# Comparison of Two Approaches to Building a Vertical Search Tool: A Case Study in the Nanotechnology Domain

**Michael Chau, Hsinchun Chen, Jialun Qin, Yilu Zhou, Yi Qin, Wai-Ki Sung, Daniel McDonald**
Artificial Intelligence Lab
Department of Management Information Systems
The University of Arizona
Tucson, Arizona 85721, USA
1-520-621-2748
{mchau, hchen, qin, yilu, yiqin, wai-ki, dmm}@bpa.arizona.edu

**ABSTRACT**
As the Web has been growing exponentially, it has become increasingly difficult to search for desired information. In recent years, many domain-specific (vertical) search tools have been developed to serve the information needs of specific fields. This paper describes two approaches to building a domain-specific search tool. We report our experience in building two different tools in the nanotechnology domain — (1) a server-side search engine, and (2) a client-side search agent. The designs of the two search systems are presented and discussed, and their strengths and weaknesses are compared. Some future research directions are also discussed.

**Categories and Subject Descriptors**
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *clustering, information filtering, search process, digital libraries.*

**Keywords**
Information retrieval, Web search engine, vertical search engine, Internet spider, Internet searching and browsing, post-retrieval analysis, indexing, noun-phrasing, self-organizing map, personalization, summarization.

## 1 INTRODUCTION

The Web has become a very rich source of information for almost any field, ranging from music to histories, from sports to movies, from science to culture, and many more. However, it has become increasingly difficult to search for desired information on the Web. Users are facing the problem of information overload [1], in which a search on a general-purpose search engine such as Google (www.google.com) results in thousands of hits.

Because a user cannot specify a search domain (e.g., medicine, music), a search query may bring up Web pages both within and outside the desired domain. For example, a user searching for "cancer" may get Web pages related to the disease as well as those related to the Zodiac sign. As a result, the user has to browse through the list of results to identify relevant Web pages, a task which requires significant mental effort. Directory services such as Yahoo! (www.yahoo.com) provide users with a hierarchy of classified topics. While the precision is high, recall rate suffers as each page included in the results has to be manually evaluated.

In this paper, we report our experience in building a customized search tool for the nanotechnology domain. Nanotechnology is the study of science and technology at the scale of nanometer – a billionth of a meter. It is one of the fastest growing disciplines in science and technology. The ability to manipulate substances at the molecular level provides great opportunities for various applications, such as nanoscale circuits for tiny computers, and nanobots that could travel through the bloodstream of a patient for disease diagnosis [27].

Because nanotechnology is young and fast growing, the Web has become an important medium for researchers and practitioners seeking up-to-date information. However, information sources and quality on the Web are diverse [18]. Useful nanotechnology information can include scientific papers, journals, technical reports, and Web pages of widely-varied quality, among others. As a result, users need to go to multiple information sources (Web sites, search engines, online academic databases, etc.) and conduct significant manual analysis to identify quality, time-critical information.

In addition, the nanotechnology domain encompasses a diversity of research perspectives and application areas such as nanoscale physics, nanoscale medicine, and nanoscale electronics. This has resulted in terminology and vocabulary differences [6, 12]. Researchers in different disciplines use different terminologies to describe their findings and user search terms may not be the same as the indexing terminologies used in databases. In addition, there also exists a dichotomy between academia and industry in


*JCDL '02*, July 13-17, 2002, Portland, Oregon, USA.

the field. The first seeks long-term research issues and the second focuses on short-term development.

Language differences make the problem worse, because researchers in the discipline come from a variety of regions and countries. Research findings written in a language other than English can be difficult to search for on the web.

Because of the diversity and overwhelming interest in the field, the speed of knowledge creation and information generation is faster than ever, resulting in a problem known as fluidity of concepts, which further complicates the retrieval issue [9]. A concept may be perceived differently by different researchers and may convey different meanings at different times. Subcategories in the field also are changing much more quickly than in established fields and new categories evolve frequently. As a result, building a successful domain-specific search tool to provide information search and analysis capabilities for this young and evolving field is very challenging and somewhat uncertain.

To address these problems, our project has undertaken to develop an integrated, "one-stop shopping" service for the domain. The project was divided into two phases. In the first, we aimed to develop for the domain a search tool with basic search and categorization functionalities. In the second phase, we focused on enhancing the analysis capabilities of the tools by incorporating functions such as vocabulary suggestion and multi-lingual support.

Two approaches have been adopted and compared. The first was to develop a domain-specific (vertical) search engine accessible on the Web for users to perform searches in the domain. This involved sending out spiders (search agent programs) to automatically collect Web pages in the nanotechnology domain and to create from these pages an index searchable by users through a Web interface.

The second approach involved building a client-side search tool – search agent that can be installed and run on a user's computer. This kind of search agent can either collect pages from the Web on-the-fly or retrieve search results from other search engines (a process called meta-search). Compared with the first approach, this method allows more computer processing time and memory to be allocated to the search process and more functionalities are possible. A client-side search tool also allows users to have more control and personalization options during the search process.

The continuous growth of the Web has made it important and urgent to develop vertical search tools to address the information needs of various domains. Developers must carefully choose a suitable approach, based on specific user needs. To analyze and compare the strengths and weaknesses of the two approaches, two prototype systems for the nanotechnology domain — NanoSearch and NanoSpider — have been built and analyzed. In this paper, we compare and discuss the pros and cons of adopting the two approaches. To our knowledge, this paper is the first to compare prototype vertical search tools built by using the two approaches.

The rest of the paper is organized as follows. Section 2 reviews related research and products, including Web search engines, search agents, and post-retrieval analysis. Section 3 describes the architecture of the server-side, Web-based vertical search engine approach and Section 4 describes the architecture of the client-side, meta-search approach. In Section 5, we compare the two approaches and discuss the advantages and disadvantages of each. In Section 6, we conclude our paper and suggest some future research directions.

## 2 RELATED WORK

### 2.1 Web Search Engines

#### *2.1.1 General-purpose Search Engines*

Search engines have been built to help users find what they want on the Web. Most search engines rely on spiders or crawlers to traverse the Web to collect pages by following hyperlinks. One of the earliest search engines is the World Wide Web Worm built in 1994 [24], which indexes 110,000 documents and provides users with a search interface. WebCrawler [25] and Lycos [23] are other examples of early search engines that are still in operation today. While the size of the Web has increased exponentially in recent years, search engines also have grown at a similar scale. As of December 2001, Google (www.google.com) [2], one of the largest search engines, had more than 2 billion pages in its database.

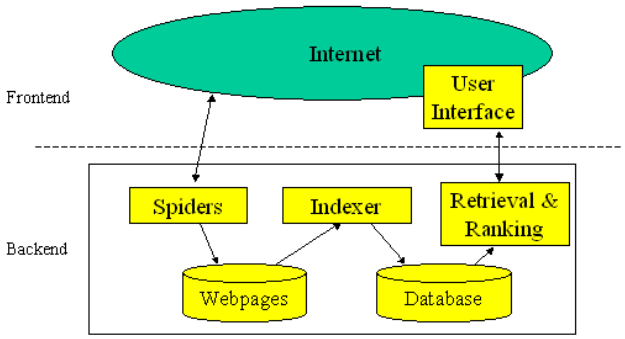


**Figure 1. Typical search engine architecture**

The architecture of a typical search engine is shown in Figure 1. A search engine usually consists of the following components:

- Spiders (a.k.a. crawlers): to collect Web pages from the Internet using different graph search algorithms.
- Indexer: to index Web pages and store the indices into database.

- Retrieval and Ranking: to retrieve search results from the database and present ranked results to users.
- User Interface: to allow users to query the database and customize their searches.

#### *2.1.2 Domain-specific Search Engines*

As the Web continues to grow, general-purpose search engines can no longer satisfy the needs of most users searching for specific information on a given topic. Many domain-specific search engines, also known as vertical search engines, have been built to facilitate more efficient search in particular domains. These search engines usually provide more precise results and more customizable functions. For instance, LawCrawler (lawcrawler.findlaw .com) specializes in searching legal information on the Web, and GolfHelp (www.golfhelp.com) searches for golf-related Web pages. Other examples include SciSeek (www.sciseek.com), BioView (www.bioview.com), and BuildingOnline (www.buildingonline.com).

### 2.2 Personal Web Search Agents

As opposed to a server-side search engine that allows multiple users to submit search queries at a Web site, personal search agents are designed to run search sessions on the user's computer. Many Web software programs based on the concept of spiders, agents or softbots have been developed. tueMosaic is a prominent early example [10]. Using tueMosaic, users can enter keywords, specify the depth and width of search for links contained in the current homepages displayed, and request the spider agent to fetch homepages connected to the current homepage. TueMosaic uses the "fish search" algorithm, a modified best first search method. Since then, more powerful client-side spiders have been developed. For example, Blue Squirrel's WebSeeker (www.bluesquirrel.com) and Copernic 2000 (www. copernic.com) connect with other search engines, monitor Web pages for any changes, and schedule automatic search. Inquirus, also known as the NECI meta search engine, downloads actual result pages and generates a new summary of each page based on the search query. Pages which are no longer available (dead links) or which no longer contain the search terms are filtered from the search results [17]. MetaSpider not only connects with other search engines but also performs categorization on the combined search results [7]. Focused Crawler locates Web pages relevant to a pre-defined set of topics based on example pages provided by the user. In addition, it also analyzes the link structures among the pages collected [3].

### 2.3 Post-retrieval Analysis

In most commercial Web search systems, search results are presented to users as a list of URLs and titles ranked according to the estimated relevance to the user search query. However, such lists do not provide users with a quick "feel" of the documents retrieved. Users know little about a document's content until they click on and read it. This can be very time-consuming and disruptive in a dynamic, fast-changing electronic information environment.

To alleviate the problem, many techniques have been developed to perform analysis on the retrieved results. Applying an automatic indexing algorithm is one technique that has been used widely to extract key concepts from textual data. Linguistics approaches such as noun phrasing have been employed to perform indexing for phrases rather than just words [28]. These techniques are useful in extracting meaningful terms from text documents, not only for document retrieval but also for further analysis.

Categorization, or clustering, is another kind of post-retrieval analysis. The goal is to classify documents into different categories automatically. Categorization tools allow users to quickly identify the key topics in the set of search results returned from a search engine. Vivisimo (www.vivisimo.com) and NorthernLight (www. northernlight.com) are two commercial examples. Both search engines put search results into folders, each of which represents a subtopic.

In document clustering, there are in general two approaches. In the first, documents are categorized based on individual document attributes. An attribute might be the query term's frequency in each document [14, 29]. NorthernLight is an example of this approach. The retrieved documents are organized based on the size, source, topic or author of each document. Other examples include Envision [11] and GRIDL [26].

In the second approach, documents are classified based on inter-document similarities. This approach usually includes some kind of machine learning algorithms. For example, the Self-Organizing Map (SOM) approach classifies documents into different categories that are defined during the process, using a neural network algorithm [16]. Based on this algorithm, the SOM technique automatically categorizes documents into different regions based on the similarity of the documents. It produces a data map consisting of different regions, each of which contains similar documents. Regions that are similar are located close to each other. Several systems utilizing this technique have been built [8, 20].

Text summarization is another post-retrieval analysis technique that helps users understand a set of retrieved documents. The approaches to text summarization vary greatly. A distinction commonly is made between approaches that utilize text extraction and those that utilize text abstraction. Text extraction is the most common approach [15], utilizing sentences from a document to create a summary. Early examples of summarization


**Figure 2. Sample user session with NanoSearch**

techniques were based on text extraction. Using sentences from the text was thought to limit the bias that might otherwise appear in a summary [21]. Text abstraction programs, on the other hand, produce grammatical sentences that summarize a document's concepts. While the formation of an abstract may better fit the idea of a summary, its creation involves greater complexity and difficulty [15]. The AI Summarizer utilizes text extraction.

Various techniques have been used to identify sentences that effectively summarize a document. Luhn utilized word-frequency-based rules in the late 1950's to identify sentences for summaries [21]. Other language-oriented heuristics, such as cue phrases, sentence position, and sentence length were added in the 1960's and early 1970s [22]. Later, Information Retrieval-style methods were used to break documents into segments [13]. From within those segments or topics, sentences were then selected to form summaries.

## 3 A SERVER-SIDE, VERTICAL SEARCH ENGINE APPROACH

In our first approach, we built a server-side Web search engine for the nanotechnology domain called NanoSearch. Similar to a typical vertical search engine, NanoSearch collects pages from the Web, indexes these pages, stores the indexes in a database, and provides users with a Web-based interface to search for nanotechnology-related pages from this database. A sample user session is shown in Figure 2. First, a user inputted the search term(s) through the Web interface (step 1). In the example, "nanomedicine" was used. The search query was then sent to the back-end search engine and the results were displayed to the user (step 2). Much like a typical search engine, the user could click on any result and see the corresponding Web page. The user could summarize the Web page using the built-in text summarizer (step 4). Additionally, the user could request the system to extract the key concepts from these pages using Arizona Noun Phraser and categorize the Web pages into a 2-D topic map using SOM (step 5). In this example, pages were categorized into topics such as "medical nanbots" and "human body".

### 3.1 System Architecture

NanoSearch follows the typical search engine architecture discussed in Section 2.1.1. The main technologies used include HTML, Java, Java Server Page (JSP), Java Bean and Java Servlet. The architecture is shown in Figure 3. The right-hand side of the figure shows the batch process of building the search engine. First, a simple Vertical Spider developed in Java collects Web pages in the nanotechnology domain and calculates the number of inlinks (hyperlinks pointing to a page from other Web sites). The pages collected by the spider are then processed by the Indexer to generate index files. Information in the index files is loaded into an Oracle database automatically by a tool called SQL Loader.

The left-hand side of the figure shows the search process and demonstrates how the search engine interacts with its users. The different components are connected by a Middleware implemented as a JSP file. Search queries from the User Interface are passed to the Middleware and converted into a proper format. The Middleware generates executes SQL statements (through a Java Bean file) to retrieve relevant Web pages from the database. After the retrieval, the results will be returned to the User Interface. If the user wants to perform further analysis on the retrieval set to find more precise results, the Middleware invokes the AI Summarizer or the Self Organizing Map (SOM) through two Java Servlet files.

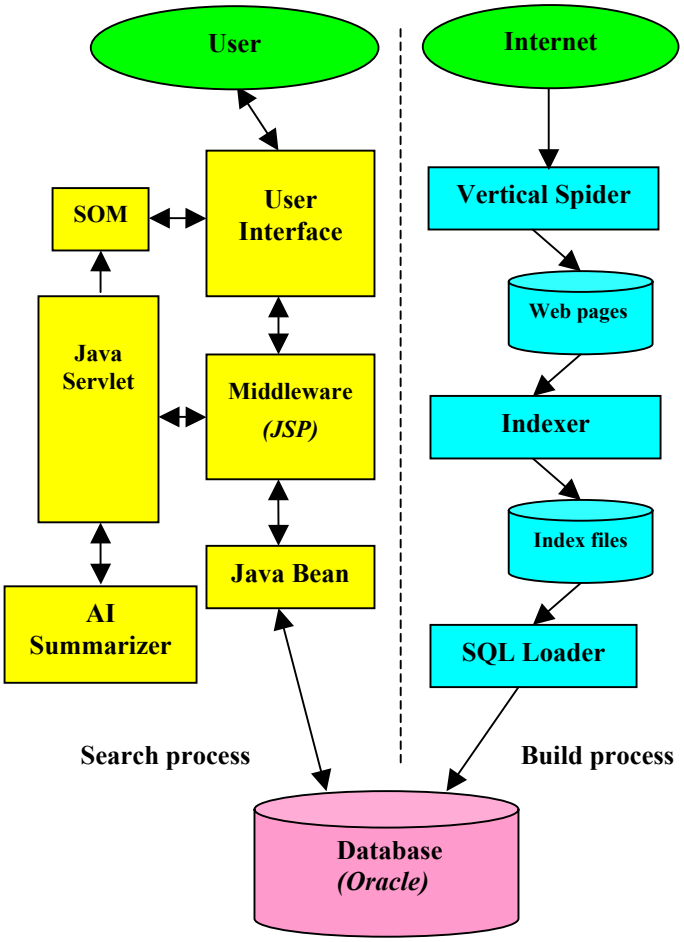


**Figure 3. NanoSearch Architecture**

### 3.2 Vertical Spidering and Indexing

A Vertical Spider program developed in Java by our research group was used in NanoSearch to collect nanotechnology-related pages from the Web. Thirty-three URLs in the nanotechnology domain were manually identified by reviewing a set of pages (and their neighbor pages) obtained from searching the word "nanotechnology"

in Yahoo! and Google. These URLs were used as the "seed URLs" in the spider program. The spider collects pages by following links on each page collected, using a simple breadth-first search algorithm. The spider stops after a specified number of pages have been collected. In our first prototype, the spider downloaded 15,000 pages.

After the pages have been collected, an Indexer program developed in C language is used to index each word's occurrence in each page. The indexer records the frequency that each term appears in each document. The number of inlinks for each page is also calculated. These data are then loaded into the NanoSearch database.

### 3.3 Document Retrieval and Ranking

After a user types in a search term in the User Interface and clicks the search button, the term is passed to the Middleware. The middleware separates the string into single keywords and removes words that are too common (such as "a", "of" and "is") according to a predefined stop-word list. A SQL query to retrieve Web page records containing the search term is generated and executed via a database connection through a Java Bean program. The retrieval results are returned to the Middleware by the Java Bean, which contains basic information such as the URL of the page, the title of the page and the description of the page, as well as information for ranking purpose such as number of inlinks, the frequency of keywords appearing in the page (*tf*) and the inversed document frequency of pages containing the keywords (*idf*). In our system, *idf* is calculated by: $\log_2$(total number of pages / number of pages containing the keywords).

Two factors are considered in the page ranking, namely the relevance of the page content and the popularity of the page. The relevance is represented by *tf*idf* and the popularity is represented by the number of inlinks. In the NanoSearch system, popularity is considered a little more important in ranking than the content relevancy, so a final score is calculated for each page which is 60%*inlink+40%**tf***idf*. The weights are arbitrary and can be easily adjusted. All the results retrieved are sorted according to this score. After ranking, the results are returned to the User Interface and displayed to the user.

In addition to showing the search results as a traditional ranked-list, we also provide users with a graphical way to visualize the search results. For each search result, there is a picture of a flower (called the "NanoFlower") that represents the quality of that particular Web page (see Figure 2, step 2). The size of the flower represents the term frequency and the number of petals represents the number of inlinks. A larger flower indicates that the page is more relevant to the search string, and a flower with more pedals indicates that the page is more popular. The NanoFlower provides an intuitive way for users to find the pages they want.

### 3.4 Post-retrieval Analysis

It is important to extract key concepts from each retrieved document in order to perform post-retrieval analysis. Arizona Noun Phraser (AZNP), developed by our research group, serves this purpose. It extracts all the noun phrases from each document, based on part-of-speech tagging and linguistic rules [28]. AZNP has three components. The tokenizer takes Web pages as text input and creates output that conforms to the UPenn Treebank word tokenization rules by separating all punctuation and symbols from text without interfering with textual content. The tagger module assigns every word in the document a part-of-speech designation. The last module, called the phrase generation module, converts the words and associated part-of-speech tags into noun phrases by matching tag patterns to a noun phrase pattern established by linguistic rules. For example, the phrase "carbon tube" would be considered a valid noun phrase because it matches the rule that a *noun-noun* sequence forms a noun phrase. These phrases are used in both the built-in post-retrieval analysis tools: the AI Summarizer and the Self-Organizing Map (SOM).

A user can summarize any Web page from the retrieval result of NanoSearch using the AI Summarizer. Such summarization can help the user decide quickly whether the page is interesting or not. The AI Summarizer utilizes both segmentation and summarization methods to identify representative sentences for a document. The whole process is accomplished in three main steps: 1) sentence evaluation 2) segmentation / topic boundary identification and 3) segment ranking. The first step, sentence evaluation, involves ranking of all the sentences of the original document based on the existence of cue phrases in the sentence, the position of a sentence in a paragraph, the frequency of terms in a sentence relative to the document, and the existence of proper nouns. In the second step, the document is processed by a segmentation algorithm similar to TextTiling [13]. The TextTiling algorithm analyzes a document and determines where the topic boundaries are located. A topic boundary can be thought of as the place where the author of the document changes subjects or themes. The TextTiling algorithm divides the text into blocks of equal size and these blocks are compared by a similarity function such as the Jaccard's score. A low similarity score between two adjacent blocks indicates a segment or topic boundary. The last step of the summarization process involves ranking segments based on the evaluation of the sentences in each segment. The program then produces a summary of the document by extracting the highest-ranking sentences one by one from the document until the required summary length is reached.

NanoSearch uses SOM to give users an overview of the set of documents collected [16]. SOM employs an artificial neural network algorithm to cluster the Web pages collected into different regions on a 2-D map

automatically. In SOM, each document is represented as an input vector of keywords and a two-dimensional grid of output nodes are created. The distance between the input and each output node is then computed and the node with the minimum distance is selected. After the network is trained through repeated presentation of all inputs, the documents are submitted to the trained network and each region is labeled by the phrase that is the key concept most represents the cluster of documents in that region. More important concepts occupy larger regions, and similar concepts are grouped in a neighborhood [19]. The map is displayed through the User Interface and the user can view the documents in each region by clicking on it.

## 4 A CLIENT-SIDE, META-SEARCH AGENT APPROACH

In our second approach, we implemented a client-side search agent called NanoSpider, based on the MetaSpider system developed in previous research [5, 7]. MetaSpider is a client-side search tool built in Microsoft Visual J++ 6.0, using a combination of Java Classes, Windows Foundation Classes, and Dynamic Link Libraries (DLL). It connects to 6 general-purpose search engines, and processes the combined search results using Arizona Noun Phraser and SOM, in a way similar to that of NanoSearch. In this section, we discuss the architectural design of NanoSpider and the customization made. A sample user session of NanoSpider is shown in Figure 4. A user first input the search term(s) and chose the preferred search engines (step 1). In the example given, the user input the search term "nanocomputer". The search query was then sent to the chosen search engines and the results were combined and shown to the user (step 2). After browsing the search results, the user could request the system to fetch the pages from the Web and extract the key concepts from these pages using Arizona Noun Phraser (step 3). The user could then further categorize the Web pages into a 2-D topic map using SOM (step 4).

### 4.1 System Architecture

The system architecture of NanoSpider is shown in Figure 5. There are 4 main components, namely (1) User Interface, (2) Search Spiders, (3) Arizona Noun Phraser, and (4) SOM. While the architecture is based on that of MetaSpider, these components have been customized to perform Web search and analysis in the nanotechnology domain.

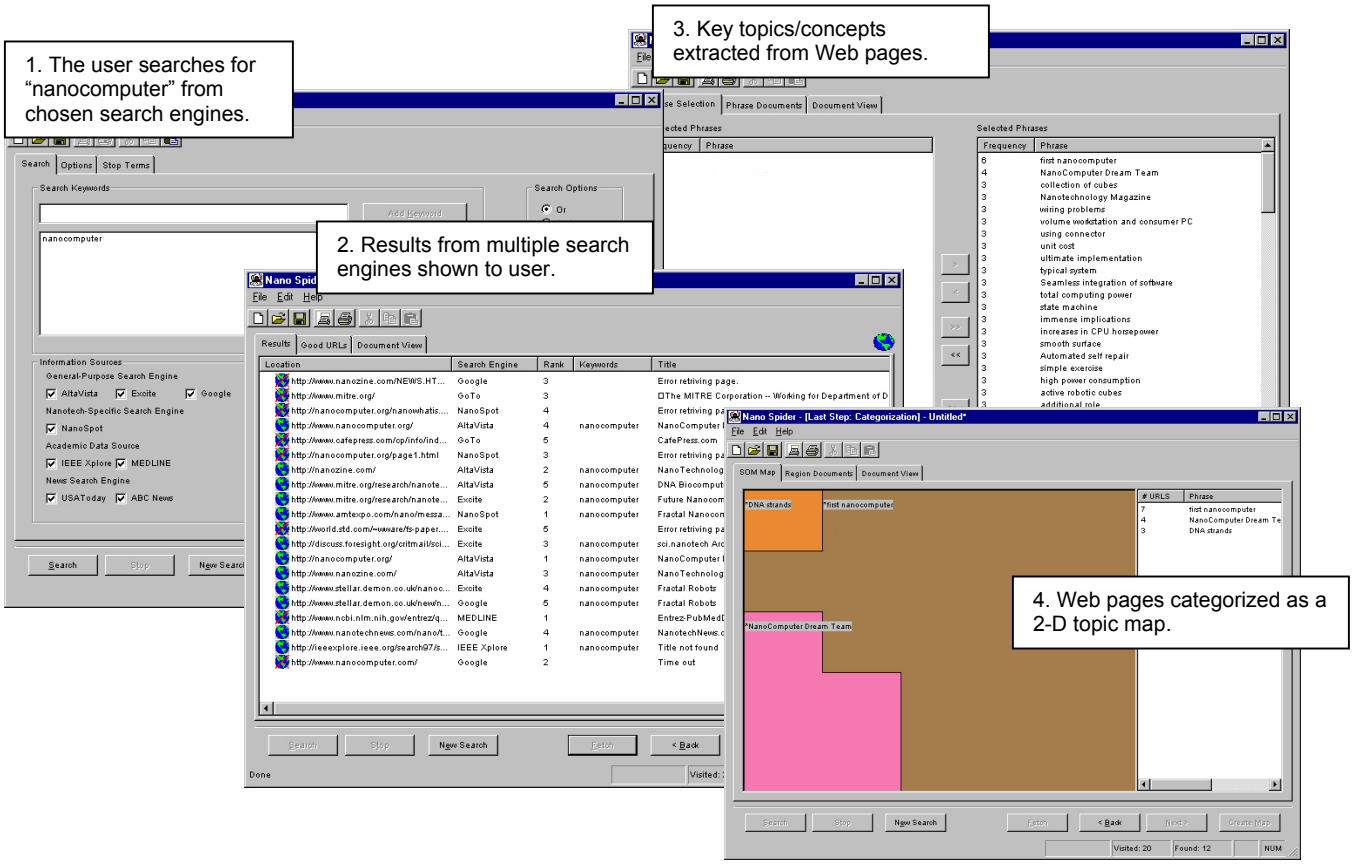


**Figure 4. Sample user session with NanoSpider**

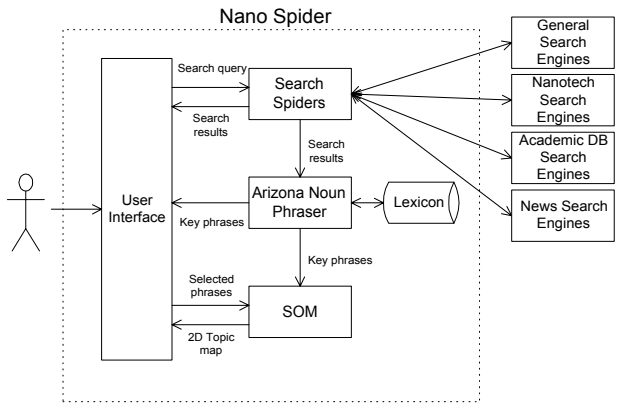


**Figure 5. NanoSpider architecture**

### 4.2 Meta-search

Based on the MetaSpider design, NanoSpider searches multiple search engines and presents the combined results, a process called meta-searching. Search Spiders are developed to perform such meta-searching. The tasks of the Search Spiders include: (1) formulating the URL specific for each search engine based on search query, (2) fetching the search result from each search engine, (3) parsing the search result details (e.g., URL, title, description for each search result) from the result page fetched in HTML format, and (4) analyzing the search results and presenting them to the user. Steps (1) and (3) vary from one search engine to another because each search engine has its own format for querying its database and presenting search results.

While the original MetaSpider is connected only with 6 general-purpose search engines, NanoSpider has a much wider range of data sources. There are four categories of search engines in NanoSpider. The first category is general-purpose search engines (AltaVista, Excite, Google, Goto, Lycos) which locate pages from the whole Web, aiming at high coverage. The second category involved nanotechnology-specific search engines (NanoSpot) and aims at high precision by searching only for pages in the nanotechnology domain. The third category consists of academic literature search engines (IEEE Xplore, MEDLINE) which tries to locate journal articles and abstracts that may not be covered by the other search engines. The fourth category is news search engines (USA Today, ABC News) trying to retrieve the latest, breaking information by searching online news articles. These data sources were added to NanoSpider rather easily, as could many other data sources.

### 4.3 Post-retrieval Analysis

In NanoSpider, we incorporated the Arizona Noun Phraser and the SOM for post-retrieval analysis. The algorithms used are same as those used in NanoSearch as discussed in Section 3.4. After the Search Spiders get the results from the different search engines, the user can browse through the list of search results. If the user wants to perform further analysis on the result set, the system can fetch all the result pages from the Web and send them to the Arizona Noun Phraser, which extracts the key phrases from each Web page. The frequency of every phrase is recorded and sent back to the User Interface. The user can then view the document frequency of each phrase and link to the documents containing that phrase. After all documents are processed, the data are aggregated and sent to the SOM for categorization. A 2-D map is generated and displayed to the user.

### 4.4 Opening and Saving Search Sessions

Another important functionality incorporated in the system is the Save function. A user can save a completed search session as a file in his/her local computer and open it at a later time. This feature allows the user to perform a search and review it later. This also helps users who want to monitor pages in a Web site or on a particular topic.

## 5 COMPARISONS

Both server-side and client-side approaches have been successfully applied to building a vertical search tool in the nanotechnology domain. In this section, we analyze and compare the strengths and weaknesses of the two approaches, based on our experience with the two prototype systems.

### 5.1 Accessibility and Usability

The server-side approach allows users to access the search service more easily through the Web site. Any user with a Web browser can use the search tool conveniently from any platform, such as Windows, Linux, or Macintosh, and no software download or installation is needed. The Web browser interface, through which the server-side search tool is accessed, is also more familiar to users. This increases the usability of the system.

On the other hand, the client-side search tool presents with the issue of installation. A user has to download the installation file from the Web and install the software on his/her own computer. This poses a problem for users with incompatible computers or operating systems. The current NanoSpider only works on MS Windows environment with a MS Windows-specific Java Virtual Machine (a particular version or a later one) installed. As this Java Virtual Machine is MS Windows-specific, users with a Linux or Macintosh computer will not be able to use the system. Using pure Java to build the search tool will alleviate the problem to some extent, but users still need to have some kind of Java environment installed in order to use the tool. Another solution is to bundle a Java Runtime Environment with the software installation package. This will, however, greatly increases the size of the package and thus the download time needed.

### 5.2 Scalability

The Web-based NanoSearch has to deal with users who perform searches at the same time. While some processes, such as spidering and indexing, can be performed in a batch mode, other processes have to be performed on-the-fly (e.g., Arizona Noun Phraser, SOM, and AI Summarizer). Because of the high computational requirements of these modules, it is more difficult to serve a large number of users without increased investment in the server computers and network bandwidth. Although a few computer servers can handle a moderate number of Web pages and search queries, a very large-scale search engine like Google is reported to need more than 6,000 computer servers to collect and index Web pages and to response to search queries.

On the other hand, multiple users can use NanoSpider concurrently because they can have different copies of the software installed on each of their own computers. Thus, the number of users is not constrained by central server processing power or server's network bandwidth.

### 5.3 Personalizability

Both NanoSearch and NanoSpider are easily customizable. In the server-side scenario, a user's settings can be saved to the database and can be retrieved upon the user's next log-on. However, it would be more memory-consuming to store the complete search sessions of all users. Saving such data on the server provides great opportunities for future research such as Web log mining. However, the issue of security and privacy needs to be handled cautiously in the server-based approach. For client-based NanoSpider, a user's settings, as well as any search sessions, can be saved to the user's local computer without requiring any space from a centralized server.

NanoSpider also allows more functionalities because it runs on the user's computer and more processing time and memory can be utilized. Categorization techniques requiring heavy computation can be used on a client-side tool without having to worry about the number of simultaneous users as is the case of server-side search engines.

### 5.4 Maintainability

The server-side tool is easy to modify. Any changes to the algorithm or interface can be incorporated into the NanoSearch in a way transparent to the users. Users can go to the same Web site and enjoy the same (or more likely, improved) service.

The client-side NanoSpider is comparatively more difficult to update, because each user needs to repeatedly download and install the software or a patch whenever a change is made. This makes it difficult to upgrade and maintain the software. Whenever a data source (e.g., AltaVista) changes its query or result display format so dramatically that the NanoSpider cannot handle it without being modified, every user must download and install a new component. Another solution is to embed a component to permit a check for automatic update every time the software is invoked. One example system using this approach is RealPlayer.

## 6 CONCLUSION AND FUTURE DIRECTIONS

In this paper, we report our experience in using two different approaches to building a customized search tool for the nanotechnology domain. We also demonstrate the feasibility of both approaches. As each approach has its strengths and weaknesses, a choice between them will depend on the objective of the tool to be built. The size, the targeted audience, the functionalities, and the quality requirements of the tool will have an impact on the decision.

Our future research plan includes conducting a detailed user experiment to compare the performance, usability, user perception, and level of satisfaction of the two search tools. It is likely that while some users prefer a Web-based service, others prefer a piece of downloadable software.

We plan to test the individual components in order to identify which of the integrated techniques contribute to an improvement in performance if any. We also prepare to conduct user studies to evaluate the visualization techniques (SOM and NanoFlower) used in our system.

Currently, we are expanding NanoSearch into a Web portal called NanoPort [4]. We also plan to study broader educational and social issues related to creating a complete community-based service for nanotechnology engineers and scientists. The search tools will be made available to local student groups as well as to the public through the Internet. We will evaluate how instructors and students interact with the search tools in nanotechnology classes. We also plan to evaluate the social impact of such a search tool on the nanotechnology community.

## 7 ACKNOWLEDGMENTS

This research is supported in part by the following grants:

- NSF/NSE/SGER, "NanoPort: Intelligent Web Searching for Nanoscale Science and Engineering," CTS-0204375, January 2002 – September 2002.
- NSF Digital Library Initiative-2, "High-performance Digital Library Systems: From Information Retrieval to Knowledge Management," IIS-9817473, April 1999 – March 2002.
- NSF/CISE/CSS, "An Intelligent CSCW Workbench: Analysis, Visualization, and Agents" IIS-9800696, June 1998 – June 2001.

We would also like to thank Li Ji, Chip La Clair, and all members of the Artificial Intelligence Lab at the University of Arizona who have contributed to the design and

implementation of the two search tools. Thanks also go to the anonymous reviewers for their invaluable comments.

## 8 REFERENCES

[1] Bowman, C. M., Danzig, P. B., Manber, U., and Schwartz F. Scalable Internet Resource Discovery: Research Problems and Approaches, Communications of the ACM, 37(8) (1994), 98-107.

[2] Brin, S. and Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In Proceedings of the 7th International World Wide Web Conference (WWW7), Brisbane, Australia, Apr 1998.

[3] Chakrabarti, S., van den Berg, M., and Dom B. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. In Proceedings of the 8th International World Wide Web Conference (WWW8), Toronto, Canada, May 1999.

[4] Chau, M., Chen, H., Qin, J., Zhou, Y., Sung, W. K., Chen, Y., Qin, Y., McDonald, D., Lally, A., and Landon, M. NanoPort: A Web Portal for Nanoscale Science and Technology. In Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'02), Portland, OR, USA, July 2002.

[5] Chau, M., Zeng, D., and Chen, H. Personalized Spiders for Web Search and Analysis. In Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'01), Roanoke, VA, USA, June 2001.

[6] Chen, H. "Collaborative Systems: Solving the Vocabulary Problem," IEEE Computer, Special Issue on Computer-Supported Cooperative Work (CSCW), 27(5) (1994), 58-66.

[7] Chen, H., Fan, H., Chau, M., and Zeng, D. MetaSpider: Meta-Searching and Categorization on the Web, Journal of the American Society for Information Science and Technology, 52(13), 1134-1147 (2001).

[8] Chen, H., Schufels, C., and Orwig, R. Internet Categorization and Search: A Self-Organizing Approach, Journal of Visual Communication and Image Representation, 7(1), 88-102 (1996).

[9] Courteau , J. "Genome Databases," Science, 254, (1991), 201-207.

[10] DeBra, P. and Post, R. Information retrieval in the World-Wide Web: Making Client-based Searching Feasible. In Proceedings of the First International World Wide Web Conference, Geneva, Switzerland, 1994.

[11] Fox, E., Hix, D., Nowell, L. T., Brueni, D. J., Wake, W. C., Lenwood, S. H., and Rao, D. Users, User Interfaces, and Objects: Envision, A Digital Library. Journal of the American Society for Information Science, 44(8) (1993), 480-491.

[12] Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. "The Vocabulary Problem in Human-System Communication" Communications of the ACM, 30(11), (1987), 964-971.

[13] Hearst, M. A. TextTiling: Segmenting Text into Multi-paragraph Subtopics Passages. Computational Linguistics, 23(1) (1997), 33-64.

[14] Hearst, M. A. and Pedersen, J. Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results, in Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96), 76-84 (1996).

[15] Hovy, E. and Lin, C. Y. Automated Text Summarization in SUMMARIST. Advances in Automatic Text Summarization, 81-94, MIT Press 1999.

[16] Kohonen, T. Self-Organizing Maps. Springer-Verlag, Berlin, 1995.

[17] Lawrence, S. and Giles, C. L., Inquirus, the NECI Meta Search Engine. In Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, Apr 1998.

[18] Lawrence, S. and Giles, C. L. Accessibility of Information on the Web, Nature, 400 (1999), 107-109.

[19] Lin, C., Chen, H., and Nunamaker, J. Verifying the Proximity and Size Hypothesis for Self-Organizing Maps. Journal of Management Information Systems, 16(3) (1999-2000), 61-73.

[20] Lin, X., Soergel, D., and Marchionini, G. A Self-organizing Semantic Map for Information Retrieval, in Proceedings of the 14th International ACM SIGIR Conference on Research and Development in Information Retrieval (1991), 262-269.

[21] Luhn, H. P. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development 2 (2), 159-165 (1959).

[22] Mani, I. and Maybury, M. T. Advances in Automatic Text Summarization. MIT Press, 1999, ix-xv.

[23] Mauldin, M. L. Lycos: Design Choices in an Internet Search Service. IEEE Expert, 12(1) (1997), 8-11.

[24] McBryan, O. A. GENVL and WWWW: Tools for Taming the Web. In Proceedings of the 1st International World Wide Web Conference, Geneva, Switzerland, 1994.

[25] Pinkerton, B. Finding What People Want: Experiences with the WebCrawler. In Proceedings of the 2nd International World Wide Web Conference, Chicago, IL, USA, 1994.

[26] Shneiderman, B., Feldman, D., Rose, A. and Grau, X. F. Visualizing Digital Library Search Results with Categorical and Hierarchical Axes, in Proceedings of 5th ACM Conference on ACM 2000 Digital Libraries, San Antonio, TX, USA, 2000.

[27] Stix, G. (ed.). Nanotechnology. Scientific America, September 2001 (entire issue).

[28] Tolle, K. M. and Chen, H. Comparing Noun Phrasing Techniques for Use with Medical Digital Library Tools. Journal of the American Society for Information Science, 51(4) (2000), 352-370.

[29] Veerasamy, A. and Belkin, N. J., Evaluation of a Tool for Visualization of Information Retrieval Results. In Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96), 85-92, 1996.