

Comparison of Three Vertical Search Spiders

In domain-specific search experiments, a Web spider based on a neural network algorithm consistently outperformed spiders based on traditional graph search and PageRank algorithms.

*Michael Chau
Hsinchun
Chen*
University of
Arizona

The Web has plenty of useful resources, but its dynamic, unstructured nature makes them difficult to locate. Search engines help, but the number of Web pages now exceeds two billion, making it difficult for general-purpose engines to maintain comprehensive, up-to-date search indexes. Moreover, as the Web grows ever larger, so does information overload in query results. A general-purpose search engine, such as Google (www.google.com) or AltaVista (www.altavista.com), usually generates thousands of hits, many of them irrelevant to the user query.

Vertical search engines solve part of the problem by keeping indexes only in specific domains. Examples include LawCrawler (www.lawcrawler.com), BuildingOnline (www.buildingonline.com), and SciSeek (www.sciseek.com). However, these engines still face the challenge of collecting a set of relevant, high-quality pages from the two billion available on the Web. Further, the demand for high-quality pages is generally—and sometimes critically—more important in vertical search engines.

Spiders are the software agents that search engines use to collect content for their databases. We investigated algorithms to improve the performance of vertical search engine spiders. The investigation addressed three approaches: a breadth-first graph-traversal algorithm with no heuristics to refine the search process, a best-first traversal algorithm that used a hyperlink-analysis heuristic, and a spreading-activation algorithm based on modeling the Web as a neural network.

INTELLIGENT SPIDERING

Most spiders use simple graph search algorithms, such as breadth-first search, to collect Web pages. Without controls, the spiders will fetch pages for any topic. There are two popular ways to control the fetch relevance and quality:

- Restrict the spiders to particular Web domains. For example, most Web pages within the www.toyota.com domain would be relevant to automobiles.
- Filter the collected pages on the basis of content. For example, a program could remove pages that had fewer than a threshold number of relevant keywords.

Both approaches have some disadvantages. Restricting the domains misses potentially relevant Web sites that are outside the original list; further, it does not work for sites that have diverse content. On the other hand, filtering the collected pages for a full Web search is inefficient.

Good spidering algorithms can improve the precision of search results, however, by predicting whether a URL points to a relevant Web page before downloading it to the local search engine database. Such predictions depend on representing Web content and structure in ways that are meaningful to machines. Current research in this area falls into one of two categories: content-based or link-based.

Content-based Web analysis

Spiders can apply indexing techniques for text

Link-Based Web Analysis: PageRank and HITS Algorithms

The PageRank algorithm calculates the quality of a page p proportionally to the quality of the pages that contain in-links to it.¹ Since PageRank also considers the quality of the in-linking pages, the score of p is calculated recursively as follows:

$$\text{PageRank}(p) = (1 - d) + d \times \sum_{\substack{\text{all } q \text{ linking} \\ \text{to } p}} \left(\frac{\text{PageRank}(q)}{c(q)} \right)$$

where d is a damping factor between 0 and 1, n is the total number of pages in the collection, and $c(q)$ is the number of in-linking pages q .

As Figure A shows, a Web page p can have a high score if many pages q link to it. The scores will be even higher if the referring pages also have high PageRank scores.

PageRank has proved effective in ranking search results in commercial search engines, such as Google. Researchers have also applied it to guide search engine spiders, as we do in our experiments, sending the spiders first to URLs with higher PageRank scores. The algorithm is computationally expensive, however, because it calculates the score of each Web page iteratively.

The Hyperlink-Induced Topic Search algorithm² is similar to PageRank. HITS defines *authority pages* as high-quality pages related to a particular topic or search query and *hub pages* as those that are not necessarily authority pages themselves but provide links to authority pages. Figure A illustrates the basic idea: A page that many other pages point to should be a good authority, and a page pointing to many others should be a good hub. HITS calculates an authority score and a hub score for each page as follows:

$$\text{AuthorityScore}(p) = \sum_{\substack{\text{all } q \text{ linking} \\ \text{to } p}} (\text{HubScore}(q))$$

$$\text{HubScore}(p) = \sum_{\substack{\text{all } r \text{ linking} \\ \text{from } p}} (\text{AuthorityScore}(r))$$

In an application of the HITS algorithm, the Clever search engine³ achieved a higher user evaluation than the manually compiled Yahoo directory. Other research has extended the basic algorithm—for example, to factor in how much a node, based on its relevance, influences its neighbors.⁴

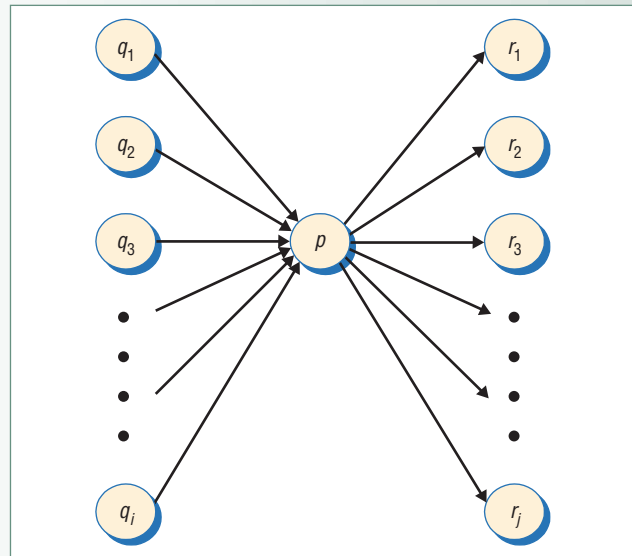


Figure A. Link-based algorithms. The PageRank score for a page p depends on the PageRank scores of referring pages, q_1 to q_i , pointing to p . In the HITS algorithm, the authority score of a page p depends on the hub scores of referring pages, q_1 to q_i , and the hub score depends on the authority scores of the pages to which p is pointing, r_1 to r_j .

Like PageRank, HITS calculates its scores iteratively and so has high computational costs.

References

1. S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proc. 7th WWW Conf.*, 1998; www7.scu.edu.au/programme/fullpapers/1921/com1921.htm.
2. J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Proc. 9th ACM-SIAM Symp. Discrete Algorithms*, ACM Press, 1998, pp. 668-677.
3. S. Chakrabarti et al., "Mining the Web's Link Structure," *Computer*, Aug. 1999, pp. 60-67.
4. K. Bharat and M.R. Henzinger, "Improved Algorithms for Topic Distillation in a Hyperlinked Environment," *Proc. 21st ACM-SIGIR Conf.*, ACM Press, 1998, pp. 104-111.

analysis and keyword extraction to help determine whether a page's content is relevant to a target domain. They can incorporate domain knowledge into their analysis to improve the results. For example, they can check the words on a Web page against a list of domain-specific terminology and assign a higher weight to pages that contain words from the list. Assigning a higher weight to words and phrases in the title or headings is also standard information-retrieval practice that spiders can apply based on appropriate HTML tags.

The URL address often contains useful information about a page. For example, <http://ourworld.com>

compuserve.com/homepages/LungCancer/ tells us that the page comes from the compuserve.com domain and that it likely has information relating to lung cancer.

An intelligent spider might consider pages from a .gov site to be more authoritative than pages from a .com site. Some metrics also consider URLs with fewer slashes more useful than those with more slashes.¹

Link-based Web analysis

Recent research has used Web link structure to infer important information about pages. Intuitively,

Vertical search engines offer more opportunity to apply domain knowledge in spider applications.

the author of a Web page A, who places a link to Web page B, believes that B is relevant to A. The term *in-links* refers to the hyperlinks pointing to a page. Usually, the larger the number of in-links, the higher a spider will rate a page. The rationale is similar to citation analysis, in which an often-cited article is considered better than one never cited.

Anchor text is the word or phrase that hyperlinks to a target page. Anchor text can provide a good source of information about a target page because it represents how people linking to the page actually describe it.

Several studies have tried to use either the anchor text or the text near it to predict a target page's content.²

It is also reasonable to give a link from an authoritative source, such as Yahoo (www.yahoo.com), a higher weight than a link from a personal homepage.

Researchers have developed several link-analysis algorithms over the past few years. The "Link-Based Web Analysis: PageRank and HITS Algorithms" sidebar describes the two most widely used algorithms.

THREE APPROACHES

Simple breadth-first and best-first algorithms exemplify the graph search approaches of most existing search engine spiders. There is little research on using more powerful algorithms. Furthermore, while Web content- and link-analysis techniques provide good heuristics in the spidering process, most applications of them, such as PageRank and HITS, are computationally expensive. Moreover, the techniques are seldom combined effectively.

Vertical search engines offer more opportunity to apply domain knowledge in spider applications. The three spiders we developed address different ways of combining content- and link-based Web analyses and integrating them with graph search algorithms.

BFS spider

Our breadth-first search spider follows a basic graph-traversal algorithm. The BFS approach assumes that a URL relevant to a target domain is likely to have other relevant Web pages in its neighborhood, and many commercial search engines have used this spidering technique.

Because the most important pages on a topic are likely to include in-links from many hosts, BFS has proved effective in discovering high-quality pages early in a spidering process.³

PageRank spider

We combine link-based analysis with a heuristics-based traversal algorithm in our PageRank spider. Adapting the algorithm reported by Junhoo Cho and colleagues,⁴ our spider performs a best-first graph search using PageRank as the heuristic. In each step, the spider gets the URL with the highest PageRank score, fetches the content, and extracts and enqueues all the page's outgoing links. It runs until it has collected a predetermined number of pages. The PageRank spider calculates scores iteratively, as described in the "Link-Based Web Analysis" sidebar. Our implementation sets the damping factor d to 0.90.

We also adapted this queuing approach for anchor text analysis.⁴ As in the earlier study, we established two priority queues for the PageRank spider, `hot_queue` and `normal_queue`, and ordered the URLs within each queue in descending order by PageRank score. The spider first dequeues from the `hot_queue`. If the `hot_queue` is empty, the spider dequeues from the `normal_queue`.

In our design, the spider places a URL in the `hot_queue` if the anchor text pointing to this URL contains a relevant term from a predefined list of domain terminology.

Hopfield Net spider

A neural network is a graph of many active nodes (neurons) that are connected with each other by weighted links (synapses). The network uses activation algorithms over the nodes to represent and retrieve knowledge.^{5,6} We can apply these algorithms to Web applications by modeling the Web as a neural network in which the nodes are Web pages and the links are simply hypertext links.

Accordingly, we modeled the Web as a weighted, single-layer neural network called a Hopfield Net.⁶ A Hopfield Net activates its nodes in parallel and combines activation values from different sources for each individual node until the node activation scores across the network converge to a stable state. This model combines a parallel search algorithm with content-based and link-based analysis.

Our Hopfield Net spider incorporates a spreading-activation algorithm for knowledge discovery and retrieval, though we modified it significantly to account for the Web's unique characteristics. The implementation occurs in three steps.

Initialization. Starting with a set of seed URLs, each represented as a node with a weight of 1, the spider fetches and analyzes the seed Web pages in iteration 0. The weight of node i at iteration t is expressed as $\mu_i(t)$. Thus, for all seed URLs, $\mu_i(0) = 1$.

The spider adds the new URLs found in the seed pages to the network.

Activation, weight computation, and iteration. Proceeding to the next iteration, the spider calculates each node's weight as follows:

$$\mu_i(t+1) = f_s \left(\sum_{\substack{\text{every known} \\ \text{parent } b \text{ of } i}} w_{b,i} \mu_b(t) \right)$$

where $w_{b,i}$ is the weight of the link between two nodes and f_s is the sigmoidal transformation function that normalizes the weight to a value between 0 and 1.

The link weight $w_{b,i}$ estimates whether a URL i pointed to from a Web page b is relevant to the target domain, based on a measure of the anchor text. For example, we can calculate the weight $w_{b,i}$ as a function of the number of words relevant to the target domain used in page b 's anchor text linking to page i .

We adopted a slightly modified sigmoidal function as follows:

$$f_s(x) = \left(\frac{1}{1 + e^{-x}} - 0.5 \right) \times 2$$

After the spider calculates the weights of all nodes in the current iteration, it activates (visits) that set of nodes (URLs) and fetches them from the Web in descending order of weight. To filter out low-quality URLs, the spider bypasses nodes with a weight below a threshold θ . Figure 1 illustrates the activation process.

After the spider has visited and downloaded all the pages with a weight greater than the threshold θ , it updates the weight of each node in the new iteration to reflect the quality and relevance of the downloaded page content as follows:

$$\mu_i(t+1) = f_s[\mu_i(t) \times p_i]$$

where p_i is a weight that represents the relevance of the textual content of a page i . This score is a function of the number of phrases in a page's content that are relevant to the target domain. A page with more relevant phrases will receive a higher score.

Stopping condition. The process iterates until the spider has collected a predetermined number of Web pages or until the average weight of all nodes in an iteration is smaller than a maximum allowable error (a small number).

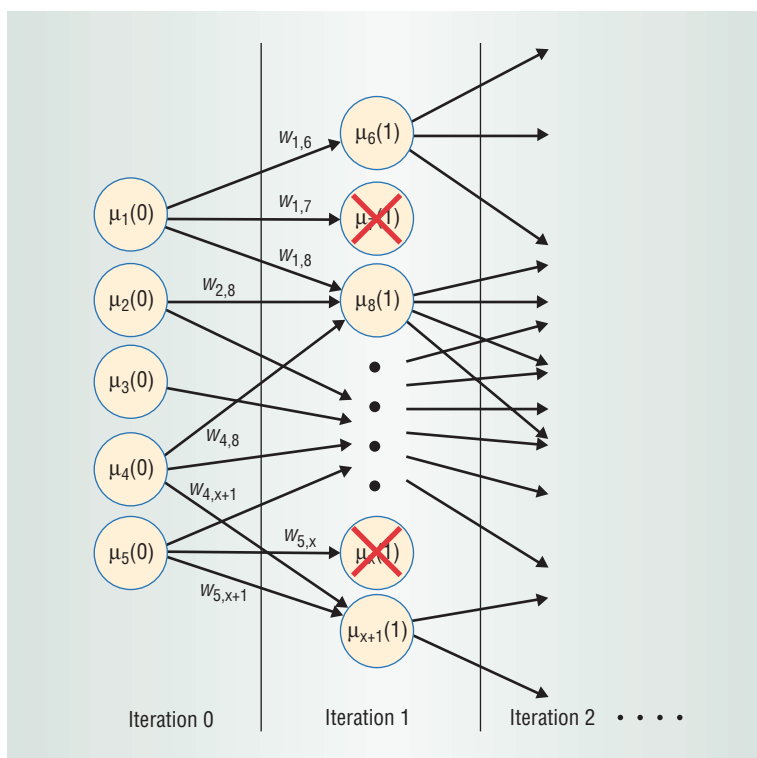


Figure 1. Spreading activation. Starting with a set of seed URLs, the Hopfield Net spider activates neighbor URLs, combines weighted links, and determines the weights of newly discovered nodes. Nodes with a low weight (in this case, nodes 7 and x) are discarded.

MEDICAL DOMAIN TEST BED

To evaluate the three approaches, we implemented them as backend spiders for a medical search engine called HelpfulMed (ai.bpa.arizona.edu/helpfulmed/).⁸ Medical Web pages are highly distributed, highly variable in their quality, and difficult to locate. HelpfulMed is designed to collect high-quality information that can support professional users in potentially significant decisions. It provided an ideal test bed for comparing spider performance in a vertical domain.

Establishing domain knowledge

To facilitate content-based analysis, we used the Unified Medical Language System to develop a medical lexicon, called the Good Phrase List. UMLS is a long-term research and development project of the US National Library of Medicine that provides knowledge sources for medical professionals and researchers. A medical librarian reviewed the semantic types in the UMLS meta-thesaurus and extracted about 300,000 medical phrases from the entire collection for the Good Phrase List.

In addition, the librarian manually compiled a Bad Phrase List that contains 118 unwanted, non-medical words that frequently appear in medical Web pages—for example, “Job posting” and “Contact us.”

Finally, the librarian also identified a set of 354 medical domain Web sites that were either good hubs or good authorities, such as the US National Library of Medicine Web site (www.nlm.nih.gov). From this list, the librarian further identified five

The Hopfield Net spider incorporates domain-specific content analysis into the scoring function.

high-quality hub pages as seed URLs, which served as the starting points in our experiment.

Creating the test bed

To prevent variations in network load and traffic from affecting the performance results, we set up a controlled environment for our experiments by creating a local repository of a portion of the Web relevant to our study.

The repository supported virtual spidering, meaning that a spider fetched a page's content from the local repository rather than the Web.

We created the repository by running a random-first search, using the five seed URLs identified by our medical expert as starting points and then fetching all new links in a random order. The resulting test bed consisted of 1,040,388 valid, unique Web pages and 6,904,026 links. The repository contained pages both inside and outside the starting Web sites as well as both medical and non-medical Web pages.

We also ran the Arizona Noun Phraser⁹ to extract noun phrases from each test bed page. We then calculated the total number of phrases and the number of phrases that appeared in the Good Phrase List for each page.

EXPERIMENTS

We designed and conducted two experiments to compare the three spiders.

Simulation

In the first experiment, we simulated the spidering processes to analyze their speed and quality.

Experiment design. First, we executed each spider on the test bed. Although the local repository contained information for all test bed pages, each spider could access only information based on pages it already had visited. For each spider, we used the same seed URL set that we used to create the test bed, and we ran the spider until it had visited 100,000 pages. To ensure that each spider collected the same number of pages, we did not use the Hopfield Net spider's convergence property in the experiment.

To compare performance in terms of the quality of each Web page visited, we introduced the notion of Good Page, which estimated a Web page's relevance to the medical domain automatically. Based on a previous experiment on a similar but smaller collection (about 100,000 Web pages), we considered a Web page to be a Good Page if the number of medical phrases divided by the total number of

phrases found in the page was greater than a certain threshold. In our previous experiment, we used this method to classify a set of randomly sampled Web pages and found the error rate to be 5.0 percent for the medical domain. Using this classification, the test bed in the current experiment contained 171,405 Good Pages.

Using the notion of Good Page, we defined the precision and recall rates of the spiders as follows:

$$\text{precision rate} = \frac{\text{number of Good Pages visited by the spider}}{\text{number of all pages visited by the spider}}$$

$$\text{recall rate} = \frac{\text{number of Good Pages visited by the spider}}{\text{number of Good Pages in the test bed}}$$

Because the total number of Good Pages in the test bed was 171,405 and the total number of all pages visited by a spider was fixed at 100,000, the precision and recall rates were directly proportional to each other for each spider. We focus our discussion here on the precision rate. We also compared efficiency by measuring the time each spider used to visit 100,000 pages.

Results. Figure 2 summarizes the simulation results. Figure 2a shows that the Hopfield Net spider retrieved 40,014 Web pages (40.0 percent of all pages visited) compared with 36,307 (36.3 percent) by the BFS spider and 19,630 (19.6 percent) by the PageRank spider. The Hopfield Net spider took 12.6 minutes to retrieve 100,000 pages, the BFS spider took 12.7 minutes, and the PageRank spider took significantly longer at 1,183.6 minutes.

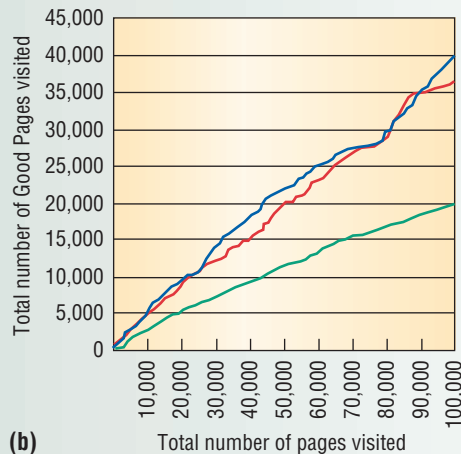
In addition to the final collection, we studied spider performance during different stages of the process. Figure 2b shows the total number of Good Pages found during the spidering process for each system. The Hopfield Net spider consistently achieved the best performance. The BFS spider was slightly less effective, and the PageRank spider performed at a considerably lower level.

To analyze the data further, we divided the 100,000 pages that each spider visited into 1,000 equal portions, each containing 100 consecutive pages according to the original visiting order of each spider. Within each portion, we calculated the percentage of Good Pages in the search results. As there were 100,000 pages, we obtained 1,000 data points for each spider.

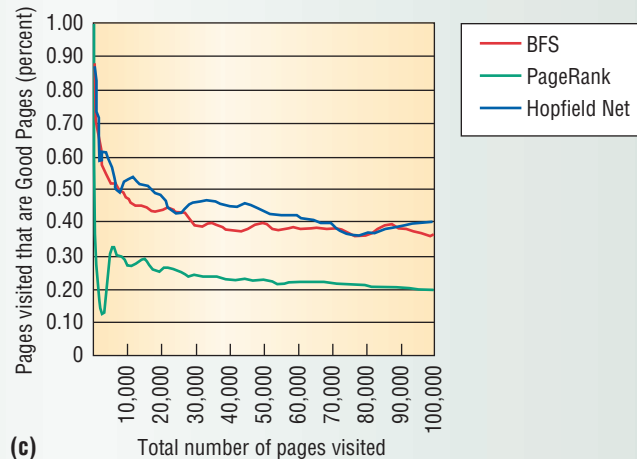
Figure 2c shows the results of paired t-tests conducted on these data. At the 1 percent level, the Hopfield Net spider had significantly better preci-

	Number of pages visited	Number of Good Pages visited	Precision (percent)	Time (minutes)
BFS spider	100,000	36,307	36.3	12.7
PageRank spider	100,000	19,630	19.6	1,183.6
Hopfield Net spider	100,000	40,014	40.0	12.6

(a)



(b)



(c)

sion than the BFS and PageRank spiders. The PageRank spider performed significantly below either of the other two throughout the process.

Discussion. The BFS spider obtained high-quality pages simply by visiting the URLs close to the starting URLs, which tended to point to relevant pages. We had expected the PageRank spider to perform at least as well as the BFS spider, which did not use any heuristics. However, detailed data analysis showed that the PageRank spider visited more irrelevant URLs than the other two spiders early in the search (the first 3,000 pages in Figure 2c).

For example, many of the first few hundred pages contained a link to the Adobe Web site (www.adobe.com), which provided information on how to open and read PDF files. Because of the large number of referring pages, the Adobe site URL had a high PageRank score. The spider therefore visited it early in the spidering process and, because the PageRank algorithm is recursive, propagated the high score to other URLs contained in the page. Although the PageRank algorithm is robust for large collections of pages, we believe its scores can be misleading for small collections, especially early in the spidering process.

The Hopfield Net spider did not suffer from this problem because it incorporates domain-specific content analysis into the scoring function, thus effectively combining content- and link-based analysis. Filtering the URLs also increased the precision rate.

The PageRank spider's heavy computational requirements also made it take much longer to execute than the other two spiders. Research has shown that it takes several hours for the PageRank algo-

rithm to calculate the scores of a set of 19 million pages.¹⁰ The PageRank spider made the recursion problem worse by calculating the PageRank scores not just once but each time the spidering process found any new URLs. Consequently, the spider became exponentially slower as the number of visited URLs increased. While PageRank is a good measure for ranking Web search results, it is impractical for the spidering process in large collections.

User study

The second experiment was a user study to determine how domain experts rated the Web pages by each system collected.

Experiment design. We recruited two senior graduate students with medical training to judge the quality of the pages collected by each spider. Each expert was assigned 100 Web pages randomly drawn from the collection of pages fetched by each spider during the simulation. To eliminate possible bias, we did not disclose the source for a page. The experts independently judged each page's quality and relevance to the medical domain on a score ranging from one to four.

Results. In general, the user study results agreed with those of the simulation experiment. The Hopfield Net spider score was 2.30, the highest relevance score among the three. The BFS spider scored 2.13, and the PageRank spider scored 1.78. The Hopfield Net and BFS spiders both scored significantly better than the PageRank spider at the 5 percent level. The Hopfield Net spider had a higher relevance score than the BFS spider, but our experiment did not statistically confirm the difference.

Figure 2. Simulation results: (a) tabular summary; (b) total number of Good Pages visited; (c) percentage of Good Pages among the pages visited.

Discussion. Our judges found that the PageRank spider collected many nonmedical pages from large sites such as Yahoo and eBay (www.ebay.com), where the pages tended to have higher PageRank scores and were visited by the PageRank spider during an early stage of the spidering process. As a result, the average relevance of the collection the PageRank spider built was not as high as the other two collections.

We have combined content and link structure analysis to improve the performance of vertical search spiders, but the combination holds promise for other Web applications as well. In addition, modeling the Web as a neural network opens a large body of research to Web applications. Our work continues to address the intersections of these fields. For instance, we are currently studying how the Web's link structure can enhance the performance of Web page classification and clustering applications. ■

Acknowledgments

This research is partly supported by the National Science Foundation under grant #9817473 and the National Institutes of Health under grant #1-R01-LM06919-1A1. We thank the National Library of Medicine for making UMLS freely available to researchers. We also thank the medical experts who participated in the user studies and the members from the University of Arizona Artificial Intelligence Lab, in particular Wojciech Wyzga, Haiyan Fan, Ming Yin, and Yohanes Santoso, who contributed to this project.

References

1. A. Arasu et al., "Searching the Web," *ACM Trans. Internet Technology*, vol. 1, no. 1, 2001, pp. 2-43.
2. E. Amitay, "Using Common Hypertext Links to Identify the Best Phrasal Description of Target Web Documents," *Proc. ACM Special Interest Group on Information Retrieval (SIGIR 98)*, Post-Conf. Workshop on Hypertext Information Retrieval for the Web, ACM Press, 1998; www.mri.mq.edu.au/~einat/sigir/.
3. M. Najork and J.L. Wiener, "Breadth-First Search Crawling Yields High-Quality Pages," *Proc. 10th WWW Conf.*, 2001; www10.org/cdrom/papers/208/.
4. J. Cho, H. Garcia-Molina, and L. Page, "Efficient Crawling through URL Ordering," *Proc. 7th WWW*

Conf., 1998; www7.scu.edu.au/programme/fullpapers/1919/com1919.htm.

5. H. Chen and T. Ng, "An Algorithmic Approach to Concept Exploration in a Large Knowledge Network (Automatic Thesaurus Consultation): Symbolic Brand-and-Bound Search vs. Connectionist Hopfield Net Activation," *J. Am. Soc. Information Science*, vol. 46, no. 5, 1995, pp. 348-369.
6. K.L. Kwok, "A Neural Network for Probabilistic Information Retrieval," *Proc. 12th ACM-SIGIR Conf.*, ACM Press, 1989, pp. 21-30.
7. J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Nat'l Academy of Science*, vol. 79, no. 4, 1982, pp. 2554-2558.
8. H. Chen et al., "HelpfulMed: Intelligent Searching for Medical Information over the Internet," accepted for publication in *J. Am. Soc. for Information Science and Technology*.
9. K.M. Tolle and H. Chen, "Comparing Noun Phrasing Techniques for Use with Medical Digital Library Tools," *J. Am. Soc. Information Science*, vol. 51, no. 4, 2000, pp. 352-370.
10. T.H. Haveliwala, *Efficient Computation of PageRank*, tech. report, Stanford Univ., Stanford, Calif., 1999; http://dbpubs.stanford.edu:8090/pub/1999-31.

Michael Chau is a doctoral candidate and a research associate of the Artificial Intelligence Lab in the Department of Management Information Systems at the University of Arizona. His research interests include information retrieval, Web mining, knowledge management, machine learning, and intelligent agents. Chau received a BS in computer science and information systems from the University of Hong Kong. Contact him at mchau@bpa.arizona.edu.

Hsinchun Chen is McClelland Professor of Management Information Systems and director of the Artificial Intelligence Lab and Hoffman E-Commerce Lab in the Department of Management Information Systems at the University of Arizona. His research interests include digital libraries, semantic retrieval, knowledge discovery, e-government, and medical informatics. Chen received a PhD in information systems from New York University. Contact him at hchen@bpa.arizona.edu.