of the Third Conference on Human Factors and the Web*, Denver, CO, 1997.

[26] Jakob Nielsen. *Designing Excellent Websites: Secrets of an Information Architect*. New Riders Publishing, 1999. To appear. See www.useit.com.

[27] Natanya Pitts-Moultis and Cheryl Kirk. *XML Black Book*. The Coriolis Group, 1999.

[28] Marc Sebrechts, Joanna Vasilakis, Michael S. Miller, John V. Cugini, and Sharon J. Laskowski. Visualization of search results: A comparative evaluation of text, 2d, and 3d interfaces. In Marti A. Hearst, Fredric Gey, and Richard Tong, editors, *Proceedings of the 22nd Annual International ACM/SIGIR Conference*, pages 3–10, Berkeley, CA, 1999.

[29] Loren Terveen and Will Hill. Finding and visualizing inter-site clan graphs. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '98)*, Los Angeles, CA, April 1998. ACM.

[30] Kent Wittenburg and Eric Sigman. Integration of browsing, searching, and filtering in an applet for web information access. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, Late Breaking Track*. ACM, 1997.