

# 10. Personalized and Focused Web Spiders

Michael Chau and Hsinchun Chen

Department of Management Information Systems  
The University of Arizona  
Tucson, Arizona 85721, USA

## Abstract

As the size of the Web continues to grow, searching it for useful information has become increasingly difficult. Researchers have studied different ways to search the Web automatically using programs that have been known as spiders, crawlers, Web robots, Web agents, Webbots, etc. In this chapter, we will review research in this area, present two case studies, and suggest some future research directions.

## 10.1 Introduction

The number of indexable pages on the World-Wide Web has exceeded 2 billion and is still growing at a substantial rate [10.60]. It has become increasingly difficult to retrieve information on the Web. To address this problem, many programs have been built to automatically retrieve Web pages. These programs have been known by a variety of names: spiders, crawlers, Web robots, Web agents, Webbots, wanderers, and worms, among others. The term “spiders” will be used throughout this chapter.

Web spiders have been defined as “software programs that traverse the World Wide Web information space by following hypertext links and retrieving Web documents by standard HTTP protocol” [10.28]. By broader definition, they can include any software that automatically retrieves Web documents by standard HTTP protocol, either by following hypertext links or other methods. As such they include programs such as metasearch spiders (spiders that connect to different search engines and combine the results) [10.23, 10.76]. In the remainder of this chapter, our discussion accepts the broader definition. Other Web robots such as shopbots [10.34], chatbots or chatterbots [10.87] are generally not considered as spiders. Research in spiders began in the early 1990’s, shortly after the World-Wide Web begin to attract increasing traffic and attention. Wanderer, written in 1993, was claimed to be the first spider for the Web [10.42]. Many different versions of spiders have since been developed and studied. An overview of Web spider research is given in the following subsection.

### 10.1.1 Web Spider Research

In general, Web spider research directions can be classified into the following categories:

1. *Speed and efficiency*. In this category, researchers study different ways to increase the harvest speed of a spider. These projects focus on building fast spiders that can be scaled up to large collections by applying program-optimization

techniques to operations such as I/O procedures and IP address lookup. Mercator [10.46, 10.47], Internet Archive's crawler [10.13, 10.50], and Google's crawler [10.11] are some examples. Currently, sophisticated spiders can download more than 10 million documents per day on a single workstation.

2. *Spidering policy.* Research in this category studies the behaviors of spiders and their impacts on other individuals and the Web as a whole. A well-designed, "polite" spider should avoid overloading Web servers [10.38]. Also, Webmasters or Web page authors should be able to specify whether they want to exclude particular spiders' access. There are two standard ways. The first one, called the robot exclusion protocol, allows Web site administrators to indicate, by specifying a file named *robots.txt* in the Web site's root directory, which parts of their site should not be visited by a robot [10.53]. In the second method, usually known as the robots META tag, Web page authors can indicate to visiting robots whether a document may be indexed, or used to extract more links [10.63]. Although these standards are not strictly enforced, most commercial spiders are reported to follow them. Some studies survey the use of these standards in Web sites and investigate their potential impacts [10.35].
3. *Information retrieval.* Most Web spider research fall into this category. These studies investigate how different spidering algorithms and heuristics can be used such that spiders can retrieve relevant information from the Web more effectively. Many of these studies apply to Web spiders techniques that have been shown to be effective in traditional information retrieval applications, e.g., the vector space model [10.75]. In this chapter, we focus mainly on research in this category.

### 10.1.2 Applications of Web Spiders

Spiders have been shown to be useful in various Web applications. There are four main areas where spiders have been widely used:

1. *Personal search.* Personal spiders try to search for Web pages of interest to a particular user. Because these spiders usually run on the client-machine, more computational power is available for the search process and more functionalities are possible [10.17]. This will be discussed in more detail in Sect. 11.2.
2. *Building collections.* Web spiders have been extensively used to collect Web pages that are needed to create the underlying index of any search engine [10.11, 10.47, 10.62, 10.72]. In addition to building search engines, spiders can also be used to collect Web pages that are later processed to serve other purposes. For example, Bharat and Broder used a spider to crawl the Yahoo hierarchy to create a lexicon of 400,000 words; Henzinger et al. [10.8, 10.45] used Mercator to do random URL sampling from the Web; many others have used spiders to collect email addresses or resumes from the Web. More details on this type of spider will be presented in Sect. 11.3.
3. *Archiving.* A few projects have tried to archive particular Web sites or even the whole Web [10.50]. To meet the challenge of the enormous size of the Web,

fast, powerful spiders are developed and used to download targeted Web sites into storage tapes.

4. *Web statistics*. The large number of pages collected by spiders is often used to provide useful, interesting statistics about the Web. Such statistics include the total number of servers on the Web, the average size of a HTML document, or the number of URLs that return a 404 (page not found) response. These statistics are useful in many different Web-related research projects and many spiders have been designed primarily for this purpose [10.12, 10.42].

### 10.1.3 Analysis of Web Content and Structure

There has been much research on different ways of representing and analyzing the content and structure of the Web, which are very important to Web spiders that may need to rely on such information to guide their searches. In this section, different analysis techniques that are relevant to Web spider research will be reviewed. In general, Web analysis techniques can be classified into 2 main categories: (1) content-based approaches, and (2) link-based approaches.

In content-based approaches, the actual HTML content of a Web page is analyzed to induce information about the page. For example, the body text of a Web page can be analyzed to determine whether the page is relevant to a target domain. Indexing techniques can be used to extract the key concepts that represent a page. In addition, the relevance of a page can often be determined by looking at the title. Words and phrases that appear in the title or headings in the HTML structure are usually assigned a higher weight [10.11, 10.15].

Domain knowledge also can be incorporated into an analysis to improve the results. For example, words can be checked against a list of domain-specific terminology. A Web page containing words that are found in the list can be considered more relevant.

The URL address of a Web page often contains useful information about the page. For example, from the URL

“<http://ourworld.compuserve.com/homepages/LungCancer/>”,

we can tell that the URL comes from the domain *compuserve.com*, and that it is likely to be related to the topic *Lung Cancer*. We also know that this page comes from a *.com* site, which may be considered less authoritative than pages from a *.gov* site. Some metrics also consider URLs with fewer slashes more useful than those with more slashes [10.2].

Link-based approaches have attracted increasing attention in recent years as Web link structure has come to be used to infer important information about pages. The basic assumption is that if the author of a Web page A places a link to a Web page B, he or she believes that B is relevant or similar to A, or of good quality [10.44]. We use the term *in-links* to indicate the hyper-links pointing to a given page. Usually, the larger the number of in-links, the better a page is considered to be. The rationale is that a page referenced by more people is likely to be more important than a page

that is seldom referenced. This is similar to citation analysis, in which an often-cited article is considered better than one never cited.

Link analysis has been used in more and more applications in recent years. Link analysis techniques were first used to guide searching in spider applications [10.30, 10.73, 10.79]. Focused Crawler [10.16] and HyPursuit [10.86] use hyperlinks information to enhance Web page classification and clustering respectively. Link analysis has also been applied to identify cyber communities on the Web (groups of individuals who share a common interest, together with the Web pages most popular amongst them) [10.39, 10.54].

By analyzing the pages containing a link of interest, it is also possible to obtain the anchor text that describes the link. Anchor text is the underlined, clickable text of an outgoing link in a Web page. Anchor text may provide a good description of the target page because it represents other people's actual description of the page. Several studies have tried to make use of anchor text or any text nearby to predict the content of the target page [10.1, 10.3]. Some studies also analyze the text that appears near a hyper-link [10.74].

In addition, it is reasonable to give a link from an authoritative source (such as Yahoo) a higher weight than a link from an unimportant personal homepage. Researchers have developed several algorithms to address this issue. Among these, PageRank [10.11] and HITS [10.51] are the two most widely used.

The PageRank algorithm computes a page's score by weighting each in-link to the page proportionally to the quality of the page containing the in-link [10.11]. The quality of these referring pages also are determined by PageRank. Thus, the PageRank of a page  $p$  is calculated recursively as follows:

$$PageRank(p) = (1 - d) + d \times \sum_{\forall q \text{ pointing to } p} \left( \frac{PageRank(q)}{c(q)} \right) \quad (10.1)$$

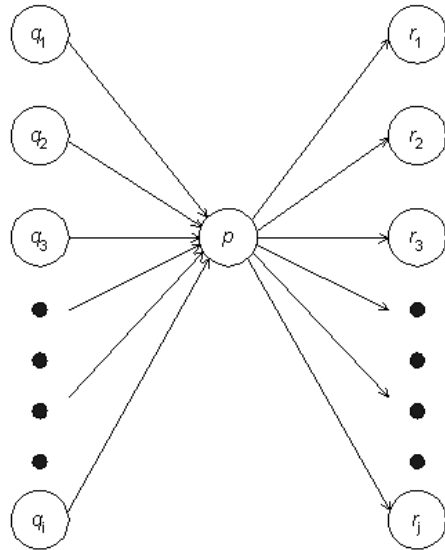
where

$d$  is a damping factor between 0 and 1,  
 $c(q)$  is the number of outgoing links in  $q$ .

Intuitively, a Web page can have a high PageRank if the page is linked from many other pages, and the scores will be even higher if these referring pages are also good pages (pages that have high PageRank scores). This is illustrated in Fig. 11.1. It is also interesting to note that the PageRank algorithm follows a random walk model - the PageRank of a page is proportional to the probability that a random surfer clicking on random links will arrive at that page.

The PageRank algorithm, applied in the commercial search engine Google, has been shown to be very effective for ranking search results [10.11]. Computation time, however, is a major problem in using PageRank. The PageRank score of each Web Page has to be calculated iteratively, making it computationally expensive [10.43].

Kleinberg [10.51] proposed the HITS (hyper-link-induced topic search) algorithm which is similar to PageRank. In the HITS algorithm, authority pages are de-



**Fig. 10.1.** PageRank and HITS: The PageRank score of a page  $p$  depends on the PageRank scores of pages pointing to  $p$  ( $q_1$  to  $q_i$ ). In the HITS algorithm, the Authority score of a page  $p$  depends on the Hub scores of pages pointing to  $p$  ( $q_1$  to  $q_i$ ); the Hub score of a page  $p$  depends on the Authority scores of the pages  $p$  is pointing to ( $r_1$  to  $r_j$ )

finer as high-quality pages related to a particular topic or search query. Hub pages are those that are not necessarily an authority themselves but provide pointers to other authority pages. A page to which many others point should be a good authority, and a page that points to many others should be a good hub. Based on this intuition, two scores are calculated in the HITS algorithm for each Web page: an authority score and a hub score, as illustrated in Fig. 11.1. They are calculated as follows:

$$AuthorityScore(p) = \sum_{\forall q \text{ pointing to } p} HubScore(q), \quad (10.2)$$

$$HubScore(p) = \sum_{\forall r \text{ being pointed by } p} AuthorityScore(r). \quad (10.3)$$

A page with a high authority score is one pointed to by many hubs, and a page with a high hub score is one that points to many authorities. One example that applies the HITS algorithm is the Clever search engine [10.14], which achieves a higher user evaluation than the manually compiled directory of Yahoo. The Teoma search engine also uses a similar algorithm in its ranking process. Bharat and Henzinger [10.9] have added several extensions to the basic HITS algorithm, such as regulating how much a node, based on its relevance, influences its neighbors. Similarly to the PageRank algorithm, a drawback of the HITS algorithm is its high com-

putational requirement, because the hub and authority scores have to be calculated iteratively.

#### 10.1.4 Graph Traversal Algorithms

Traditional graph search algorithms have been extensively studied in the field of computer science. Since most researchers view the Web as a directed graph with a set of nodes (pages) connected with directed edges (hyper-links), some of these algorithms have been applied in Web applications. In this section, we review three categories of graph search algorithms that are relevant to our study, namely, (1) uninformed search, (2) informed search, and (3) parallel search [10.71, 10.90].

The first category of graph search algorithms consists of simple algorithms such as breadth-first search and depth-first search. They are also known as *uninformed search* as they do not make use of any information to guide the search process. Breadth-first search is one of the most popular methods used in Web search spiders that collect all pages on the current level before proceeding to the next level. Although these algorithms are easy to implement and use in different applications, they are usually not very efficient because of their simplicity.

The second category is *informed search*, in which some information about each search node is available during the search process. Such information is used as the heuristics to guide the search. Best-first search is one example that is widely used. Best-first search explores the most promising node at each step. This class of algorithms has been studied in different search engine spiders or search agent systems with different variations [10.21, 10.30]. Different metrics, such as number of in-links, PageRank score, keyword frequency, and similarity to search query, have been used as guiding heuristics.

Another category is *parallel search*. Algorithms in this category try to explore different parts of a search space in parallel. One example is the spreading activation algorithm used in artificial neural network models, which tries to achieve human-like performance by modeling the human nervous system. A neural network is a graph of many active nodes (neurons) that are connected by weighted links (synapses). A neural network uses spreading activation over the nodes to represent and retrieve concepts and knowledge [10.5, 10.25, 10.55]. Another example is genetic algorithms, which increasingly have been used in optimization problems such as financial portfolio optimization and resource allocation. Genetic algorithm is an evolutionary process designed to search for an optimal solution through crossover and mutation operations [10.66]. Gordon [10.41] provides a model for using genetic algorithms in textual analysis. Chen et al. [10.21] extended the model and used it in a Web spider. Although these algorithms are powerful and have been used in traditional information-retrieval applications [10.19], they have not been widely applied in Web applications.

## 10.2 Web Spiders for Personal Search

Many Web spiders have been developed to help individual users search for useful information on the Web. Because these spiders usually run on the client machine, more CPU time and memory can be allocated to the search process and more functionalities are possible. These tools also allow users to have more control and personalization options during the search process.

### 10.2.1 Personal Web Spiders

tueMosaic is a prominent early example of personal Web spiders [10.32]. 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 its introduction, many more powerful personal spiders have been developed.

Some spiders have been designed to provide additional functionalities. The TkWWW robot is a program integrated in the TkWWW browser [10.80]. It can be dispatched from the browser and search Web neighborhoods to find relevant pages and returns a list of links that look promising. SPHINX, a spider written in Java, allows users to perform breadth-first search and view the search results as a 2-dimensional graph [10.67]. CI Spider performs linguistic analysis and clustering of the search results [10.20]. Collaborative Spider, an extended version of CI Spider, is a multi-agent system designed to improve search effectiveness by sharing relevant search sessions among users [10.18].

In other studies, spiders use more advanced algorithms during the search process. The Itsy Bitsy Spider searches the Web using a best-first search and a genetic algorithm approach [10.21, 10.22]. Each URL is modeled as an individual in the initial population. Crossover is defined as extracting the URLs that are pointed to by multiple starting URLs. Mutation is modeled by retrieving random URLs from Yahoo. Because the genetic algorithm is an optimization process, it is well suited to finding the best Web pages according to particular criteria. Webnaut is a later spider that also applies genetic algorithms [10.70]. Other advanced search algorithms also have been used in personal spiders. Yang et al. [10.91] apply hybrid simulated annealing in a personal spider application. 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 Web pages collected [10.16]. Context Focused Crawler uses a Naive Bayesian classifier to guide the search process [10.33]. Relevance feedback has also been applied in spiders [10.4, 10.84].

Many commercial Web spiders are also available. WebRipper, WebMiner, and Teleport are examples of software that help users download specified files from given Web sites. Excalibur’s RetrievalWare and Internet Spider collect, monitor and index information from text documents on the Web as well as graphic files. Autonomy’s products support a wide range of information collection and analysis tasks,

which include automatic searching and monitoring information sources in the Internet and corporate Intranets, and classifying documents into categories predefined by users or domain experts. Verity's knowledge-management products, such as Agent Server, Information Server and Intelligent Classifier, also perform similar tasks in an integrated manner.

Another important category of personal spiders is composed of meta spiders, programs that connect to different search engines to retrieve search results. Lawrence and Giles [10.58] showed that the best search engine covered only about 16% of Web sites in 1999. Therefore, combining results from different search engines achieves more comprehensive coverage. MetaCrawler was the first meta spider reported [10.76, 10.77]. It provides a single interface allowing users to search simultaneously from six different search engines, namely Lycos, WebCrawler, Infoseek, Galaxy, Open Text, and Yahoo. MetaCrawler, with much more search options available, is still in service now.

Following the success of MetaCrawler, many metasearch services have been developed. Profusion allows users to choose among a list of six search engines that can be queried [10.40]. 37.com connects with 37 different search engines. Dogpile provides metasearch service for news, Usenet, white pages, yellow pages, images, etc., in addition to Web pages. SavvySearch connects with a large number of general and topic-specific search engines and selects those likely to return useful results based on past performance [10.49]. Similarly, Yu et al. [10.92] use link analysis to decide which are the appropriate search engines to be used, and Chen and Soo [10.27] use domain ontology in meta search agents to assist users in query formulation. Blue Squirrel's WebSeeker and Copernic's software connect with other search engines, monitor Web pages for any changes, and schedule automatic search. Grouper, an extension of MetaCrawler, clusters the search results from various search engines based on a suffix-tree clustering algorithm [10.93].

In addition to getting a list of URLs and summaries returned by other search engines, some meta spiders download and analyze all the documents in the result set. Inquirus, also known as the NECI metasearch engine, downloads actual result pages and generates a new summary of each page based on the search query. Pages that are no longer available (dead links) or which no longer contain the search terms are filtered from the search results [10.56, 10.57]. Meta Spider provides the same functionalities as Inquirus, but also performs linguistic analysis and clustering of the search results [10.23]. Another similar system is TetraFusion, which performs hierarchical and graph-based classification on the result set [10.31]. Focused Crawler [10.16] and Fetuccino [10.6] use search results from popular search engines and expand the result set based on these URLs. Dwork et al. [10.36] proposed the use of a Markov chain to combine search results from different search engines and achieved promising experimental results.

Recently, search spiders have also been developed on the basis of peer-to-peer (P2P) technology, following the success of other P2P applications such as Napster. JXTA Search, formerly known as InfraSearch, uses Gnutella as its backbone and links to other computers when a request is received from a user. The request is



passed to neighboring computers to see if any of them can fulfil the request. Each computer can have its own strategy on how to respond to the request [10.85].

### 10.2.2 Case Study

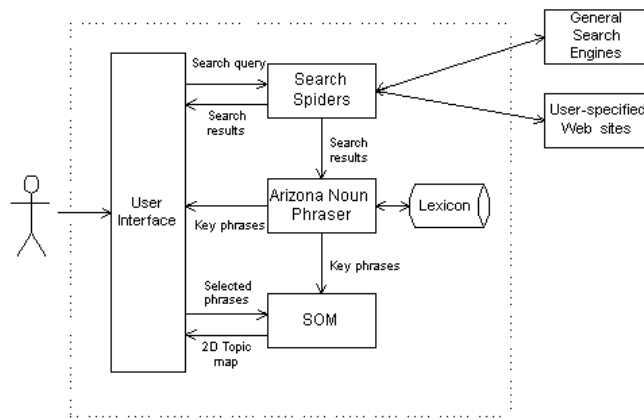
In this section, we present the architecture of two search agents enhanced with postretrieval analysis capabilities. Competitive Intelligence Spider, or CI Spider, is a search agent that collects Web pages on a real-time basis from Web sites specified by the user and performs indexing and categorization analysis on them, to provide the user with a comprehensive view of the Web sites of interest [10.20]. The second tool, Meta Spider, has functionalities similar to those of the CI Spider but, instead of performing breadth-first search on a particular Web site, connects to different search engines on the Internet and integrates the results [10.23]. The architecture of CI Spider and Meta Spider is shown in Fig. 11.2. There are 4 main components, namely (1) user interface, (2) Internet spiders, (3) Arizona noun phraser, and (4) self-organizing map (SOM). These components work together as a unit to perform Web search and analysis. The Arizona noun phraser, developed at the University of Arizona, is the indexing tool used to index the key phrases that appear in each document collected from the Web by the Internet spiders. It extracts all the noun phrases from each document based on part-of-speech tagging and linguistic rules [10.83].

The SOM employs an artificial neural network algorithm to automatically cluster the Web pages collected into different regions on a 2-dimensional map [10.24, 10.26, 10.52]. Each document is represented as an input vector of keywords and a 2-dimensional grid of output nodes is created. After the network is trained, the documents are submitted to the network and clustered into different regions. Each region is labeled by the phrase that is the key concept most accurately representing the cluster of documents in that region. More important concepts occupy larger regions, and similar concepts are grouped in a neighborhood [10.59]. The map is displayed through the user interface and the user can view the documents in each region by clicking on it.

Two separate experiments have been conducted to evaluate CI Spider and Meta Spider. The experimental tasks were designed to permit evaluation of how a combination of their functionalities performed in identifying the major themes related to a certain topic being searched. Thirty undergraduate subjects were recruited to participate in each experiment.

In the first experiment, CI Spider was compared with Lycos and manual “within-site” browsing and searching. Our experimental results showed that both the precision and recall rates for CI Spider were significantly higher than those of Lycos at a 5% significance level. CI Spider’s usability also achieved a statistically higher value than those of Lycos and within-site browsing and searching.

In the second experiment, Meta Spider was compared with MetaCrawler and NorthernLight. In terms of precision, Meta Spider performed better than either of these, and the difference from NorthernLight was statistically significant. Meta Spider’s recall rate was comparable to that of MetaCrawler and better than that of NorthernLight.



**Fig. 10.2.** Architecture of CI Spider and Meta Spider

We suggest that the main reason for the high precision rate of CI Spider and Meta Spider is their ability to fetch and verify the content of each Web page in real time. This means these two spiders can ensure that every page shown to the user contains the keyword being searched. On the other hand, indexes in Lycos and NorthernLight, like those of most other search engines, were often outdated. The high recall rate of CI Spider is mainly attributable to its exhaustive searching characteristic. Lycos showed the lowest recall rate because, like most other commercial search engines, it samples only a number of Web pages in each Web site, thereby missing other pages that contain the search keyword. A user performing manual within-site browsing and searching is likely to miss some important pages because the process is mentally exhausting. Many subjects also commented that they liked the postretrieval capabilities of the Arizona noun phraser and the SOM.

### 10.3 Using Web Spiders to Create Specialized Search Engines

Use of Web search engines such as Google, AltaVista, NorthernLight, Excite, Lycos, and Infoseek is the most popular way to look for information on the Web. Many users begin their Web activities by submitting a query to a search engine. All these search engines rely on spiders to collect Web pages that are then processed to create their underlying search indexes. Examples include AltaVista's Scooter, Google's Googlebot, Lycos's T-Rex, and Excite's Architext.

Search engine spider research began as early as 1994 with World Wide Web Worm, the first widely used search engine spider that indexed only a document's title and header [10.64]. The repository-based software engineering (RBSE) Spider was the first spider to have full-text indexing capabilities [10.37]. Soon after, many full-text indexing spiders were developed, including WebCrawler [10.72], Lycos [10.62],

and Harvest [10.10]. All these spiders follow a simple breadth-first search algorithm that is still widely used now by the spiders behind most major commercial general-purpose search engines to crawl the Web. Some research has also studied the use of an incremental spider that tries to collect only Web pages that have changed [10.29].

### 10.3.1 Specialized Search Engines

Because of the enormous size of the Web, general-purpose search engines can no longer satisfy all the needs of those who are searching for more specific information. Many specialized search engines have been built to address various problems. These search engines specialize in particular Web site(s), topics (such as computers or medicine), languages (such as Chinese or Japanese), file types (such as images or research papers), and so on. Because their size is more manageable (much smaller than the entire Web), these search engines usually provide more precise results and more customizable functions. For instance, BuildingOnline specializes in searching in the building industry domain, CollegeBot searches for educational resources, and LawCrawler specializes in searching for legal information on the Web. There are also content-type-specific search engines. For example, DejaNews searches for news articles, and WebSeek searches for image files [10.78].

Like general-purpose search engines, these search engines rely on spiders to collect Web pages. This task is relatively easy for site-specific search engines, in which spiders can be restricted to downloading only Web pages with a given domain name in the URL [10.61, 10.82, 10.88, 10.89]. For example, spiders can be instructed to discard any URL not starting with the string “http://www.arizona.edu/”. However, the task becomes more difficult for other specialized search engines in which the spiders need to address two main problems:

1. The spiders need to identify from a list of unvisited URLs the ones most likely to contain relevant information. To improve efficiency, a spider should first visit such Web pages.
2. For each downloaded document, the spiders need to determine its relevance according to a specific purpose. The spiders should avoid irrelevant or poor-quality documents by determining the quality and reputation of each document.

This kind of search engine spider sometimes is also known as a focused spider or a focused crawler. In recent years, different focused search engine spiders have been developed and evaluated. We next will focus on research investigating the use of efficient spidering algorithms that aim to address the first problem.

### 10.3.2 Focused Spidering Algorithms for Specialized Search Engines

Despite its simplicity, breadth-first search is widely used in specialized search engines primarily because it is easy to implement and fast to execute. Intuitively, if a URL is relevant to a target domain it is likely that the Web pages in its neighborhood are also relevant. It has been shown that breadth-first search can discover

high-quality pages early on in a spidering process. As the most important pages have many links pointing to them from numerous hosts, those links usually can be found early in the search process [10.68]. Many people choose to use free tools, such as WebGlimpse [10.61], Alkaline, ht://dig [10.82], and GreenStone [10.88], to build specialized search engines. While they work well for building site-specific search engines, it is more difficult to use them for building topic-specific search engines because there is no heuristic for locating and identifying relevant Web pages. To alleviate this problem, users are usually allowed to specify the maximum depth that a spider should explore [10.61, 10.81]. However, this tactic is usually inadequate and results in collections that are too diverse for specific topics.

Best-first search is another widely used algorithm in focused spiders. Depending on the application, different heuristics can be used, such as keyword frequency, similarity to starting examples, number of in-links, or PageRank score. Cho et al. [10.30] have shown that in an experiment on the Stanford Web site, a best-first search spider using PageRank score performed better than a breadth-first search spider or a best-first search spider using the number of in-links. Chakrabarti et al. [10.16] combined similarity and link analysis in Focused Crawler, which visits URLs based on each page's probability of having relevant content.

Machine learning techniques also have been applied in search engine spiders. McCallum et al. [10.74] used reinforcement learning to design a spider that traverses the Web based on immediate and future reward as measured in terms of Web page relevance. That spider was used in Cora, a computer science research paper search engine [10.65].

### 10.3.3 Case Study

Seeking to combine different Web content and structure analysis techniques with traditional graph search techniques to build spider programs for vertical search engines, we developed and compared three versions of Web spiders, namely, (1) Breadth-First Search Spider, (2) PageRank Spider, and (3) Hopfield Net Spider. In this section, we describe the designs and approaches adopted in our study.

The Breadth-First Search Spider (or BFS Spider) collects all Web pages on the current level before proceeding to the next level. In other words, it visits URLs based on the order in which they are discovered. This is implemented using a first-in-first-out queue like that generally used in breadth-first search applications. It runs until the required number of pages are collected.

The PageRank Spider was adapted from the algorithm reported in [10.30]. Aiming to combine link-based analysis and a heuristics-based traversal algorithm, it was designed to perform best-first search using PageRank (as described earlier) as its heuristics. URLs with higher PageRank scores are to be visited earlier.

In each step, the spider gets the URL with the highest PageRank score, fetches the content, and extracts and enqueues all the outgoing links in the page. The process runs until the required number of pages have been collected. PageRank score is calculated iteratively using the algorithm described earlier until convergence is reached. The damping factor  $d$  is set to 0.90 in our implementation. The hot queue

approach used in the original study has also been adopted in the PageRank Spider for anchor text analysis. Two priority queues are established: *hot\_queue* and *normal\_queue*. The URLs within each queue are ordered by PageRank score in descending order. The spider first dequeues from the *hot\_queue*. If the *hot\_queue* is empty, the spider dequeues from the *normal\_queue*. In our design, a URL will be placed in the *hot\_queue* if the anchor text pointing to this URL contains a relevant term.

In the Hopfield Net Spider, the Web is viewed as a large network structure of massive, distributed knowledge composed of pages and hyperlinks contributed by all Web page authors. This can be viewed as a neural network, in which nodes are represented by pages and links are simply represented by hyperlinks. In this approach we model the Web as a Hopfield Net, which is a single-layered, weighted neural network [10.48]. Nodes are activated in parallel and activation values from different sources are combined for each individual node until the activation scores of nodes on the network reach a stable state (convergence).

Based on this spreading activation algorithm, which has been shown to be effective for knowledge retrieval and discovery in a Hopfield Net, we developed the Hopfield Net Spider to perform searching on the Web. In this approach, we aimed to combine a parallel search algorithm with content-based and link-based analysis. Our implementation incorporated the basic Hopfield Net spreading activation idea, but significant modification was made to take into consideration the unique characteristics of the Web.

The Hopfield Net Spider starts with a set of seed URLs represented as nodes, and then activates neighboring URLs, combines weighted links, and determines the weights of newly discovered nodes. The process repeats until the required number of URLs have been visited. The algorithm adopted is as follows:

1. *Initialization with Seed URLs*. An initial set of seed URLs is given to the system and each of them is represented as a node with a weight of 1.  $\mu_i(t)$  is defined as the weight of node  $i$  at iteration  $t$ .

$$\mu_i(0) = 1 \text{ for all seeds URLs } i. \quad (10.4)$$

The spider fetches and analyzes these seed Web pages in iteration 0. The new URLs found in these pages are added to the network.

2. *Activation, Weight Computation, and Iteration*. Proceeding to the next iteration, the weight of each node is calculated as follows:

$$\mu_i(t+1) = f_s\left(\sum_{\forall \text{ known parent } h \text{ of } i} \omega_{h,i} \mu_h(t)\right), \quad (10.5)$$

where

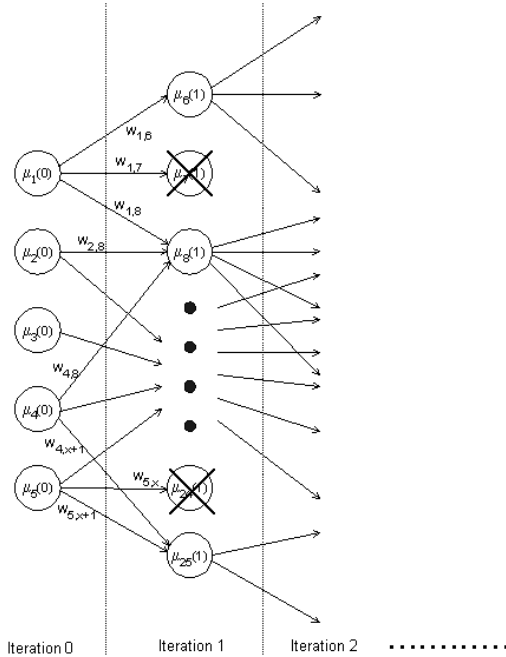
$\omega_{h,i}$  is the weight of the link between two nodes,

$f_s$  is a SIGMOID transformation function that normalized a weight to a value between 0 and 1.

Because  $\omega_{h,i}$  is the weight of the linkage between two URLs, it tries to estimate whether a URL  $i$  pointed to from a Web page  $h$  is relevant to the target

domain, based on the use of anchor text in  $h$ . This score is calculated as a function of the number of anchor-text words relevant to the target domain. We use a slightly modified SIGMOID function to make sure the resulting value is a positive number.

After the weights of all the nodes in the current iteration have been calculated, the spider needs to decide which node (URL) should be activated (visited) first. As the weights decide the order in which URLs are to be visited, they are critical to the effectiveness of the algorithm. The set of nodes in the current iteration are then visited and fetched from the Web in descending order of weight. In order to filter out low-quality URLs, nodes with a weight smaller than a threshold  $\theta$  are not visited. The activation process is illustrated in Fig. 11.3.



**Fig. 10.3.** Spreading activation: Starting with a set of seed URLs, the Hopfield Net Spider activates neighboring URLs, combines weighted links, and determines the weights of newly discovered nodes. Nodes with a low weight (e.g., node 7 and node 24) are discarded

After all the pages with a weight greater than  $\theta$  have been visited and downloaded, the weight of each node in the new iteration is updated to reflect the quality and relevance of the downloaded page content as follows:

$$\mu_i(t + 1) = f_s(\mu_i(t + 1) \times p_i) \tag{10.6}$$

where

$p_i$  is a weight that represents the relevance of the textual content of a page  $i$ .

The  $p_i$  score is a function of the number of phrases found in a page's content that are relevant to the target domain. A page with more relevant phrases will receive a higher score. Phrases can be extracted from each page using the Arizona Noun Phraser [10.83].

3. *Stopping Condition.* The above process is repeated until the required number of Web pages have been collected or until the average weight of all nodes in an iteration is less than a maximum allowable error (a small number).

The three different approaches were implemented as the backend spiders for a medical search engine called HelpfulMed. A simulation experiment and a user study were carried out to evaluate the precision and execution time of the three spiders. The experiment results show that the Hopfield Net Spider had significantly better precision than both the BFS Spider and the PageRank Spider at the 1% level. The BFS Spider also did significantly better than the PageRank Spider at the 1% level. For execution time, we found that the Hopfield Net Spider used slightly more time than the BFS Spider, while the PageRank Spider spent almost 90 times more execution time than the other two spiders.

## 10.4 Conclusions

Over the past decade, Web spiders have evolved from simple breadth-first search spiders to intelligent, adaptive spiders. At the same time, the size of the Web has also grown by more than 250,000 times, from 130 Web hosts in June 1993 to more than 38,000,000 hosts in February 2002 [10.42, 10.69]. The content on the Web has also become more diverse in terms of topic, language, file type, encoding method, and so on, with many dynamically generated Web pages. Locating desired information on the Web is still not easy, despite the availability of various search spiders and search engines.

Spiders can be improved and extended in several ways:

- Currently, most spiders can index only static Web pages. As the amount of dynamic content on the Web increases, spiders need to be able to retrieve and manipulate dynamic content autonomously.
- Spiders can perform better indexing by applying computational linguistic analysis to extract meaningful entities rather than mere keywords from Web pages. This will become a more interesting issue as the Semantic Web [10.7] becomes more mature.
- As the quality and credibility of Web pages vary considerably, spiders need to use more advanced intelligent techniques to distinguish between good and bad pages.

- An ideal personal spider should behave like a human librarian who tries to understand and answer user queries through an autonomous or interactive process using natural language.

As we have witnessed, state-of-the-art search services such as WebCrawler and Lycos that were introduced less than a decade ago have been surpassed by services such as Google that utilize newer algorithms. As the Web continues to evolve, spiders and search engines also must evolve in order to accommodate the size and dynamics of the Web.

## Acknowledgement

The CI Spider, Meta Spider, and HelpfulMed projects were funded in part by the following grants:

- NSF Digital Library Initiative-2 (PI: H. Chen), “High-performance Digital Library Systems: From Information Retrieval to Knowledge Management,” IIS-9817473, April 1999–March 2002;
- NIH/NLM Grant (PI: H. Chen), “UMLS Enhanced Dynamic Agents to Manage Medical Knowledge,” 1 R01 LM06919-1A1, February 2001–January 2004;
- NSF/CISE/CSS (PI: H. Chen), “An Intelligent CSCW Workbench: Analysis, Visualization, and Agents”, IIS-9800696, June 1998–June 2001.

We would also like to thank all current and previous members of the Artificial Intelligence Lab at the University of Arizona who have contributed to these projects.

## 10.5 Appendix A: URLs of Spiders and Search Engines

37.com – <http://www.37.com/>  
 AltaVista – <http://www.altavista.com/>  
 Autonomy – <http://www.autonomy.com/>  
 Blue Squirrel’s WebSeeker – <http://www.bluesquirrel.com/>  
 BuildingOnline – <http://www.buildingonline.com/>  
 CollegeBot – <http://www.collegebot.com/>  
 Copernic – <http://www.copernic.com/>  
 DejaNews – <http://www.dejanews.com/>  
 Dogpile – <http://www.dogpile.com/>  
 Excalibur – <http://www.excalib.com/>  
 Excite – <http://www.excite.com/>  
 Google – <http://www.google.com/>  
 HelpfulMed – <http://ai.bpa.arizona.edu/helpfulmed/>  
 Infoseek – <http://infoseek.go.com/>  
 JXTA Search – <http://search.jxta.org>



LawCrawler – <http://www.lawcrawler.com/>  
 Lycos – <http://www.lycos.com/>  
 MetaCrawler – <http://www.metacrawler.com/>  
 NorthernLight – <http://www.northernlight.com/>  
 SavvySearch – <http://www.savvysearch.com/>  
 Teoma – <http://www.teoma.com/>  
 Verity – <http://www.verity.com/>  
 WebSeek – <http://www.ctr.columbia.edu/webseek/>  
 Yahoo – <http://www.yahoo.com/>

## References

- 10.1 Amitay, E.: Using Common Hypertext Links to Identify the Best Phrasal Description of Target Web Documents. *Proc. the 21st ACM-SIGIR Post-Conference Workshop on Hypertext Information Retrieval for the Web* (Melbourne, Australia, 1998)
- 10.2 A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan: Searching the Web. *ACM Transactions on Internet Technology*, 1 (1), 2-43 (2001)
- 10.3 R. Armstrong, D. Freitag, T. Joachims, T. Mitchell: WebWatcher: A Learning Apprentice for the World-Wide Web. *Proc. the AAAI-95 Spring Symposium on Information Gathering from Heterogenous, Distributed Environments* (Stanford, California, USA, 1995)
- 10.4 M. Balabanovic, Y. Shoham: Learning Information Retrieval Agents: Experiment with Web Browsing. *Proc. the AAAI-95 Spring Symposium on Information Gathering from Heterogenous, Distributed Environments* (Stanford, California, USA, 1995)
- 10.5 R.K. Belew: Adaptive Information Retrieval: Using a Connectionist Representation to Retrieve and Learn about Documents. *Proc. the 12th ACM-SIGIR Conference on Research and Development in Information Retrieval* (Cambridge, Massachusetts, USA, 1989)
- 10.6 I. Ben-Shaul, M. Herscovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalhaim, V. Soroka, S. Ur: Adding Support for Dynamic and Focused Search with Fetuccino. *Proc. the 8th World-Wide Web Conference* (Toronto, May 1999)
- 10.7 T. Berners-Lee: Weaving the Web. Harper, San Francisco (1999)
- 10.8 K. Bharat, A. Broder: A Technique for Measuring the Relative Size and Overlap of Public Web Search Engines. *Proc. the 7th International World-Wide Web Conference* (Brisbane, Australia, 1998)
- 10.9 K. Bharat, M.R. Henzinger: Improved Algorithms for Topic Distillation in a Hyperlinked Environment. *Proc. the 21st ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne* (Australia, 1998)
- 10.10 C. Bowman, P. Danzig, U. Manber, M. Schwartz: Scalable Internet Resource Discovery: Research Problems and Approaches. *Communications of the ACM*, 37 (8) 98-107 (1994)
- 10.11 S. Brin, L. Page: The Anatomy of a Large-scale Hypertextual Web Search Engine. *Proc. the 7th International World-Wide Web Conference* (Brisbane, Australia, 1998)
- 10.12 A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. Wiener: Graph Structure in the Web. *Proc. the 9th International World-Wide Web Conference* (Amsterdam, Netherlands, May 2000)
- 10.13 M. Burner: Crawling Towards Eternity: Building an Archive of the World-Wide Web. *Web Techniques*, 2 (5) (1997)

- 10.14 S. Chakrabarti, B. Dom, S.R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, J. Kleinberg: Mining the Web's Link Structure. *IEEE Computer*, 32 (8), 60-67 (1999)
- 10.15 S. Chakrabarti, M. Joshi, V. Tawde: Enhanced Topic Distillation using Text, Markup Tags, and Hyperlinks. *Proc. the 24th ACM-SIGIR Conference on Research and Development in Information Retrieval* (New Orleans, Louisiana, USA, Sep. 2001)
- 10.16 S. Chakrabarti, M. van den Berg, B. Dom: Focused Crawling: A New Approach to Topic-specific Web Resource Discovery. *Proceedings of the 8th International World-Wide Web Conference* (Toronto, Canada, May 1999)
- 10.17 M. Chau, D. Zeng, H. Chen: Personalized Spiders for Web Search and Analysis. *Proc. the 1st ACM-IEEE Joint Conference on Digital Libraries* (Roanoke, Virginia, USA, Jun 2001) pp. 79-87.
- 10.18 M. Chau, D. Zeng, H. Chen, M. Huang, D. Hendriawan: Design and Evaluation of a Multi-agent Collaborative Web Mining System. *Decision Support Systems* (2002) in press.
- 10.19 H. Chen: Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms. *Journal of the American Society for Information Science*, 46 (3), 194-216 (1995)
- 10.20 H. Chen, M. Chau, D. Zeng: CI Spider: A Tool for Competitive Intelligence on the Web. *Decision Support Systems* (2002) in press.
- 10.21 H. Chen, Y. Chung, M. Ramsey, C.C. Yang: A Smart Itsy-Bitsy Spider for the Web. *Journal of the American Society for Information Science*, Special Issue on AI Techniques for Emerging Information Systems Applications, 49 (7), 604-618 (1998)
- 10.22 H. Chen, Y. Chung, M. Ramsey, C.C. Yang: An Intelligent Personal Spider (Agent) for Dynamic Internet/Intranet Searching. *Decision Support Systems*, 23, 41-58 (1998)
- 10.23 H. Chen, H. Fan, M. Chau, D. Zeng: MetaSpider: Meta-searching and Categorization on the Web. *Journal of the American Society of Information Science & Technology*, 52 (13), 1134-1147 (1998)
- 10.24 H. Chen, A. Houston, R.R. Sewell, B. Schatz: Internet Browsing and Searching: User Evaluations of Category Map and Concept Space Techniques. *Journal of the American Society for Information Science*, Special Issue on AI Techniques for Emerging Information Systems Applications, 49 (7) 582-603 (1998)
- 10.25 H. Chen, T. Ng: An Algorithmic Approach to Concept Exploration in a Large Knowledge Network (Automatic Thesaurus Consultation): Symbolic Branch and Bound Search vs. Connectionist Hopfield Net Activation. *Journal of the American Society for Information Science*, 46 (5) 348-369 (1995).
- 10.26 H. Chen, C. Schufels, R. Orwig: Internet Categorization and Search: A Self-organizing Approach, *Journal of Visual Communication and Image Representation*, 7 (1) 88-102 (1996)
- 10.27 Y.J. Chen, V.W. Soo: Ontology-based Information Gathering Agents. *Proc. the 1st Asia-Pacific Conference on Web Intelligence* (Maebashi City, Japan, Oct 2001) pp. 423-427.
- 10.28 F.C. Cheong: *Internet Agents: Spiders, Wanderers, Brokers, and Bots* (New Riders Publishing, Indianapolis, Indiana, USA, 1996)
- 10.29 J. Cho, H. Garcia-Molina: The Evolution of the Web and Implications for an Incremental Crawler. *Proc. the 26th International Conference on Very Large Databases (VLDB 2000)* (Cairo, Egypt, 2000)
- 10.30 J. Cho, H. Garcia-Molina, L. Page: Efficient Crawling through URL Ordering. *Proc. the 7th International World-Wide Web Conference* (Brisbane, Australia, Apr 1998)
- 10.31 F. Crimmins, A.F. Smeaton, T. Dkaki, J. Mothe: TetraFusion: Information Discovery on the Internet. *IEEE Intelligent System*, Jul-Aug, 55-62 (1999)

- 10.32 P. DeBra, R. Post: Information Retrieval in the World-Wide Web: Making Client-based Searching Feasible. *Proc. the First International World-Wide Web Conference* (Geneva, Switzerland, 1994)
- 10.33 M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, M. Gori: Focused Crawling using Context Graphs. *Proc. the 26th International Conference on Very Large Databases (VLDB 2000)* (Cairo, Egypt, 2000) pp. 527-534
- 10.34 R.B. Doorenbos, O. Etzioni, D.S. Weld: A Scalable Comparison-shopping Agent for the World-Wide Web. *Proc. the First International Conference on Autonomous Agents (Agents'97)* (Marina del Rey, California, USA, Feb 1997) pp. 39-48
- 10.35 Drott, M. C.: Indexing Aids at Corporate Websites: The Use of robots.txt and META Tags. *Information Processing and Management*, 38, 209-219 (2002)
- 10.36 C. Dwork, R. Kumar, M. Noar, D. Sivakumar: Rank Aggregation Methods for the Web. *Proc. the 10th International World-Wide Web Conference* (Hong Kong, May 2001)
- 10.37 D. Eichmann: The RBSE Spider Balancing Effective Search Against Web Load. *Proc. the 1st International World-Wide Web Conference* (Geneva, Switzerland, 1994)
- 10.38 D. Eichmann: Ethical Web Agents. *Proc. the 2nd International World-Wide Web Conference* (Chicago, Illinois, USA, 1994)
- 10.39 G.W. Flake, S. Lawrence, C.L. Giles, F. Coetzee: Self-organization of the Web and Identification of Communities. *IEEE Computer*, 35 (3), 66-71 (2002)
- 10.40 S. Gauch, G. Wang, M. Gomez: Profusion: Intelligent Fusion from Multiple Different Search Engines. *Journal of Universal Computer Science*, 2 (9) (1996)
- 10.41 M. Gordon: Probabilistic and Genetic Algorithms for Document Retrieval. *Communications of the ACM*, 31 (10) 1208-1218 (1988)
- 10.42 M. Gray: Internet Growth and Statistics: Credits and Background. [Online]. Available at <http://www.mit.edu/people/mkgray/net/background.html> (1993)
- 10.43 T.H. Haveliwala: Efficient Computation of PageRank. Stanford University Technical Report. Available at: <http://dbpubs.stanford.edu:8090/pub/1999-31> (1999)
- 10.44 M. R. Henzinger: Hyperlink Analysis for the Web. *IEEE Internet Computing*, 5 (1), 45-50 (2001).
- 10.45 M.R. Henzinger, A. Heydon, M. Mitzenmacher, M. Najork: On Near-uniform URL Sampling. *Proc. the 9th International World-Wide Web Conference* (Amsterdam, Netherlands, May 2000)
- 10.46 A. Heydon, M. Najork: Performance Limitations of the Java Core Libraries. *Proc. the 1999 ACM Java Grande Conference*, (Jun 1999) pp. 35-41.
- 10.47 A. Heydon, M. Najork: Mercator: A Scalable, Extensible Web Crawler. *World-Wide Web*, 219-229 (Dec 1999)
- 10.48 J.J. Hopfield: Neural Network and Physical Systems with Collective Computational Abilities. *Proc. the National Academy of Science, USA*, 79 (4), 2554-2558 (1982).
- 10.49 A.E. Howe, D. Dreilinger: SavvySearch: A Meta-search Engine that Learns which Search Engines to Query. *AI Magazine*, 18 (2) 19-25 (1997)
- 10.50 B. Kahle: Preserving the Internet. *Scientific America* (Mar 1997).
- 10.51 J. Kleinberg: Authoritative Sources in a Hyperlinked Environment. *Proc. the 9th ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, California, USA, Jan 1998) pp. 668-677.
- 10.52 T. Kohonen, T.: *Self-organizing Maps* (Springer, Berlin, 1995)
- 10.53 M. Koster: A Standard for Robot Exclusion. [Online]. Available at: <http://www.robotstxt.org/wc/norobots.html> (1994)
- 10.54 R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins: Trawling the Web for Emerging Cyber-communities. *Proc. the 8th World-Wide Web Conference* (Toronto, May 1999)
- 10.55 K.L. Kwok: A Neural Network for Probabilistic Information Retrieval. *Proc. the 12th ACM-SIGIR Conference on Research and Development in Information Retrieval* (Cambridge, Massachusetts, USA, Jun 1989) pp. 21-30

- 10.56 S. Lawrence, C.L. Giles: Inquirus, the NECI Meta Search Engine. *Proc. the 7th International World-Wide Web Conference* (Brisbane, Australia, Apr 1998)
- 10.57 S. Lawrence, C.L. Giles: Context and Page Analysis for Improved Web Search. *IEEE Internet Computing*, Jul-Aug, 38-46 (1998).
- 10.58 S. Lawrence, C.L. Giles: Accessibility of Information on the Web. *Nature*, 400, 107-109 (1999)
- 10.59 C. Lin, H. Chen, J. Nunamaker: Verifying the Proximity and Size Hypothesis for Self-organizing Maps. *Journal of Management Information Systems*, 16 (3) 61-73 (2000)
- 10.60 P. Lyman, H.R. Varian: How Much Information. [Online]. Available at <http://www.sims.berkeley.edu/how-much-info/> (2000)
- 10.61 U. Manber, M. Smith, B. Gopal: WebGlimpse: Combining Browsing and Searching. *Proc. the USENIX 1997 Annual Technical Conference* (Anaheim, California, Jan 1997)
- 10.62 M.L. Mauldin: Lycos: Design Choices in an Internet Search Service. *IEEE Expert*, 12 (1) 8-11 (1997)
- 10.63 M.L. Mauldin: Spidering BOF Report. *Report of the Distributed Indexing/Searching Workshop*, (Cambridge, Massachusetts, USA, May 1996)
- 10.64 O.A. McBryan: GENVL and WWW: Tools for Taming the Web. *Proc. the 1st International World Wide Web Conference* (Geneva, Switzerland, 1994)
- 10.65 A. McCallum, K. Nigam, J. Rennie, K. Seymore: A Machine Learning Approach to Building Domain-specific Search Engines. *Proc. the International Joint Conference on Artificial Intelligence (IJCAI-99)* (1999) pp. 662-667
- 10.66 Z. Michalewicz (1992): *Genetic Algorithms + Data Structures = Evolution Programs*. (Springer, Berlin, 1992)
- 10.67 R.C. Miller, K. Bharat: SPHINX: A Framework for Creating Personal, Site-specific Web Crawlers. *Proceedings of the 7th International World-Wide Web Conference* (Brisbane, Australia, Apr 1998)
- 10.68 M. Najork, J.L. Wiener: Breadth-first Search Crawling Yields High-quality Pages. *Proceedings of the 10th International World-Wide Web Conference* (Hong Kong, May 2001)
- 10.69 Netcraft: Web Server Survey. [Online]. Available at <http://www.netcraft.com/Survey/Reports/0202/> (2002)
- 10.70 Z.Z. Nick, P. Themis: Web Search Using a Genetic Algorithm. *IEEE Internet Computing*, 5 (2) 18-26 (2001)
- 10.71 J. Pearl: Heuristics: Intelligent Search Strategies for Computer Problem Solving. (Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1984)
- 10.72 B. Pinkerton: Finding What People Want: Experiences with the WebCrawler. *Proc. the 2nd International World-Wide Web Conference* (Chicago, Illinois, USA, 1994)
- 10.73 P. Pirolli, J. Pitkow, R. Rao: Silk from a Sow's Ear: Extracting Usable Structures from the Web. *Proc. the ACM Conference on Human Factors in Computing Systems* (Vancouver, Canada, Apr 1996)
- 10.74 J. Rennie, A.K. McCallum: Using Reinforcement Learning to Spider the Web Efficiently. *Proc. the 16th International Conference on Machine Learning (ICML-99)* (Bled, Slovenia, 1999) pp. 335-343
- 10.75 G. Salton: Another Look at Automatic Text-retrieval Systems. *Communications of the ACM*, 29 (7) 648-656 (1986)
- 10.76 E. Selberg, O. Etzioni: Multi-service Search and Comparison using the MetaCrawler. *Proc. the 4th World-Wide Web Conference* (Boston, MA USA, December 1995)
- 10.77 E. Selberg, O. Etzioni: The MetaCrawler Architecture for Resource Aggregation on the Web. *IEEE Expert*, Jan-Feb, 11-14 (1997)
- 10.78 J. Smith, S.F. Chang: Visually Searching the Web for Content. *IEEE Multimedia*, 4 (3), 12-20 (1997)
- 10.79 E. Spertus: ParaSite: Mining Structural Information on the Web. *Proc. the 6th International World-Wide Web Conference* (Santa Clara, California, USA, Apr 1997)

- 10.80 S. Spetka: The TkWWW Robot: Beyond Browsing. *Proc. the 2nd International World-Wide Web Conference* (Chicago, Illinois, USA, 1994)
- 10.81 R.G. Sumner, K. Yang, B.J. Dempsey: An Interactive WWW Search Engine for User-defined Collections. *Proc. the 3rd ACM Conference on Digital Libraries* (Pittsburgh, Pennsylvania, USA, Jun 1998) pp. 307-308
- 10.82 The ht://dig Group.: htdig Reference. [Online]. Available at <http://www.htdig.org/htdig.html>
- 10.83 K.M. Tolle, H. Chen: Comparing Noun Phrasing Techniques for Use with Medical Digital Library Tools. *Journal of the American Society for Information Science, Special Issue on Digital Libraries*, 51 (4) 352-370 (2000)
- 10.84 S. Vrettos, A. Stafylopatis: A Fuzzy Rule-based Agent for Web Retrieval-filtering. *Proc. the 1st Asia-Pacific Conference on Web Intelligence* (Maebashi City, Japan, Oct 2001) pp. 448-453
- 10.85 S. Waterhouse, D.M. Doolin, G. Kan, Y. Faybishenko: Distributed Search in P2P Networks. *IEEE Internet Computing*, 6 (1) 68-72 (2002)
- 10.86 R. Weiss, B. Velez, M.A. Sheldon: HyPursuit: A Hierarchical Network Search Engine that Exploits Content-link Hypertext Clustering. *Proceedings of the ACM Conference on Hypertext* (Washington, DC, USA, 1996)
- 10.87 J. Weizenbaum: Eliza – A Computer Program for the Study of Natural Language Communication between Man and Machine. *Communication of the ACM*, 9 (1), 36-45 (1966)
- 10.88 I.H. Witten, D. Bainbridge, S.J. Boddie: Greenstone: Open-source DL Software. *Communications of the ACM*, 44 (5), 47 (2001)
- 10.89 I.H. Witten, R.J. McNab, S.J. Boddie, D. Bainbridge: Greenstone: A Comprehensive Open-source Digital Library Software System. *Proc. the 5th ACM Conference on Digital Libraries* (San Antonio, Texas, USA, 2000) pp. 113-121
- 10.90 A.H. Winston: *Artificial Intelligence* (Addison-Wesley Publishing Company Inc., Reading, Massachusetts, Second Edition, 1984)
- 10.91 C.C. Yang, J. Yen, H. Chen: Intelligent Internet Searching Agent Based on Hybrid Simulated Annealing. *Decision Support Systems*, 28, 269-277 (2000)
- 10.92 C. Yu, W. Meng, K.L. Liu: Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents. *Proc. the 2001 ACM SIGMOD International Conference on Management of Data* (Dallas, Texas, May 2001) pp. 187-198
- 10.93 O. Zamir, O. Etzioni: Grouper: A Dynamic Clustering Interface to Web Search Results. *Proc. the 8th World-Wide Web Conference* (Toronto, May 1999)