Supporting Classroom Information Management with SCOUT

Quranna Khan[♣] D. Scott McCrickard[♦] Sherian Clay[♣]
[♣]Computer Science Department
Hampton University
Hampton, VA
cskhan@hotmail.com
[♦]Georgia Institute of Technology
College of Computing
Atlanta, GA 30332-0280
mccricks@cc.gatech.edu sherianc@hotmail.com

INTRODUCTION

The classroom of the new millennium will not look like the pencil-and-paper versions that we remember growing up. New technology promises to capture the information exchanged in a class and to make it accessible at later times. Blackboards and whiteboards will be replaced with liveboards and smartboards, cameras will be placed in the classroom, and students will use laptops and hand-held computers to ask questions and take notes. However, collecting the information is only a small part of the problem. Somehow, students and faculty must be able to manage this information and stay abreast of updates and changes.

This problem has already found its way into the classroom because of the existence of course sites on the World Wide Web. A course Web site is a set of Web pages that typically contains information about class notes, assignment specifications, and examination overviews. In the not-so-distant future, these sites will contain recordings of the lectures, reports of class discussions, and other information. Professors could simply point students to the Web page and tell them to keep up with the notes and assignments for the term.

The Web site for a course is a dynamic entity. Pages are continually added, and existing pages are constantly updated. Due dates get moved, specifications for programs are re-specified, and just in general professors have some kind of change of heart or mind about something previously written on their tentative schedule. Keeping up with the changes is a continual challenge for students, and few tools have been developed to help them.

'99 ACM Southeast Regional Conference

While the indexing and searching problems have been widely studied for many years (resulting in solutions such as TFIDF [4]), the explosion of Web use has provided new forums for their use. Several systems have demonstrated the viability of TFIDF for Web-based tasks such as citation indexing and information surfing [1,3]. However, often the standard solutions must be augmented to meet the needs of a specific situation such as classroom information management.

We address these problems with an information management tool called SCOUT. SCOUT (Searching Course Objects Using TFIDF) allows the user to stay aware of the contents and changes in content that might be made to a Web site by providing information on recently modified pages, important upcoming dates, and keyword-based searches.
SCOUT combines widely used technology and solutions specific to course-related problems to create a tool capable of alleviating students' information management problems.

SCOUT

SCOUT provides students and faculty with the means necessary to stay abreast of changes to course Web sites. A user provides a course base URL, and SCOUT indexes the information at that page and at all linked pages that are in the course directory structure. The user can then perform three types of searches on the information:

- **Last modification** searches allow the user to search for pages at a class Web site that have been recently modified. Users can select the number of days since the last time they visited the site, and SCOUT will provide a list of all pages that have been modified in the time.

- **Calendar outlooks** identify pages that contain date references for future events. The user specifies a number of days, and SCOUT finds all date references for that number of days in the future. It is best used for finding homework due dates, midterms, and special events.

- **Keyword searches** work much like a search at Alta Vista, Excite, and other search engines. The user can type in a series of words, and SCOUT returns an ordered list of pages that it thinks are most relevant.

The user can select pages is the returned list to get information about the page, including its title, URL, and last modification time.

Perhaps the best way to demonstrate the potential uses of SCOUT is to examine some scenarios where it proves to be useful.

Last Modification Professor Smith is notorious for repeatedly changing assignment specifications on his course Web site throughout the school term. Deadlines are extended, page limits are changed, content descriptions are modified, all to the frustration of his students. One student has found a solution with SCOUT -- whenever she logs in, she types in her course URL and the approximate number of days since she last checked the site, and SCOUT provides a list of course Web pages that have been recently modified (see Figure 1).

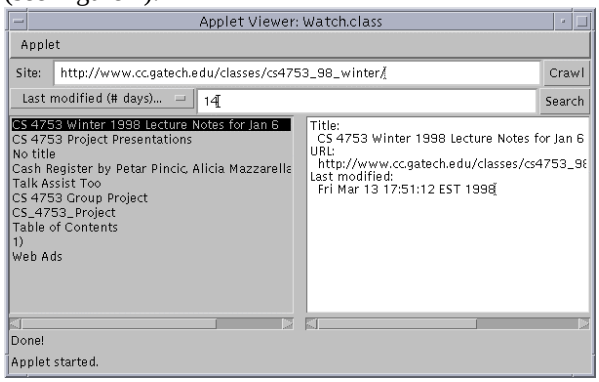

Figure 1: Results from a last modification search.

Calendar Outlook Assignment due dates, special talks, and scheduled midterms are typically stored on a course Web site. Since items are continually added, it is difficult to keep track of all of the important upcoming events. By checking the calendar outlook on SCOUT for a certain number of days, users can see what events have been planned and can adjust their schedules appropriately. For example, if on May 5 a user performs a calendar outlook for the next seven days, SCOUT would return a list of all date references between April 5 and April 12. SCOUT understands numerous abbreviations and shortcuts for date references, so pages containing April 7, Apr. 10, and Apr 8 would all be returned.

Keyword Search Professor Brown heads a discussion class that requires its students to write essays on topics and readings from class. One student wants to write an essay on hypertext authoring systems, a topic that has been discussed many times in class. Although the class summaries are provided at the Web site, it is difficult to remember which classes covered the topic of interest. After entering the search phrase “hypertext authoring systems” into SCOUT, it returns a list of class summary pages that contain one or more of the words. Best of all, the page list is ordered from best to worst match, so the student can start with the discussions that contain the most relevant information (see Figure 2).

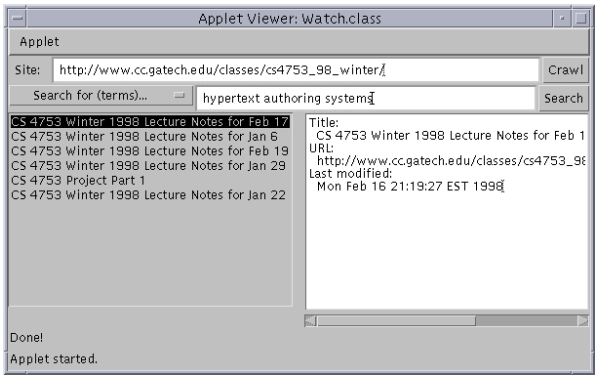

Figure 2: Results of a term search.

HOW SCOUT WORKS

SCOUT consists of a user interface, a site crawler, and a search engine. When a user enters a site into the user interface and presses the “Crawl” button, the site crawler creates an index of the information at the site. When the user selects a search type and enters the appropriate parameters, the search engine checks the index and returns a list of pages that matches the query.

To provide a robust and platform-independent system, we chose to implement SCOUT in Java. Java is Internet-aware, making it easy to access and download Web pages. Java's object-oriented nature allows us to expand the functionality of the system by adding and extending objects. The new functionality could provide advanced processing and filtering or could handle new sources of information like audio or video streams.

User Interface

The SCOUT user interface allows a user to crawl a course Web site and search the pages that it contains. Figures 1 and 2 show the interactors in the SCOUT interface. The “Site'” entry box allows a user to enter a course Web site. The crawl button indexes (or re-indexes) the contents of the indicated site. The search menu allows the user to choose between the different types of search. Each type requires a parameter that must be entered in the accompanying entry box. The two scrollable text boxes contain the results of the search. The left box contains the list of all pages that match the search. When the user selects a list entry, the right box shows information about the selected entry. The text message at the bottom provides feedback on the activities of the system.

Site Crawler

To collect the Web site information, we implemented a *site crawler* that crawls the pages directly relating to a course Web site. Given a base page, our site crawler recursively visits all linked pages that are contained in the base page's directory location. By limiting the crawling to pages under the base page, our crawler is prevented from visiting enormous numbers of pages that may not be relevant to the course. As an additional precaution, the crawler will only visit pages within five hops of the base page - if a user could not get to a page by visiting five links starting from the base, then that page will not be indexed.

For each page, after the vital statistics like the title and last modification time are noted, the words on the page are collected for indexing.
Each word is checked against a *stoplist* of commonly occurring words - if the word is in the stoplist, it is not indexed. Stoplisted words (words like “the”, “and”, and “that”) are acknowledged as poor index terms because they occur in almost every document and thus cannot be used to distinguish between documents. We use a stoplist of the most common words in the English language [2]. All words from the document not in the stoplist are added to a *document vector*, a sparse vector that indicates the number of times each word appears in the document. The document vector is used by the search engine to compute the similarity between a query and the document.

In addition to the document vector, the crawler maintains a list of references to dates.
The dates can be in a variety of different formats with various abbreviations for months and various punctuation styles. Days of the week and years are optional - if the year is missing the date is assumed to be for the current year. The list of dates is used by the search engine for calendar outlooks.

Search engine

The search engine accepts a search query from the user and returns a list of matching pages. For last modification searches, the engine returns a list of pages that have been modified in the indicated time period.
For calendar outlooks, it returns pages with a date reference in the passed range.

For keyword searches, the search engine uses a weighting formula known as TFIDF to calculate the similarity between each document and the query, and returns an ordered list of all non-zero results.

TFIDF (term frequency, inverse document frequency) is based on the assumption that a document is similar to a term if it has a high term frequency (the term appears often in the document) but the term has a low inverse document frequency (the term does not appear often in the set of all documents).

The formula $w_{ik} = f_{ik}\ log_2\ (n/d_k)$ is used to calculate the weight of term k in document i, where f_{ik} is the frequency of occurrences of term k in document i, n is the total number of documents, and d_k is the number of documents containing term k. The weights for each term in the query are summed to find the overall weight of the document for the query.

CONCLUSIONS AND FUTURE WORK

SCOUT provides methods to manage the changing and growing information resources that are artifacts of today's classroom. SCOUT not only provides the general searching mechanisms found in many search engines today, but it can supply up-to-the-minute indexing and specialized searches that uniquely attack the problems of course Web pages.

As the online information grows, the model provided by SCOUT should grow with it. Classroom audio and video can be textualized and indexed using a scheme like TFIDF. Just as specialized search mechanisms were added to handle date references and modification times, other such mechanisms may need to be added to handle other task-specific types of information. As technology invades the classroom, the colored notebook binder of years past must give way to new technologies capable of handing more than scribbled notes and paper handouts. We expect that SCOUT is a step in the right direction.

REFERENCES

1. K. Bollacker et al. CiteSeer: An autonomous Web agent for automatic retrieval and identification of interesting publications. In *Proceedings of Agents '98*, pages 116-123, 1998.

2. C. Fox. *A stop list for general text*. SIGIR Forum, 21 (2): 19-35, 1990.

3. H. Lieberman. Autonomous interface agents. In *Proceedings of SIGCHI '97*, pages 67-74, 1997.

4. G. Salton and M. McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.