Adaptive Content Delivery to Assist Blind Students in Accessing Course Materials

Mehmet E. Donderler, Lina Peng, and K. Selcuk Candan

Abstract

Currently, blind users can access information on the web only with the help of a screen reader program (such as Window Eyes and JAWS for Windows), which provides output in either audio or Braille format. Even though screen reader software is helpful, it is generally very difficult, as well as time consuming, to find a link of interest, or a piece of information on a Web page using a screen reader, since the user has to listen to the audio output from the screen reader for every link and textual content. Moreover, blind users cannot get an overview of the structure of the information presented in a Web page or a Web site at a glance on the screen as sighted people do. Therefore, it is much easier for blind users to get disoriented in a hypertext environment.

In this paper, we present a software system, called iCare-Assistant that attempts to address these challenges for blind students in accessing course materials from a course server, my.asu.edu, over the Internet at ASU. The iCare-Assistant provides unobtrusive, task-oriented, and individualized delivery of electronic course information, and handles the tasks of information integration, filtering, and modulation to minimize the information overload that renders existing educational interfaces unusable by blind students.

1. Introduction

With the passage of the 508 web accessibility mandate, many companies and federal government agencies are required to follow accessibility guidelines when designing their web sites. Such guidelines are very effective when designing mostly static and non-individualized information outlets. However, when the material being delivered is information rich, when it is dynamically generated through users' preferences and annotations, and when the users follow non-linear, individualized pathways through this material (as would be the case where a student is studying for an exam using her notes, multiple reference books, graded assignments, and project reports), the navigational challenge is compounded. Indeed, the blind students who are using the current prototype of our *iCare-Device*[1] have emphasized that, although the screen reader software enables them to access the electronic material that was not available to them before, they still have to struggle when accessing the course server, *my.asu.edu*. This is mainly due to the richly-structured, heterogeneous, and constantly growing content in the course server, which disorients blind students.

Courses presented on the web in hyperlinked form allow students to explore the content freely, based on their interests and goals. The learner is engaged in an activity to create both meaning and structure through her decisions with respect to which link to follow next. However, state-of-the-art browser-based interfaces [1-3] rely heavily on users' visual skills for information presentation and can result in significant navigational burden. Existing navigational helps, such as site maps and visual cues [4], on the other hand, alleviate this load for only sighted users. Furthermore, for blind students popular screen reader programs, such as Window Eyes and JAWS, are not always helpful in large hypertext documents with many links, as users must listen to the entire document.

We present a software system, called *iCare-Assistant* [2], to help blind students to easily, with a minimum cognitive navigational load, locate course materials on a dynamic network of information pages. Our goal of reducing the cognitive navigational load on blind students involves various technical challenges some of which are listed below:

- The system establishes, as precisely as possible, what information is needed, based on the current content, current context, access history, user preferences

[1] *iCare-Device* includes the integration of several components, such as a wireless handheld computer, cameras, microphones, natural language processor, text-to-speech device, digital audio recorder, and refreshable Braille device for accessing educational materials.

[2] Supported as part of a recently awarded NSF-ITR grant.

and information, and the hyperlink structure, and present this information to the student in the proper format.
- Logical relationships between various related content, such as announcements that refer to assignments, are captured and used in providing navigational help.
- Blind students are able to easily traverse the site providing keywords, using predefined keystrokes, as well as following provided guidance links.

The *iCare-Assistant* is built on top of an existing course server, *my.asu.edu*, available for the use of students at ASU. *my.asu.edu* is implemented using Blackboard software, and it hosts course home pages, each of which contains lectures, syllabus, assignments, projects, documents, announcements, external links (links to materials residing in different hosts or different locations in the course server), grades, a calendar, group pages, and a discussion board. The course pages are prepared by individual instructors in a voluntary-basis. Some of the content is fixed, meaning that it does not change within a semester (e.g. course syllabi), while others evolve (e.g. grades and discussion boards) through contributions by the instructors, TAs, and students. The *iCare-Assistant* provides a transparent interface between the Blackboard environment and the blind student, and adds context and keyword-based navigation (query) facility to locate the desired information with a few interactions only, reducing the navigational load in accessing information.

The first prototype of the *iCare-Assistant* has already been developed, and it is currently being used for test purposes. This prototype supports keyword-based navigation, along with some predefined key combinations (keystrokes) that may be used to access topic-specific information pages on course web sites, such as projects pages, announcement pages, etc. The context-based navigational support and dynamic restructuring of the information space for navigational purposes, as well as user profiling and task modeling will also be incorporated to the system in future. Overall system architecture of the *iCare-Assistant* in reference to its initial prototype is presented next in Section 2.

2. System Architecture

The *iCare-Assistant* consists of four components: 1) a web crawler that fetches pages and extracts links from them; 2) a parser that extracts content (information units) from pages; tokenizes, filters, and creates high quality keyword vectors that represent information nodes (information units); 3) a database manager (indexer) that stores information nodes generated by the parser, and maintains a global term and document frequency database; and 4) a retrieval engine that ranks information nodes based on the similarity between the nodes and a query. To navigate a course site, the student enters a query, and as a result is presented with the best content fragment (information unit) determined by the system based on the query. An information unit may be a course page or a component in a page (e.g. a specific lecture in a course documents page, or an announcement in the announcements page). We explain individual components of the *iCare-Assistant* in more detail in the following sections.

2.1 Crawler

The Crawler explores the *my.asu.edu* course server (MyASU web site) by following links from one page to another starting from the root page of a course. For each course, there is a separate crawler running, and each crawler terminates its execution after all pages on MyASU reachable from the course root page are visited. To keep the index information up-to-date, this crawling process is repeated regularly to reflect the changes on the course pages to the database.

MyASU is a secure web site, and it can only be accessed by students and instructors for the courses they are registered to or teaching, respectively. Hence, the crawler needs to authenticate itself as an authorized agent before it can fetch the content pages. The crawler speaks to the MyASU server using the HTTP/1.1 protocol, parses the header of HTTP request/response, and caches web authentication cookies in memory to continue communicating with the server. Once the crawler passes web authentication, it begins to fetch pages and extracts links from them. To avoid crawling the URLs that have already been visited, the crawler maintains a list of visited URLs. To avoid downloading executables, the crawler simply checks the URL file extensions. The crawler is also responsible for extracting annotation text which characterizes each link found on the page downloaded. This annotation information is used later by the parser to determine the type of the page. An annotation for a link is composed of the URL string and the anchor text of the URL for the page processed.

The crawler obtains the root URL for a course from the URL database. For each page explored, the crawler outputs an XML file that contains the URL of the page, its annotation information, as well as references (hyperlinks) to other pages from the page. This output is later used by the parser to determine the type of the page, and in turn, the structure of the page, in order to extract information units and create the corresponding nodes for indexing and retrieval purposes. Reference information is passed within XML files by the crawler so that the parser would not need to extract them again to store in the nodes.

2.2 Parser

The parser is a multi-threaded application, and each thread of execution is associated with a specific course page for which an XML file is created by a crawler agent. Based on the information stored in the XML file, the parser retrieves the corresponding content file and determines the type by comparing the annotation information (anchor text) with the keyword lists of types. Once the type is determined, a DOM tree is constructed (using the public-domain HTML Tidy Parser) and a configuration file for the type is read. This configuration file, specific to each type of page, guides the parser on how to navigate within the DOM tree as well as how to extract information units using the tree.

For example, in an *announcement* type of page, each announcement is an information unit which is rendered as a row in a table in HTML, having a bolded and emphasized title and a regular formatted message. Hence, the configuration file of an *announcement* page consists of two rules which specifies that: 1) the root of an information unit (announcement) is a table (given by a pair of <table> and </table> in HTML), and 2) each information unit (announcement) is a row that contains a title and a message, structured by the pattern <tr> <td>   title   message </td> </tr>. For each extracted information unit and the page from which the units have been extracted, an information node is created. These information nodes are indexed and stored in the database by the database manager, later.

For each node, a string tokenizer creates a list of keyword candidates. Keyword candidates are filtered using a list of stop words, such as “a”, “the”, “by”, etc., because those words do not have any useful content information. The quality of keywords is further enhanced by a publicly available stemmer package which removes inflexional endings from words in English. For example, “announce” and “announcement” are normalized as a new term “announc” by this stemmer. If a keyword being parsed is encountered for the first time, it is inserted into a words table as a new entry. Otherwise, the document frequency associated with that word in the words table is incremented by one to reflect the change on the number of documents that the word appears in the collection processed thus far.

2.3 Database Manager (Indexer)

The database manager is responsible for maintaining a persistent node table for the information units, a word table for the words extracted to be used as keywords for the information units, and an inverted keyword-to-node index table to determine in which information units a specific keyword appears and how many times it appears in each information unit. For each information unit, a new node is inserted into a node table. A node is identified by a node ID and classified into a specific type at the time of parsing. A page node is defined as a level-1 node while an information-unit node is defined as a level-2 node. Stored with each node is also the URL of the information unit and its relative position within the page (for an information unit that is a page itself, this value does not have any meaning). This information is utilized to display an information unit (corresponding to the best answer to a user query) at the top of the content frame on the user interface (user interaction module) of the *iCare-Assistant*., when the page containing the information unit is returned.

The inverted index is updated for all keywords in a node. The updates occur whenever a node is created. The term frequency associated with a word takes into account the position and importance of the keyword on the page. For example, the term frequency of a keyword is weighted up if the keyword appears in the title and headers, or if it is formatted in bold, italics, or emphasized.

2.4 Retrieval Engine

The retrieval engine computes the importance (similarity) values for the information units, given a user query expressed by a list of keywords, comparing the keyword list stored in the nodes for the information units to the keyword list of the query. The *iCare-Assistant* returns the most relevant content fragment (information unit) to a

user query and also provides the user with a short list of alternatives ranked in descending order of importance. The user has an option to go through the list of alternatives if she is not satisfied with the best answer displayed. The number of alternatives for a query result may be adjusted for each user based on her preferences kept in the user profiles. The list of alternatives is kept short because providing too many alternatives, as the search engines do, is not desirable; it increases the navigational overload, which the *iCare-Assistant* attempts to minimize.

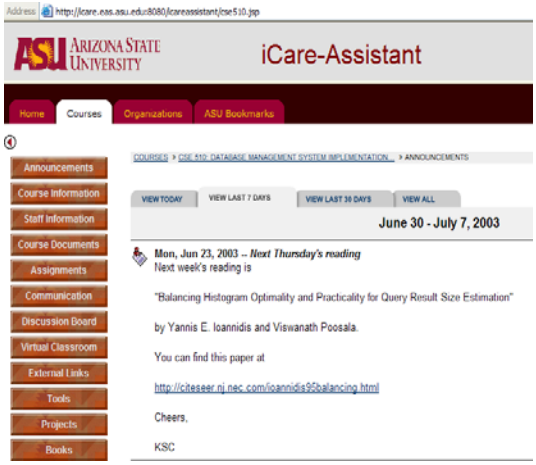

Figure 1: iCare-Assistant User Interaction Module

3. User Interaction Module

The *iCare-Assistant* user interaction module is built on top of the Blackboard environment and communicates with the *my.asu.edu* course server. It is implemented as an HTML Application (HTA) to overcome the security restrictions enforced by the Internet Explorer, and it uses JavaScript to interact with the user and to control the display of the query result. It listens to keystroke events; brings up and presents the most relevant content to the user if a query is given; and allows the user to freely browse course pages on the Blackboard environment. The overall look of the *iCare-Assistant* user interaction module is shown in Figure 1, where the MyASU web site is framed below the *iCare-Assistant* title bar.

The user can navigate through course pages as she would do using a web browser, after logging onto the MyASU web site. The user interaction module of the *iCare-Assistant* is designed to be fully accessible through keyboard and JAWS for blind students. It is transparent to the user in that it feels as if the blind user were accessing the Blackboard system through a web browser only, but with the query functionality added to locate the most relevant course content with a few interactions, which the Blackboard system lacks.

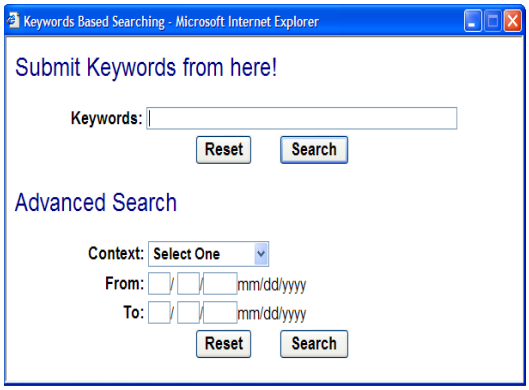

Figure 2: iCare-Assistant Query Interface

The user launches the query dialog by keystrokes (ALT+s) and inputs her keywords for a search. Figure 2 gives a snapshot of the query dialog. After the query is submitted, the page that contains the most relevant information to the user's query is found, loaded, and displayed on the content frame with the information unit determined to be the best answer at the top. JAWS speaks the returned content to the blind user starting from the top of the content frame, where the query answer (information unit) is located, rather than reading from the beginning of the loaded page (default action in JAWS) which would be done only if the page is the best answer to the query as an information unit itself. The navigational load is minimized by bringing up the most relevant content to blind students from the hyperlinked course information space with only a few interactions by the user. The blind user does not need to have a specific training to learn how to use the *iCare-Assistant*, except for remembering a few key combinations to launch the query dialog and for the predefined topic-specific retrieval tasks. Hot keystrokes defined in the initial prototype of the *iCare-Assistant* are as following:

- **Alt+s (**Pops up a Search Dialog)
- **Alt+i** (for the instructor information)
- **Alt+l** (for syllabus)
- **Alt+c** (for the Course Documents page)
- **Alt+p** (for Projects)

4. Conclusions and Future Work

We have developed the first prototype of the *iCare-Assistant*, which is currently being used for test purposes. It supports keyword-based navigation and some predefined key combinations

(keystrokes) for accessing topic-specific information pages on course web sites, such as projects pages, announcement pages, etc. The context-based navigational support, dynamic restructuring of the information space for navigational purposes, user profiling, and task modeling will be implemented in future.

The *iCare-Assistant* will provide context- and task-dependent, personalized navigational guidance to blind students. First of all, in order to prevent disorientation, the blind student will be able to traverse the site providing keywords and using predefined keystrokes. Since *iCare-Assistant* integrates the educational material that it receives from various sources as they become available, most annotations and navigational pathways will be automatically created using context-dependent and user-dependent profiles, as well as the semantic content of the lecture material as in Figure 3. To keep the number of available options (which can be large, due to the densely packed nature of the hypermedia material) from becoming cumbersome and bothersome, we will develop and use a fixed and limited interaction language that (1) is easy learn and use, and (2) always presents predictable and consistent navigational results. Dynamic adaptation of the material available at the *my.asu.edu* course server will be the underlying technique that we will employ to help blind students to access electronic course material. We have already explored dynamic adaptation of Web-information structures [5, 6], mining document associations [7-10], and summarization of Web sites [8, 10] for sighted people. We will build on our existing work by developing dynamic adaptation, mining, and summarization techniques suitable for blind users. Such dynamic adaptation requires the *iCare-Assistant* to adjust the information space available at the course server based on the current context provided by the user. When the context changes, some information becomes much more important than the rest, and thereby, should be more accessible. Based on the current content, context, access history, user preferences and information, and the hyperlink structure, the *iCare-Assistant* will determine the information needed and will present it to the student in the appropriate format. Logical relationships between various content (such as announcements that refer to particular assignments), will be captured and used in providing navigational help.

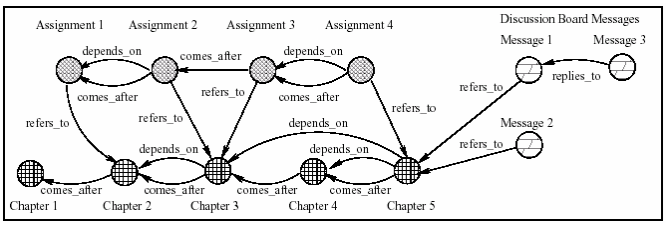

Figure 3: An example graph representation for a course

References

1. H. Lieberman, "Letizia: An agent that assists Web browsing", 14th Int. Joint Conf. on Artificial Intelligence, Montreal, Canada, 1995.

2. J. Kleinberg, "Authoritative sources in a hyperlinked environment", Journal of the ACM, vol. 46, pp. 604-632, 1999.

3. S. Brin and L. Page, "The anatomy of a large scale hypertextual web search engine", 7th Int. WWW Conf., 1995.

4. I. Herman, G. Melancon, and M. S. Marshall, "Graph Visualization and Navigation in Information Visualisation: a Survey", IEEE TVCG, 6(1), pp. 24-43, 2000.

5. W.-S. Li, K. S. Candan, Q. Vu, and D. Agrawal, "Retrieving and organizing web pages by information unit", *WWW 2001*, pp. 230-244.

6. W.-S. Li., K. S. Candan, Q. Vu, and D. Agrawal, "Query relaxation by structure and semantics for retrieval of logical web documents", IEEE TKDE, 2001.

7. K. S. Candan and W.-S. Li, "Using random walks for mining web document associations", Pacific Asian Conf. on Knowledge Discovery and Data Mining, 2000**.**

8. K. S. Candan and W.-S. Li, "Discovering web document associations for web site summarization", WWW Posters, 2001**.**

9. K. S. Candan and W.-S. Li, "Discovering web document associations for web site summarization", *DaWaK 2001*, pp. 152-161.

10. K. S. Candan and W.-S. Li, "Reasoning for web document associations and its applications in site map construction", Int. Journal of Data and Knowledge Engineering, 2002.