Evaluating the Use of Search Engine Development Tools in IT Education

Michael Chau and Cho Hung Wong
School of Business, The University of Hong Kong, Pokfulam, Hong Kong.
E-mail: mchau@business.hku.hk; joewch@graduate.hku.hk

Yilu Zhou
Information Systems and Technology Management, George Washington University, Washington, DC. E-mail: yzhou@gwu.edu

Jialun Qin
Department of Management, University of Massachusetts Lowell, Lowell, MA.
E-mail: jialun_qin@uml.edu

Hsinchun Chen
Department of Management Information Systems, The University of Arizona, Tucson, AZ.
E-mail: hchen@eller.arizona.edu

It is important for education in computer science and information systems to keep up to date with the latest development in technology. With the rapid development of the Internet and the Web, many schools have included Internet-related technologies, such as Web search engines and e-commerce, as part of their curricula. Previous research has shown that it is effective to use search engine development tools to facilitate students' learning. However, the effectiveness of these tools in the classroom has not been evaluated. In this article, we review the design of three search engine development tools, SpidersRUs, Greenstone, and Alkaline, followed by an evaluation study that compared the three tools in the classroom. In the study, 33 students were divided into 13 groups and each group used the three tools to develop three independent search engines in a class project. Our evaluation results showed that SpidersRUs performed better than the two other tools in overall satisfaction and the level of knowledge gained in their learning experience when using the tools for a class project on Internet applications development.

Introduction

To develop an information system (IS), information technology (IT) professionals need all-rounded skills. First, they need to be proficient in various technical areas including databases, networking, and mathematics. On top of that, as real-world systems are having more components and, thus, become more complex, IT professionals should possess skills that let them choose the best components available and integrate them into system development. To do this, knowledge and experience in system design, project management, and personal communication are desired. In particular, as a result of the rapid growth of e-business and e-commerce, proficiency and knowledge related to the Internet and World Wide Web are becoming essential; these include more advanced technologies such as Web interface, script languages, and Internet protocols. Moreover, IT professionals may find it challenging to keep pace with the rapid evolving rate of these technologies.

To prepare students for the above-mentioned challenge, many educational institutions have designed projects and assignments that require students to develop systems by integrating various software or hardware components. A successful example has been demonstrated in a project called "Build Your Search Engine in 90 Days" (Chau, Huang, & Chen, 2003), which was started in Fall 1999 in a data structures and algorithms course taught in the Department of Management Information Systems at the University of Arizona. In the project, students were asked to form groups of four to six students and each group had to develop a topic-specific Web search engine within 3 months. The search engine should focus on a topic of interest (e.g., soccer, medicine, painting) chosen by the students rather than the

Received July 30, 2008; revised July 23, 2009; accepted August 6, 2009

© 2009 ASIS&T • Published online 5 November 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/asi.21223

entire Web like what Google does. Students were offered hands-on experience and developed better project management and teamwork skills by working on the "small-scale but complete version of real-life projects."

Based on the success of the above-mentioned project, we have designed a similar project for a course called Internet Applications Development at the University of Hong Kong (HKU). Students in the course were asked to form groups of at most three students. Each group was responsible for the design, development, and maintenance of a search engine in a specific topic at students' own choice.

Although the above-mentioned two projects share a number of similarities, they have different focuses. The previous project at the University of Arizona was offered as a graduate level class with prerequisites of Java programming. To build a search engine, students were offered two basic components of a Web search engine: a Web spider and an indexer, both of which can be used directly to collect and process Web pages or can be modified according to project needs. They put major efforts in creating a database, developing search algorithm, and integrating Web applications. Emphasizing heavily on system development, database management, user interface design, and system integration, the Arizona project was a great experience for students with some programming experience.

At HKU, students attending this class were undergraduate students with IS majors. Besides the technical development side of Web search engines, we decided to also focus on the ability to analyze user requirements, develop evaluation criteria, and conduct experiments to choose the best tool. There were a number of additional requirements that made this project more challenging. First, in the HKU project, the students were offered three packaged versions of their topic-specific search engine tools (Alkaline, Greenstone, and SpidersRUs) that were freely available for creating search engines. These tools provide integrated functions including spider, index, search, and a basic interface. Each group of students had to build three versions of their search engine in the same topic by using these three tools. In addition, students in the HKU project were encouraged to develop a search engine that supports multiple languages.

After the search engines have been completed, each group had to submit a written report, which was a part of the assessment in the project. In the report, students were asked to describe their search engines, including information such as reasons for choosing the specific topic and a detailed comparison between the three versions of search engines. However, it should be noted that technical details were not the major consideration in the HKU project. Students were instead assessed on correctness and quality (e.g., search results, interface) of their work.

The objective of this study is to evaluate the value of the three search engine building tools chosen, i.e., SpidersRUs, Alkaline, and Greenstone, in the classroom. The rest of the article is structured as follows. The Background section reviews the components of a Web search engine and the corresponding resources for each component. The section that follows introduces three search engine development tools for the project and explains how they can be used to develop Web search engines. The Evaluation section gives some examples of Web search engines developed by our students. The Evaluations section discusses an evaluation study on the effectiveness of the tool in the classroom, while the last section suggests some potential adaptation of the project for future use.

Background

Teaching Software Development and Programming

In the past, courses in most computer science and information systems curricula, such as software development and programming, were assessed using individual programming and written assignments. Recently, educational institutions have stressed the importance of teamwork, as we see that group projects or group programming exercises are becoming more and more popular.

There are several reasons for such a shift. First, when students work in a group, they can learn how to cooperate with their team members, thus forming a positive learning environment among them (Dutt, 1994; McConnell, 1996). Another reason is that group projects let students improve their written and oral skills (Harris, 1995). Moreover, group exercises help students develop soft skills. In particular, it has been shown that group programming exercise improved students' problem solving abilities, knowledge gain, and interpersonal skills (Granger & Lippert, 1999; Poindexter, 2003).

Covering a broad range of technical topics, from Web page design to spidering and indexing algorithms, search engine development not only provides students with the opportunity to work in a group but also allows them to apply their knowledge gained from fundamental disciplines such as data structures and algorithms. On top of that, working on a search engine exposes students to a number of Web technologies, which is often stressed in computer science and information systems education (Hickey et al., 2002), and the exposure further motivates students to go on to take more advanced courses such as algorithmic complexity and pattern matching (Bird & Curran, 2006).

In our study, students were asked to work in groups to develop their own Web search engines. However, unlike courses taught at other universities that focused more on programming and implementation, our course focused on system integration, enhancing students' creativity and arousing their interest in later courses. As a result, students were not asked to implement the core engine of the system, which would be demanding. Instead, we encouraged them to integrate existing tools to build their systems.

Components of a Web Search Engine

A typical search engine is made up of a set of spiders, a repository of Web page, an indexer, search indexes, a query engine, and a user interface. Each of these components is described in the following. For details, please refer to the work of Brin and Page (1998) and Arasu, Cho, Garcia-Molina, Paepcke, and Raghavan (2001).

- *Spiders* (also known as Web crawlers or Web bots) are programs that retrieve Web pages for search engines using HyperText Transfer Protocols (HTTP) by recursively following URL (Uniform Resource Locator) links in pages (Cheong, 1996; Chau & Chen, 2003). First, the spiders download some Web documents from a list of starting seed URLs. The URLs contained within these downloaded documents are extracted and added to the queue. The spiders then select the next URL from the queue and retrieve the corresponding documents.
- The downloaded documents are stored in a repository of *Web pages, which may be* compressed in advance to save storage space. The repository can be in the form of a database (which is common used in large-scale search engines) or file system.
- An *indexer* builds a search index for the repository of Web pages. The occurrence of each word in the pages is recorded. The data can then be used to calculate scores, such as the term frequency and document frequency of each word. A list of indexing results is then generated.
- *Search indexes* (which are stored in databases) are actually an "inverted version" of the indexing results generated by the indexer mentioned above. Although the original indexing results map a document to a list of words in the document, search indexes map a word to a list of documents containing the word.
- A *query engine* is an intermediary between the search indexes and the user interface. It performs several important tasks: accepting search queries from users, performing searches on the search indexes, ranking the search results, generating search summaries, and storing search logs (Chau, Fang, & Yang, 2007). In some search engines, the query engine is also responsible for caching the results of popular search queries.
- A Web *user interface* is a front-end component that allows users to submit their search queries and view the search results.

Depending on the scope of search results, there are two types of search engines: general-purpose and topic-specific ones. General-purpose search engines such as Google (www.google.com) allow users to search for any pages on the Web. However, these search engines may not be able to satisfy the needs of users looking for specific information. As a result, many topic-specific search engines were developed and made available online. For example, *LawCrawler* (www.lawcrawler.com) allows users to search for legal information. *BuildingOnline* (www.buildingonline.com), *SciSeek* (www.sciseek.com), and *BioView* (www.bioview.com) are a few other examples.

Review of Three Tools for Creating Topic-Specific Search Engines

There are several existing digital library tools that provide all the necessary components (i.e., spider, indexer, query engine, Web user interface) for creating small to medium-scale search engines. Most of them provide a simple way for creating topic-specific search engines. For example, the Alkaline Search Engine is a commercial tool specially designed for businesses to build their search engines. The Greenstone Digital Library Software is an open-source software package developed by the New Zealand Digital Library Project at the University of Waikato (Witten, Bainbridge, & Boddie, 2000; Witten, McNab, Boddie, & Bainbridge, 2001). In addition, we have also developed a tool called the SpidersRUs Digital Library Toolkit with an aim to address the problems in existing tools (Chau, Qin, Zhou, Tseng, & Chen, 2008).

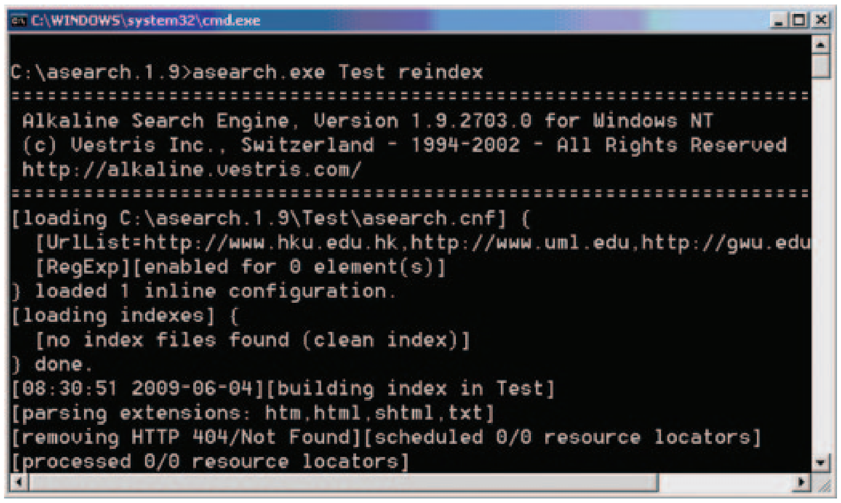
In our class project, we asked students to create a search engine using the three tools mentioned, namely, Alkaline, Greenstone, and SpidersRUs. Our intention here was to let students try different types of tools to understand the pros and cons of each tool. It also made it possible for us to study and compare the educational value of the three tools. In the following, we will give an overview of the three tools with sample user sessions.

Alkaline

Overview of Alkaline. Alkaline is a commercial search engine building toolkit and was developed using the C++ language by the Vestris Inc. (http://alkaline.vestris.com/docs/pdf/alkaline.pdf) based in Switzerland. It was designed to build a middle-scale search engine that covers 50,000 to 500,000 documents and is compatible to the Windows 98/NT/2000, Unix, MacOS, and all variations of Linux operating systems. Alkaline comprises two major components: a spidering/indexing component and a searching component (Vestris, 2004). The spidering/indexing component comprises a stand-alone Web spider that collects data from local or remote Web sites based on a given set of starting URLs. It allows the users to set up a series of constrains on the spidering process, such as the maximum downloading levels or specific URLs to exclude, etc. It also has limited ability to download and index CGI-powered dynamic Web pages and password-protected documents. Downloaded documents are automatically parsed and a word index is built for search purposes. Alkaline can index multiple types of documents such as PDF, MS Word, and LaTex files. It can also index multimedia files, such as Flash and MP3 files, if proper metadata is available. The searching component incorporates several searching and ranking algorithms. It generates a script-based search interface that can be inserted into existing Web pages. The user can also customize the format to present search results. These features were designed to meet the needs of those users who want to incorporate search engine functionalities into their own Web sites.

Sample user session of Alkaline. Alkaline does not provide a graphics user interface (GUI) for the development process. The collection building operations must be performed by entering DOS commands into a command console. This makes it much harder for users without strong technical backgrounds to use Alkaline.

To create a new collection, the user must first create a folder in his or her file system to store the data that are going to be collected. In this folder, the user first creates a file called



```
C:\asearch.1.9>asearch.exe Test reindex
================================================================
 Alkaline Search Engine, Version 1.9.2703.0 for Windows NT
 (c) Vestris Inc., Switzerland - 1994-2002 - All Rights Reserved
 http://alkaline.vestris.com/
================================================================
[loading C:\asearch.1.9\Test\asearch.cnf] {
  [UrlList=http://www.hku.edu.hk,http://www.uml.edu,http://gwu.edu
  [RegExp][enabled for 0 element(s)]
} loaded 1 inline configuration.
[loading indexes] {
  [no index files found (clean index)]
} done.
[08:30:51 2009-06-04][building index in Test]
[parsing extensions: htm,html,shtml,txt]
[removing HTTP 404/Not Found][scheduled 0/0 resource locators]
[processed 0/0 resource locators]
```

FIG. 1. Result of issuing the "asearch" command in Alkaline.

"*asearch.cnf*" in which the user specifies the configurations of the collection, such as the starting URL list. Then the user must issue the command "*asearch*" with the path of the aforementioned folder and keyword "reindex" as parameters to start the spidering and indexing process. For example, the user wants to create a demo collection called "*Test*" that includes Web pages from the following three Web sites: *www.hku.hk*, *www.gwu.edu*, and *www.uml.edu*. The user should first create a folder called "*Test*" and create a file named "*asearch.cnf*" containing the three starting URLs.

After the asearch.cnf file is created, the user should issue the following command in the command console to initiate the spidering and indexing process: "*asearch.exe Test reindex*". Figure 1 shows the results of successfully issuing the "asearch" command.

Alkaline automatically downloads the documents in the starting URLs and creates the word index. The index data will be stored in the same collection folder. After the spidering and indexing process has finished, the user can issue the "asearch" command again to enable the search service on the collection. Two parameters need to be provided: the port number on which the search service will be running on and the path of the folder where the index data are stored.

Once the search service is enabled, the user can navigate to the default search page in a regular Web browser such as the Microsoft Internet Explorer and start searching in the collection. In our example, the URL to the default search page is "*http://localhost:9999*." Figure 2 shows a screenshot of a search in the Test collection.

Greenstone

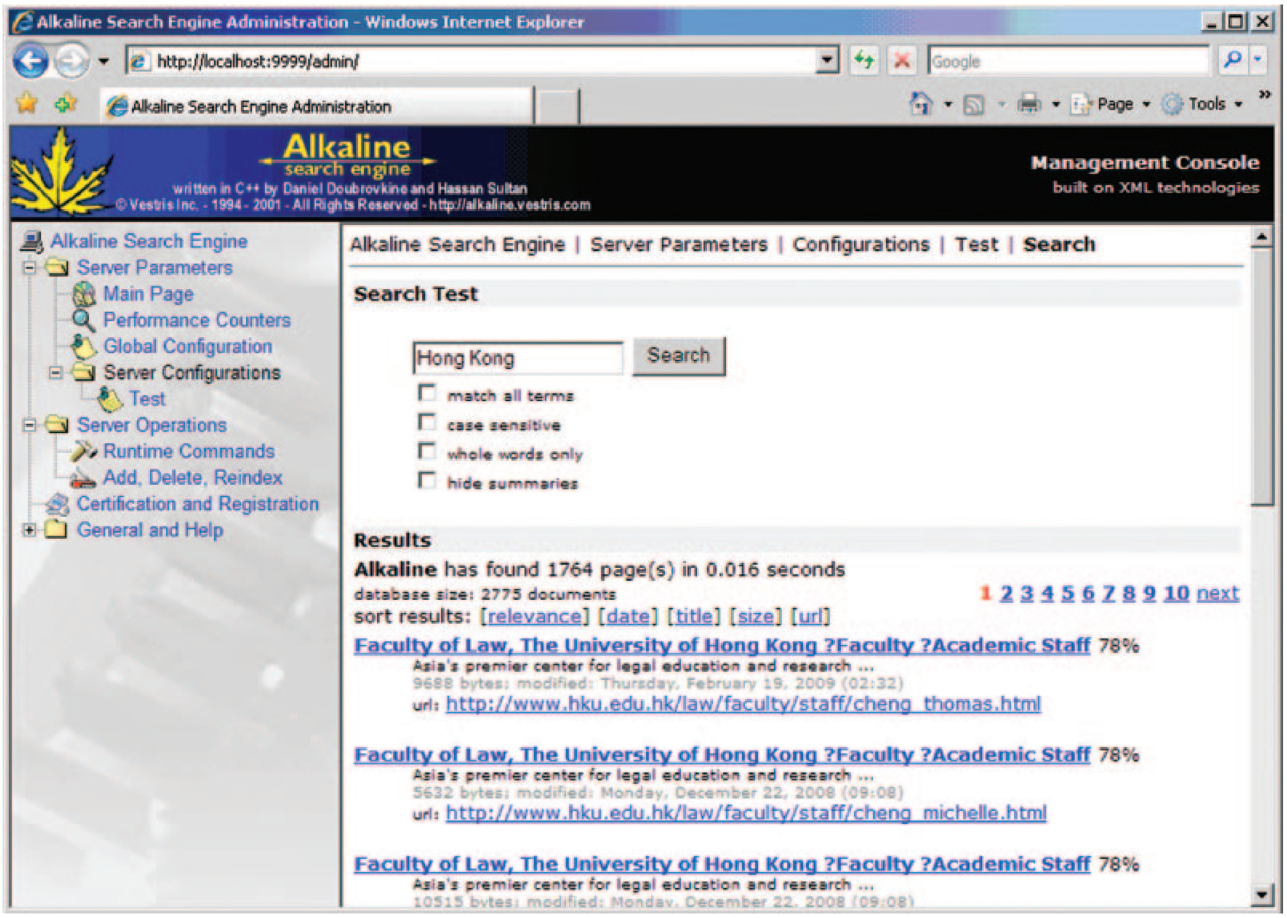
Overview of Greenstone. The main purpose of Greenstone is to organize existing information and make it maintainable, searchable, and browsable (Witten et al., 2000). Greenstone provides interfaces in multiple languages such as English, French, Spanish, Russian, and Kazakh. It also supports multiple platforms including Windows, Linux, and Mac OS. Figure 3, extracted from Greenstone's developer's guide, illustrates the implementation architecture (Bainbridge, McKay, & Witten, 2004). To use Greenstone to build a search engine, one can either use a preestablished collection or download a collection from the Web or from a digital library database. Because Web spidering is not the focus of this toolkit, implementation of downloading function is quite simple. Once the collections are imported into Greenstone, word index can be built to facilitate search function. Greenstone supports multiple types of documents including plain text, word, and PDF files. Additional document types can be added with implementation of additional plugins. It indexes multimedia files if appropriate metadata are available. Browsing function is another feature of Greenstone with appropriate document classification and hierarchy structure identified in metadata. End users can access indexed collections directly through Greenstone Web server. A developer can also write his or her own interface to connect to Greenstone Web server. Readers can refer to Bainbridge, McKay and Witten (2004) for detailed technical specifications.

Sample user session of Greenstone. Figure 4 presents the user interface creating new collections in Greenstone. First the user creates a new collection and specifies the seed URLs to be used. Greenstone offers a range of options in designing the index of documents. After specifying the seed URLs and the search options the user can start the spidering and indexing process.

After the pages have been downloaded and indexed, the user will see a page summarizing the details, such as the number of pages downloaded, of the collection. The user can then search in the collection using the search box provided on the page. A sample search result page is shown in Figure 5.

SpidersRUs

Overview of SpidersRUs. SpidersRUs was developed by the Artificial Intelligence Lab, a research group at the University of Arizona. It was designed to provide modular


Alkaline Search Engine Administration - Windows Internet Explorer
http://localhost:9999/admin/
Google
Alkaline Search Engine Administration
Page
Tools
Alkaline
search engine
written in C++ by Daniel Doubrovkine and Hassan Sultan
© Vestris Inc. - 1994 - 2001 - All Rights Reserved - http://alkaline.vestris.com
Management Console
built on XML technologies
Alkaline Search Engine
Server Parameters
Main Page
Performance Counters
Global Configuration
Server Configurations
Test
Server Operations
Runtime Commands
Add, Delete, Reindex
Certification and Registration
General and Help
Alkaline Search Engine | Server Parameters | Configurations | Test | Search
Search Test
Hong Kong
Search
match all terms
case sensitive
whole words only
hide summaries
Results

Alkaline has found 1764 page(s) in 0.016 seconds
database size: 2775 documents
1 2 3 4 5 6 7 8 9 10 next
sort results: [relevance] [date] [title] [size] [url]

Faculty of Law, The University of Hong Kong ?Faculty ?Academic Staff 78%
Asia's premier center for legal education and research ...
9688 bytes: modified: Thursday, February 19, 2009 (02:32)
url: http://www.hku.edu.hk/law/faculty/staff/cheng_thomas.html

Faculty of Law, The University of Hong Kong ?Faculty ?Academic Staff 78%
Asia's premier center for legal education and research ...
5632 bytes: modified: Monday, December 22, 2008 (09:08)
url: http://www.hku.edu.hk/law/faculty/staff/cheng_michelle.html

Faculty of Law, The University of Hong Kong ?Faculty ?Academic Staff 78%
Asia's premier center for legal education and research ...
10515 bytes: modified: Monday, December 22, 2008 (09:08)

FIG. 2. Searching the Test collection in Alkaline.

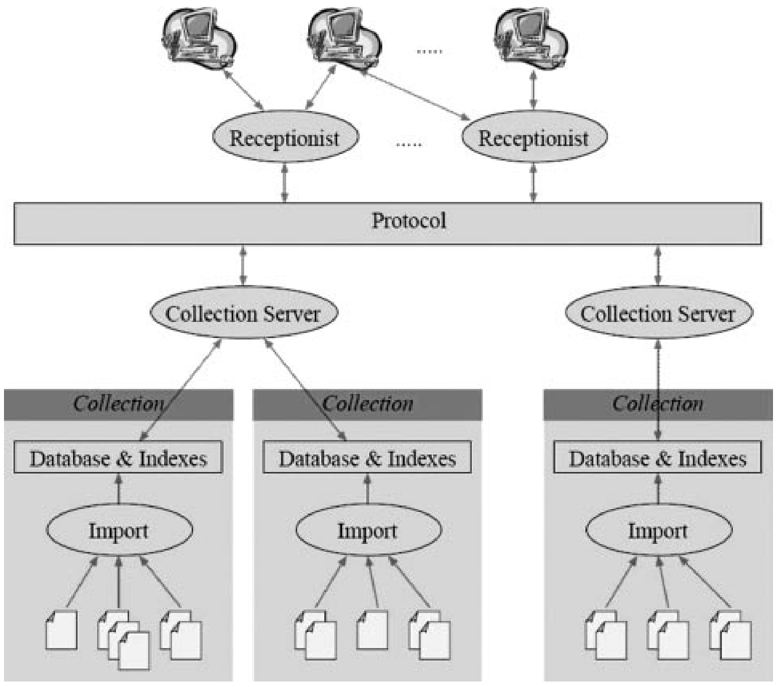
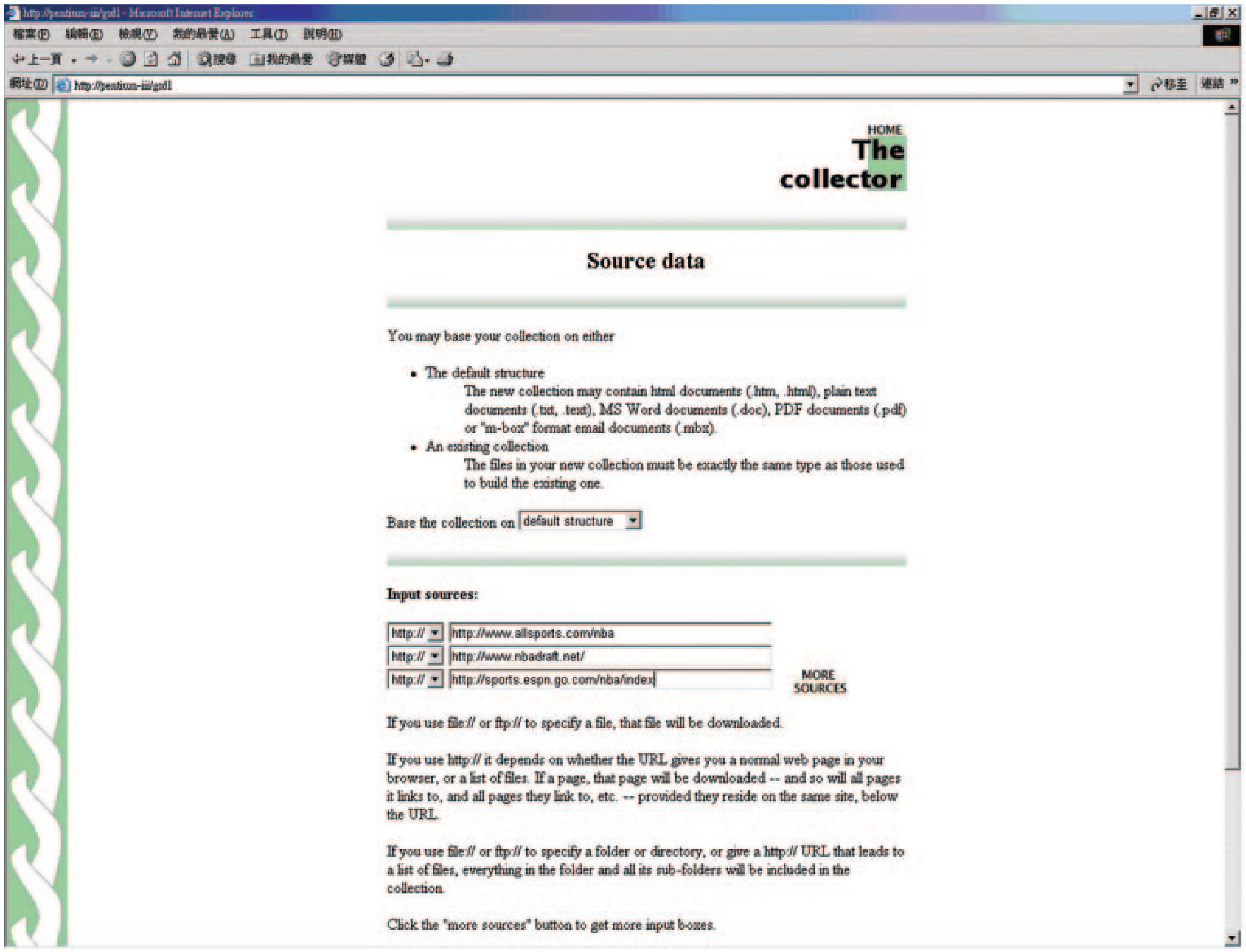


FIG. 3. System architecture of Greenstone (Bainbridge, McKay, & Witten, 2004).



Source data

You may base your collection on either

- The default structure
  The new collection may contain html documents (.htm, .html), plain text documents (.txt, .text), MS Word documents (.doc), PDF documents (.pdf) or "m-box" format email documents (.mbx).
- An existing collection
  The files in your new collection must be exactly the same type as those used to build the existing one.

Base the collection on default structure

Input sources:

http:// http://www.allsports.com/nba
http:// http://www.nbadraft.net/
http:// http://sports.espn.go.com/nba/index

MORE SOURCES

If you use file:// or ftp:// to specify a file, that file will be downloaded.

If you use http:// it depends on whether the URL gives you a normal web page in your browser, or a list of files. If a page, that page will be downloaded -- and so will all pages it links to, and all pages they link to, etc. -- provided they reside on the same site, below the URL.

If you use file:// or ftp:// to specify a folder or directory, or give a http:// URL that leads to a list of files, everything in the folder and all its sub-folders will be included in the collection.

Click the "more sources" button to get more input boxes.

FIG. 4. Creating a new collection in Greenstone.

tools for building digital libraries in multiple languages in a simple way.

SpidersRUs was written in the Java programming language. There are three reasons for this particular choice. First, Java is platform-independent, which implies that any programs written in Java can run on all computer platforms that support Java Virtual Machine (JVM). Second, Java supports multiple natural language encodings. This is made possible by the use of double-byte character in Unicode to store each character. Such a standard is especially useful for developing multilingual systems (Czarnecki & Deitsch, 2001). Last, it has been suggested that Java is suitable for building search engine development tools (Heydon & Najork, 1999).

In addition, all intermediate files of SpidersRUs, such as Web pages and search indexes, are stored as files (e.g., text files, html files). By this way all the files can be accessed easily by the users.

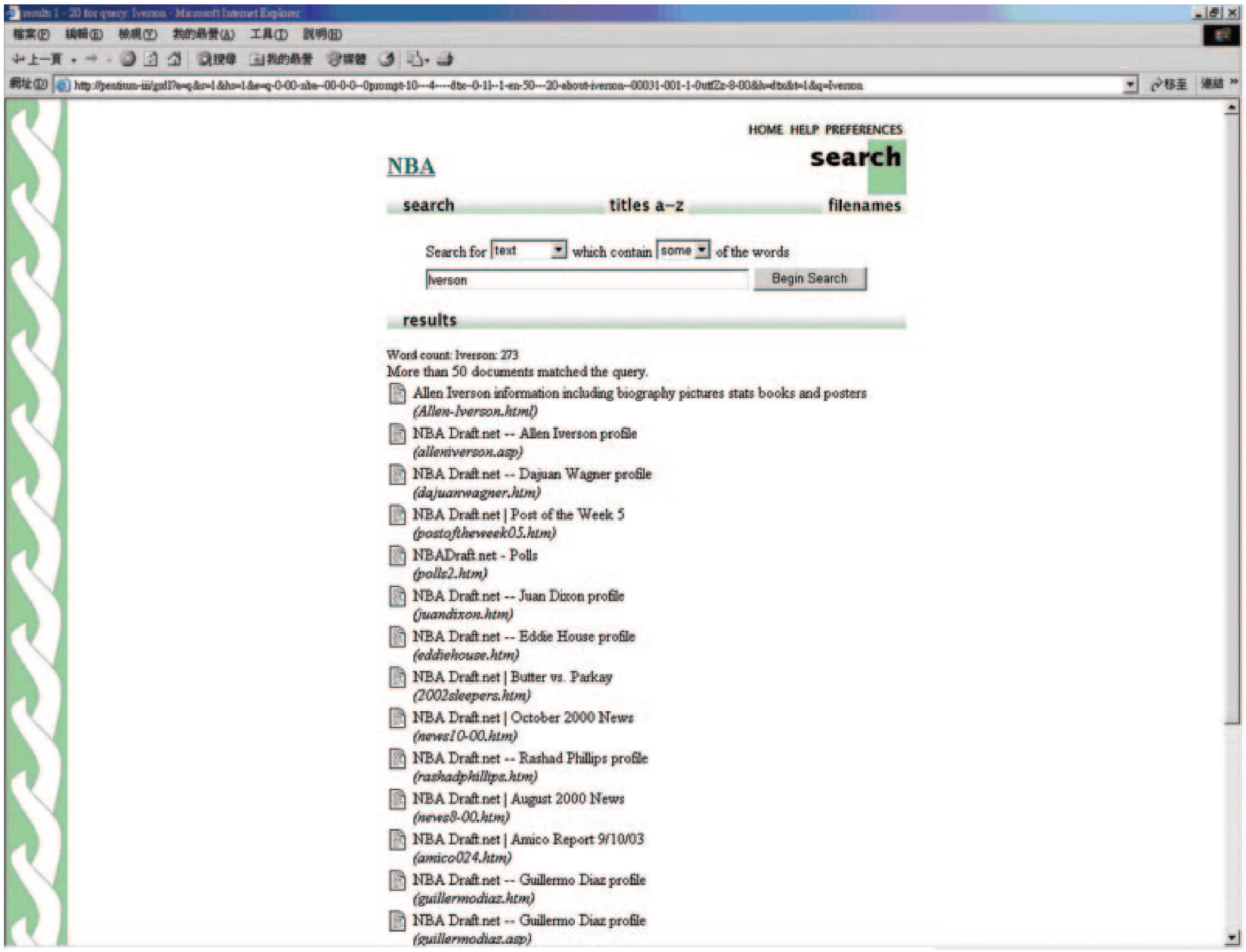
The architecture of SpidersRUs is depicted in Figure 6, and then the main components are discussed.

Spider: The Spider component comprises a prespecified number of "small spiders" that are controlled by a "spider master." Initially, each small spider is assigned an URL that links to the document to be downloaded. All downloaded documents are checked to avoid duplicates, which may otherwise result in duplicated links in the search results. Each valid document will be assigned a unique ID and stored to the local disk as the Spidered Files.

Indexer: The main role of the Indexer is to create an index about the terms extracted from the list of documents in the Spidered Files. First, each document is converted to plain text format. A preliminary index will then be created for all the documents. Words are extracted from the documents to give a word index that specifies the relative position and frequency of each word.

After the preliminary index has been created, it goes through a sorting process, which converts the preliminary index into an inverted index. Note that all index files are stored as plain text for easier reuse and access by the search engine developers, and that these files are stored in their original language encoding.

Query Engine: In addition to basic single-keyword searching, the Query Engine component supports Boolean searching (i.e., "AND", "OR" and "NOT") and phrase searching (specified using double quotes). Search results are ranked according to the frequencies of the matching keywords as indicated in the search indexes.


results 1 - 20 for query: Iverson - Microsoft Internet Explorer
http://pentium-iii/gsdl?a=q&r=1&hs=1&e=q-0-00-nba--00-0-0--0prompt-10---4----dtx--0-1l--1-en-50---20-about-iverson--00031-001-1-0utfZz-8-00&h=dtx&t=1&q=Iverson
HOME HELP PREFERENCES
NBA
search
search
titles a–z
filenames
Begin Search
results

Search for text which contain some of the words

Iverson

Word count: Iverson: 273
More than 50 documents matched the query.

- Allen Iverson information including biography pictures stats books and posters *(Allen-Iverson.html)*
- NBA Draft.net -- Allen Iverson profile *(alleniverson.asp)*
- NBA Draft.net -- Dajuan Wagner profile *(dajuanwagner.htm)*
- NBA Draft.net | Post of the Week 5 *(postoftheweek05.htm)*
- NBADraft.net - Polls *(polls2.htm)*
- NBA Draft.net -- Juan Dixon profile *(juandixon.htm)*
- NBA Draft.net -- Eddie House profile *(eddiehouse.htm)*
- NBA Draft.net | Butter vs. Parkay *(2002sleepers.htm)*
- NBA Draft.net | October 2000 News *(news10-00.htm)*
- NBA Draft.net -- Rashad Phillips profile *(rashadphillips.htm)*
- NBA Draft.net | August 2000 News *(news8-00.htm)*
- NBA Draft.net | Amico Report 9/10/03 *(amico024.htm)*
- NBA Draft.net -- Guillermo Diaz profile *(guillermodiaz.htm)*
- NBA Draft.net -- Guillermo Diaz profile *(guillermodiaz.asp)*

FIG. 5. User search result page in Greenstone.

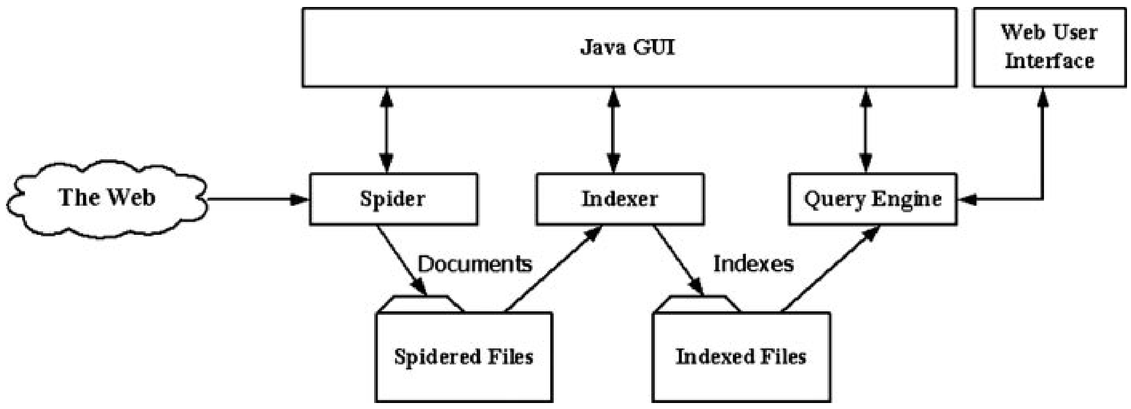
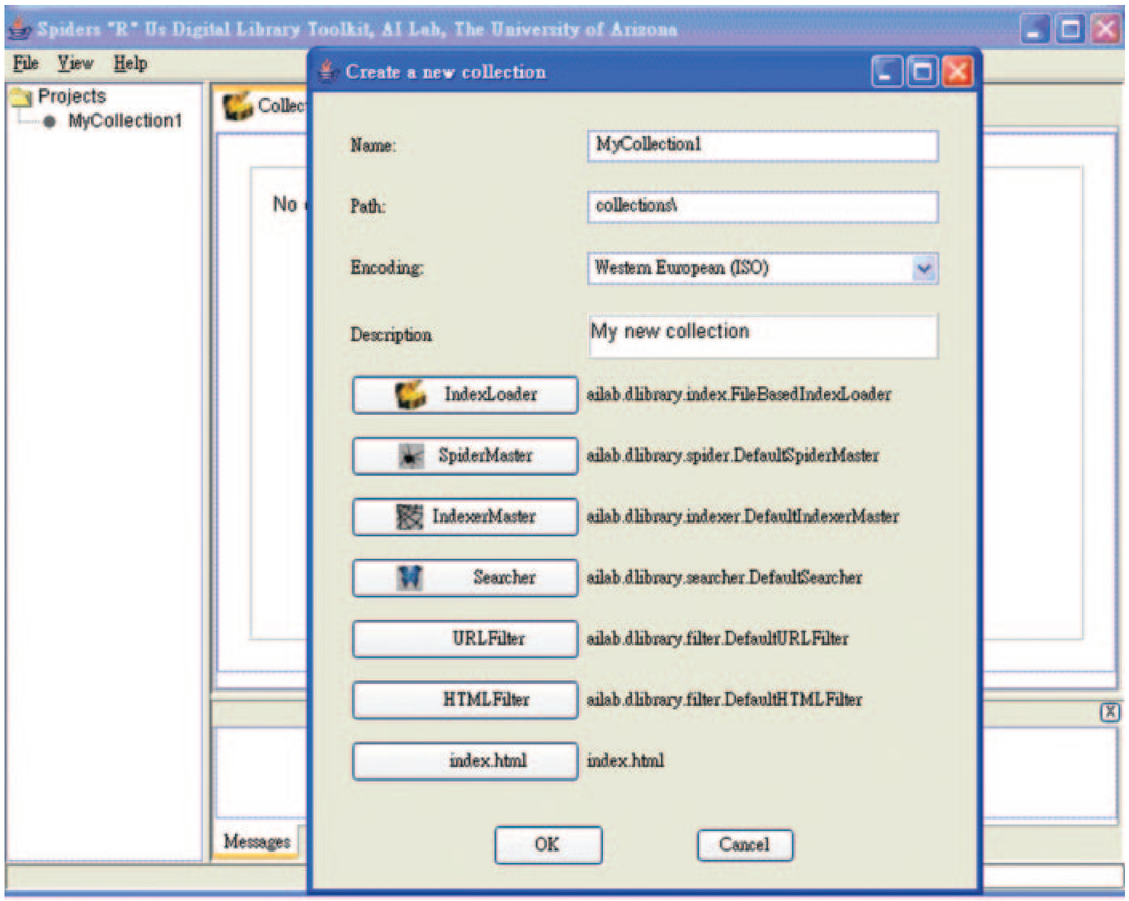


FIG. 6. System architecture of SpidersRUs.

Sample user session of SpidersRUs. The SpidersRUs toolkit provides the necessary components for developing search engines efficiently. This section describes each of the steps involved in the development. First, a user (i.e., a search engine developer) needs to create a new project for storing a collection of documents (see Figure 7). A new project can be created by selecting either "New" or "Advanced New" under the "File" menu in the toolbar. Both methods allow the user to specify the name, path, language encoding, and a short description of the collection. In the "Advanced New" option, the user can also specify settings for various components of the toolkit, which can include components developed by the users themselves.

After a new collection has been created, the project name will be added to the list of projects indicated on the left panel together with any existing projects. On the "Collection" tab



Name: MyCollection1
Path: collections\
Encoding: Western European (ISO)
Description: My new collection
IndexLoader: ailab.dlibrary.index.FileBasedIndexLoader
SpiderMaster: ailab.dlibrary.spider.DefaultSpiderMaster
IndexerMaster: ailab.dlibrary.indexer.DefaultIndexerMaster
Searcher: ailab.dlibrary.searcher.DefaultSearcher
URLFilter: ailab.dlibrary.filter.DefaultURLFilter
HTMLFilter: ailab.dlibrary.filter.DefaultHTMLFilter
index.html: index.html

FIG. 7. Creating a new project in SpidersRUs.

inside the right panel, the user will see a list of basic information of the collection such as number of items collected and indexed. As the Spider component has not been started, these values should be 0 at this stage.

The next step is to add the seed URLs to the Spider component. This can be done by first clicking the “Add Seeds” button on the “Spidering” tab. A list of seed URLs can be typed in one by one or by specifying a text file containing the list. If users want to specify other parameters for spidering, then they can click on the “Advanced” button. In the pop-up window, users can change the settings for the following parameters:

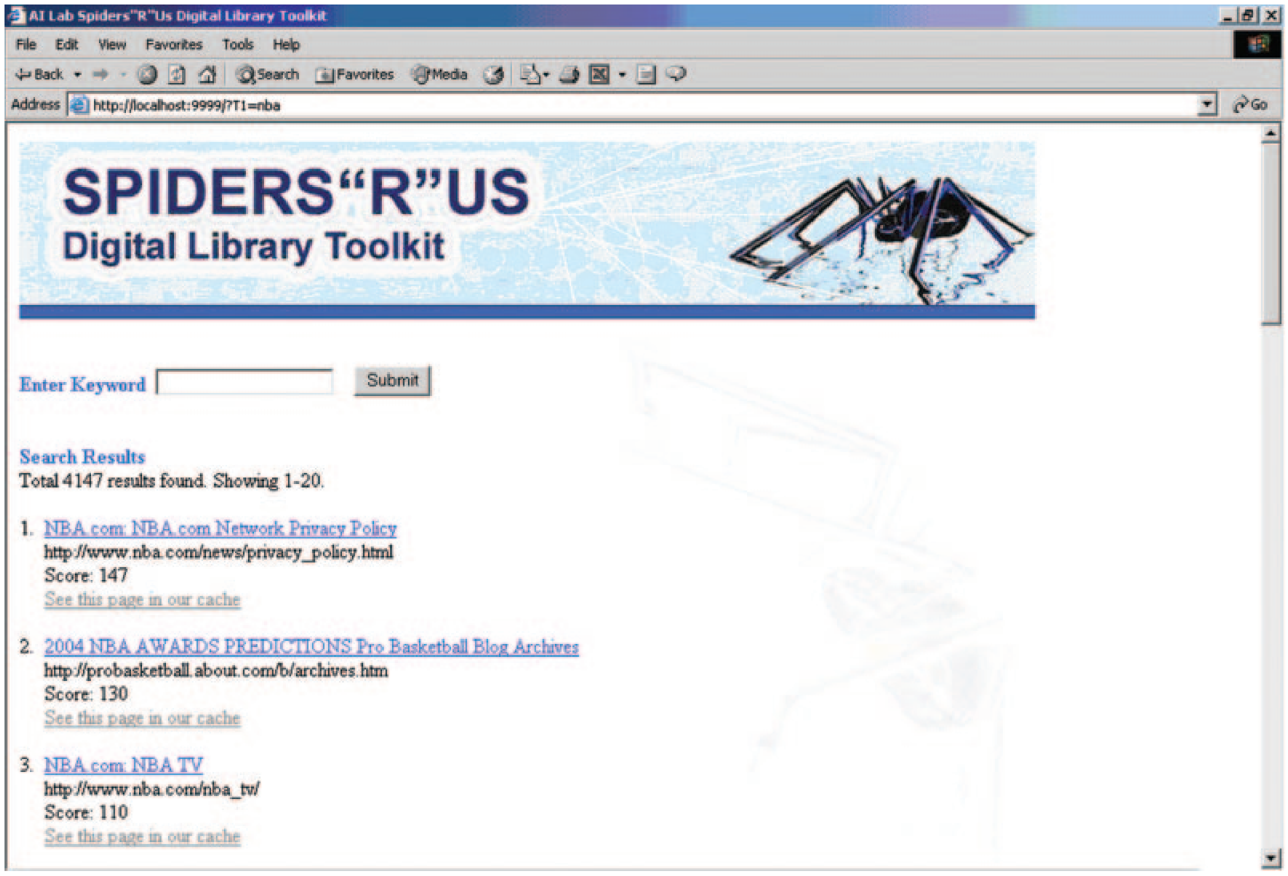
1. Number of spiders: number of threads used to download documents from the Web simultaneously.
2. Maximum levels: maximum depth that each spider will visit starting from the seed URLs
3. Timeout (sec): Maximum time allowed for downloading any document
4. Pages required: Number of pages to be downloaded
5. Only in same Web site: To limit the spiders to download documents only in the Web domain specified in the seed URLs.
6. Observes robots.txt exclusion: To specify that the tool will observe the Robot Exclusion Protocol.
7. Use Proxy Server: To specify the details of proxy server, if any.

After the user has added all seed URLs and set all parameters, the Spider component can be executed by clicking the “Start” button. The Spider component stops when it has downloaded enough documents (as specified in settings), or the spidering process has been manually stopped by the user. The user can then start the forward indexing process by clicking on the “Start Indexing” button on the “Indexing” tab. After that, the user can continue with generating the inverted index using the “Start Sorting” button.

After the search indexes have been generated, the search engine is ready to serve. If the user does not specify a port number, the default port number will be assigned. The search engine can be started by clicking the “Start Service” button on the “Start Service” tab. After that, the search service can be accessed through a Web browser by clicking the “Launch Browser” button. Any users can then submit queries to the search engine through the Web interface and start the search. A sample search result screen is shown in Figure 8.

Evaluation

To study the effectiveness of our class project and compare the educational effectiveness with the three search engine building tools, every student was asked to complete a survey at the end of the project. The survey was related to experience and opinion of using each of the tools, including such


AI Lab Spiders"R"Us Digital Library Toolkit
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:9999/?T1=nba
Go
SPIDERS"R"US
Digital Library Toolkit
Enter Keyword
Submit

Search Results
Total 4147 results found. Showing 1-20.

1. NBA.com: NBA.com Network Privacy Policy
   http://www.nba.com/news/privacy_policy.html
   Score: 147
   See this page in our cache

2. 2004 NBA AWARDS PREDICTIONS Pro Basketball Blog Archives
   http://probasketball.about.com/b/archives.htm
   Score: 130
   See this page in our cache

3. NBA.com: NBA TV
   http://www.nba.com/nba_tv/
   Score: 110
   See this page in our cache

FIG. 8. Sample session of the search engine built using SpidersRUs.

areas as user friendliness, system capability, as well as overall impression of the tools. In addition, items about the knowledge gained from using each of the tools were included. For each item, students had to give scores to the tools respectively using a 0 to 9 scale, with 9 indicating the best score. In the following, we will focus on two areas of our findings, which are overall reactions to the tools (Chin, Diehl, & Norman, 1988) and knowledge gained from the tools (Chen, Fan, Chau, & Zeng, 2003). The corresponding items are shown in the Appendix.

Overall Reactions to the Tools

The results of the survey concerning overall reactions to the tools are summarized in Tables 1 and 2. As shown in Table 1, SpidersRUs obtained the highest mean score in each of the six areas (i.e., user friendliness, being interesting, ease of use, powerfulness, flexibility, and overall impression). Pairwise t tests were further conducted to compare the three tools in these areas. Table 2 suggests that SpidersRUs was significantly better than Greenstone in all the six areas. In addition, SpidersRUs' scores were comparable to those of Alkaline, except for the attributes of being interesting and ease of use, where SpidersRUs was significantly better than Alkaline.

TABLE 1. Overall reactions to the tools: Mean scores.

	Alkaline	Greenstone	SpidersRUs
User friendliness	5.500	2.429	6.000
Being interesting	5.071	2.679	5.964
Ease of use	5.143	4.250	6.519
Powerfulness	5.741	4.393	6.222
Flexibility	5.778	3.821	5.893
Overall impression	5.750	2.321	6.071

TABLE 2. Overall reactions to the tools: p values of pairwise t test comparison.

	Alkaline vs. SpidersRUs	Greenstone vs. SpidersRUs
User friendliness	0.270	$<0.001^{***}$
Being interesting	0.034^{*}	$<0.001^{***}$
Ease of use	0.011^{*}	0.001^{**}
Powerfulness	0.199	$<0.001^{***}$
Flexibility	0.834	0.001^{**}
Overall impression	0.375	$<0.001^{***}$

$^{*}<0.05$, $^{**}<0.01$, $^{***}<0.001$.

Knowledge Gained From Using The Tools

In terms of knowledge gained from the tools, SpidersRUs scored highest among all six areas of knowledge (i.e., development of search engines, architecture of search engines,

TABLE 3. Knowledge gained from the tools: Mean scores.

Area of knowledge	Alkaline	Greenstone	SpidersRUs
Development of search engines	5.964	4.714	6.429
Architecture of search engines	5.464	4.821	5.929
World Wide Web	5.107	4.750	5.607
Web application development	4.821	4.286	5.643
Application design	5.107	4.643	5.750
Overall knowledge gained	6.107	4.929	6.500

TABLE 4. Knowledge gained from the tools: *p*-values of pairwise *t* test comparison.

Areas of knowledge	Alkaline vs. SpidersRUs	Greenstone vs. SpidersRUs
Development of search engines	0.263	0.001**
Architecture of search engines	0.278	0.027*
World Wide Web	0.233	0.064
Web application development	0.034*	0.003**
Application design	0.092	0.006**
Overall knowledge gained	0.338	0.002**

World Wide Web, Web application development, application design, and overall knowledge gained); Table 3). Similarly, pairwise *t* tests were used to compare the statistical significance of these scores. As indicated in Table 4, SpidersRUs was significantly more useful than Greenstone in helping students to gain knowledge in almost all six areas, except for the area of World Wide Web, in which no significant difference was found. As for the comparison between Alkaline and SpidersRUs, a significant difference was seen only in the area of Web application development.

We suggest that the students were able to gain more knowledge from SpidersRUs and Alkaline than from Greenstone because of two reasons. First, Greenstone is more "technical-oriented" and its user interface appears to be more complex than the other tools. Students, thus, input only the required parameters without fully understanding the details of the search engine development process. On the other hand, SpidersRUs and Akaline are more user-friendly and allow students to monitor and understand the development progress more easily, thus improving their understanding of the related areas. Another possible reason for the higher levels of knowledge gained in using SpidersRUs is the modular design of the tool. Because each component is more clearly separated from each other, students were able to learn the details of each component more easily, and thus a better overall understanding.

Comments From Students

After all the three versions of search engines had been developed, we asked each group of students to compare the three tools based on their experience. In particular, the following comments were obtained from the group that built the NBA Search Engine shown in Figure 8. First, Alkaline was the most difficult to use among the three tools as it could run only in command prompt and the parameters, such as seed URLs, and number of required pages had to be specified using a configuration file. According to the students, it was a tedious process for them to configure the tool, although the large number of available parameters had offered the largest degree of flexibility in customization.

As for Greenstone, the students said it was the least flexible tool because some of the parameters such as number of spiders were either not available or required comprehensive study of documentation. In addition, Greenstone required the longest building time because all types of files in a Web page, including photo and sound, had to be downloaded before the search engine could be built.

The third tool, SpidersRUs, provided a user-friendly interface and offered some useful features such as importing seed URLs as a text file. Moreover, the search engine built using SpidersRUs had a relatively high search speed and a well-designed layout. However, the students added that SpidersRUs was not a very efficient tool because it took a long time (about 5 minutes) to start the search engine. Another concern was the large size of the collection (i.e., search indexes), which sized over 500 megabytes for the NBA Search Engine.

In general, the group explained that all the three tools offered a similar set of components needed to build a search engine and provided similar search results (except for Greenstone, which gave more limited results as a smaller number of seed URLs were used). In terms of building time, Alkaline was the fastest and Greenstone was the slowest.

Conclusions

Implications

In this article, we have discussed a project that was carried out with an intention to compare the educational value of the three search engine development tools, namely, Alkaline, Greenstone and SpidersRUs.

From the students' comments obtained after the project, we observed that SpidersRUs had certain features that distinguished it from the other two tools. In particular, it offered a well-designed graphical user interface that was not available in Alkaline. Moreover, it supported searching documents written in multiple languages by utilizing its Java-based platform. According to the survey, SpidersRUs was ranked favorably by most students who had used all the three tools. In terms of knowledge gained from the tools, it outperformed the others for areas including search engine development and architecture, the World Wide Web, application design, and Web application development. In terms of reactions to the tools, not only were its mean scores higher than the others, but it also was significantly better than Greenstone. In summary, the results showed that SpidersRUs was, among the three tools, the most favorable one and the best in helping students gain the necessary skills and knowledge on topics such as World Wild Web and application development.

TABLE 5. Comparison of three search engine building tools.

	Alkaline	Greenstone	SpidersRUs
Main purpose	Building small-mid scale search engines	Organizing existing information	Building small-mid scale search engines
Development language	C++	C++, Java	Java
Collection Building (spidering)	Basic Web spidering functions	Customizable Web spidering functions	Customizable Web spidering functions
Parsing/Indexing Function	Parses plain texts, HTML, PDF, MS Word, MS Rich Text, LaTex, Word Perfect, and multimedia files (only metadata is indexed for multimedia files)	Parses plain texts, HTML, and multimedia files (only metadata is indexed for multimedia files)	Parses plain texts, HTML, PDF, MS Word, MS Rich Text.
Searching	Supports Boolean, metadata, and numeric search	Supports Boolean expressions and phrase search	Supports Boolean expressions and phrase search
User Interface	GUI not available for the collection building process	Web-based Interface	Client-side GUI
Platform Supported	Windows, Linux, Mac OS, Unix	Windows, Linux, Mac OS	Windows
Human Languages Supported	English	All UTF supported languages	All UTF supported languages

A summary of the main features of the three search engine development tools is provided in Table 5. We can see that each tool has its own features that may be suitable for different purposes. Based on the experimental results discussed above, it is preferable to use SpidersRUs for a class project in terms of helping students obtain knowledge in search engine and the Web. On the other hand, if the purpose is to provide students with hands-on experience on command-prompt tools, then Alkaline may be a better choice. If the main teaching purpose is to explore non-English systems, then Greenstone and SpidersRUs will be more appropriate. For courses with a digital library and information management orientation, Greenstone may be a better choice because it was designed with this perspective.

In general, the evaluation results show that the students were able to gain more knowledge about different areas of search engines and the Web from using the three tools in the project. These areas include the development of search engines, the architecture of search engines, the Web, Web application development, and application design. Because many of these topics are important for computer science and information systems curricula, we believe that this search engine development project is useful in IT education.

We suggest that the application of this project may need to be adjusted according to the nature and timeframe of the adopting course. To achieve the best learning result for a course focusing on applications and system analysis, we suggest that this project can be applied in the same way as discussed in this article. This will allow students to gain useful knowledge about the Web and search engines. On the other hand, for a course focusing on programming, a project that emphasizes more on coding (e.g., Chau et al., 2003) would be more suitable.

Future Work

There are some areas of SpidersRUs that can be improved. First, the building and starting process of search engines can be made more efficient. This problem may be caused by the large size of downloaded files and indexed files to be processed. In addition, some students suggested that the tool could be made more powerful if there were more configurable options available for the users.

In addition, all the tools discussed only have limited capabilities for dynamic Web pages, which are becoming increasingly popular. It is important to continue the research on Web spider for dynamic pages. This will allow students to create search engines for different sites, e.g., social networking sites more easily, thus enabling them to get more knowledge about different Internet-related topics through the search engine project.

Acknowledgment

This project has been supported in part by funding from the HKU Seed Funding for Basic Research (#10207565, #10208140) and the NSF National SMETE Digital Library Program (DUE-0121741).

References

Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., & Raghavan, S. (2001). Searching the Web. ACM Transactions on Internet Technology, 1(1), 2–43.

Bainbridge, D., McKay, D., & Witten, I.H. (2004). Greenstone digital library developer's guide. Retrieved October 14, 2009, from University of Waikato, New Zealand, Department of Computer Science Web site: http://www.greenstone.org/developers-guide

Bird, S., & Curran J.R. (2006). Building a search engine to drive problem-based learning. In Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, Bologna, Italy.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. In Proceedings of the 7th International Conference on the World Wide Web (WWW 07) (pp. 107–117). Amsterdam, Elsevier Science.

Chau, M., & Chen, H. (2003). Personalized and focused Web Spiders. In N. Zhong, J. Liu, & Y. Yao (Eds.), Web intelligence (pp. 197–217). Heidelberg, Germany: Springer-Verlag.

Chau, M., Fang, X., & Yang, C.C. (2007). Web searching in Chinese: A study of a search engine in Hong Kong. Journal of the American Society for Information Science and Technology, 58(7), 1044–1054.

Chau, M., Huang, Z., & Chen, H. (2003). Teaching key topics in computer science and information systems through a search engine project. ACM Journal on Educational Resources in Computing (JERIC), 3(3), 1–14.

Chau, M., Qin, J., Zhou, Y., Tseng, C., & Chen, H. (2008). SpidersRUs: Creating specialized search engines in multiple languages. Decision Support Systems, 45(3), 621–640.

Chen, H., Fan, H., Chau, M., & Zeng, D. (2003). Testing a cancer meta spider. International Journal of Human-Computer Studies, 59(5), 755–776.

Cheong, F.C. (1996). Internet agents: Spiders, wanderers, brokers, and bots. Indianapolis, IN: New Riders Publishing.

Chin, J., Diehl, V., & Norman, K. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. Proceedings of the SIGCHI conference on Human Factors in Computing Systems (pp. 213–218). New York: ACM Press.

Czarnecki, D., & Deitsch, A. (2001). Java internationalization. Sebastopol, CA: O'Reilly & Associates.

Dutt, J. (1994). A cooperative learning approach to teaching an introductory programming course. Proceedings of the 9th International Academy for Information Management (pp. 225–230). Location unknown: International Academy for Information Management.

Granger, M., & Lippert, S. (1999). Peer learning across the undergraduate information systems curriculum. Journal of Computers in Mathematics and Science Teaching, 18(3), 267–285.

Harris, A.L. (1995). Developing the systems project course. Journal of Information Systems Education, 6(4), 192–197.

Heydon, A., & Najork, M. (1999). Mercator: A scalable, extensible Web crawler. World Wide Web, 2(4), 219–229.

Hickey, T., Kumar, A., Wilkens, L., Beiderman, A., Mahadev, A., & Ellis, H. (2002). Internet-centric computing in the CS curriculum. Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education (pp. 50–51). New York: ACM Press.

McConnell, J. (1996). Active learning and its use in computer science. Proceedings of the SIGCSE/SIGCUE Conference on Integrating Technology into Computer Science Education (pp. 52–54). New York: ACM Press.

Poindexter, S. (2003). Assessing active alternatives for teaching programming. Journal of Information Technology Education, 2, 257–266.

Witten, I.H., Bainbridge, D., & Boddie, S.J. (2001). Greenstone: Open-source DL software. Communications of the ACM, 44(5), 47.

Witten, I.H., McNab, R.J., Boddie, S.J., & Bainbridge, D. (2000). Greenstone: A comprehensive open-source digital library software system. Proceedings of the ACM Digital Libraries Conference (pp. 113–121). New York: ACM Press.

Appendix

Survey Items in the Questionnaire

Overall Reactions to the Tool	
– (terrible/wonderful)	terrible wonderful 0 1 2 3 4 5 6 7 8 9
– (frustrating/satisfying)	frustrating satisfying 0 1 2 3 4 5 6 7 8 9
– (dull/stimulating)	dull stimulating 0 1 2 3 4 5 6 7 8 9
– (difficult/easy)	difficult easy 0 1 2 3 4 5 6 7 8 9
– (inadequate power/adequate power)	inadequate adequate 0 1 2 3 4 5 6 7 8 9
Knowledge Gained from the Tool	
– Is the system helpful in your learning process in the course	unhelpful helpful 0 1 2 3 4 5 6 7 8 9
– You have learned more about the process of search engine development after using the system	not at all very much 0 1 2 3 4 5 6 7 8 9
– You have learned more about the design and architecture of search engines after using the system	not at all very much 0 1 2 3 4 5 6 7 8 9
– You have gained more knowledge about the World Wide Web after using the system	not at all very much 0 1 2 3 4 5 6 7 8 9
– You have learned more about Web application development after using the system	not at all very much 0 1 2 3 4 5 6 7 8 9
– You have learned more about application design after using the system	not at all very much 0 1 2 3 4 5 6 7 8 9