Accordion Summarization for End-Game Browsing on PDAs and Cellular Phones

Orkut Buyukkokten, Hector Garcia-Molina, Andreas Paepcke
Digital Libraries Lab (InfoLab), Stanford University, Stanford, CA, 94305
{orkut, hector, paepcke}@cs.stanford.edu

ABSTRACT

We demonstrate a new browsing technique for devices with small displays such as PDAs or cellular phones. We concentrate on end-game browsing, where the user is close to or on the target page. We make browsing more efficient and easier by Accordion Summarization. In this technique the Web page is first represented as a short summary. The user can then drill down to discover relevant parts of the page. If desired, keywords can be highlighted and exposed automatically. We discuss our techniques, architecture, interface facilities, and the result of user evaluations. We measured a 57% improvement in browsing speed and 75% reduction in input effort.

Keywords

PDA (Personal Digital Assistant), WWW (World-Wide Web), HTML, WML, WAP

INTRODUCTION

PDAs provide convenient and portable access to a wealth of information. But the very portability of PDAs poses a formidable design challenge: How do we present information effectively on small devices such as a Palm Pilot, or a cellular phone? The problem is compounded because most of the information today is organized for large displays that are connected to high bandwidth links. Previous studies indicate that information retrieval tasks on the Web are much harder to complete on devices with small displays [15]. When we try to render a Web page on a small display, viewing becomes a huge burden because of the excessive scrolling effort. If we render the page as a regular desktop browser, four-way scrolling can be used. However, four-way scrolling is very annoying since the user needs to scroll the display for each line s/he reads. On the other hand, two-way scrolling requires clipping and wrapping the text, resulting in increased Web page length.

One solution is to prepare special information feeds that serve information in smaller chunks, anticipating small display areas. The wireless Palm VII model uses this approach with its "web clippings" [21]. These information pages are marked up in a subset of HTML. The pages are compiled into a compact format that can be decoded on the Palm VII. Another example for the special information feed approach is the Wireless Access Protocol (WAP), which includes its own markup language to replace HTML on small displays. The special preparation of information to suit small displays can, of course, yield excellent results. The drawback of this approach is that vast amounts of HTML information remain inaccessible.

Another approach is to miniaturize standard Web pages. This is done very well, for example, by Puma Technology's ProxiWeb [24] (formerly known as Top Gun Wingman). It converts all the pictures into small versions that can be shown on a Pilot [11]. This approach is especially useful when a user has found the main page s/he is looking for, and wishes to see all the details. The miniaturization approach is less advantageous during the navigation phase of an information task, and when the final page is even moderately large. During navigation, the user needs to process a large amount of information, and must scroll often, just to find the next link to follow. Larger pages, even when they contain the correct information, can be disorienting on small displays, especially when both vertical and horizontal scrolling is required.

Our Power Browser, part of Stanford's Digital Library Project, enables access to the World-Wide Web from wireless PDAs in ways that circumvent these difficulties. The Power Browser explicitly supports the two aspects of information access mentioned in the previous paragraph: navigation and *end-game* browsing. During navigation a user has not reached the final browsing destination and mainly follows links. During end-game browsing the user is slowing down, examining each page in more detail.

In our previous work [3] we addressed the navigation phase of information access. In this mode, the Power Browser only shows the links that are contained in each page. The user may use a left-to-right pen gesture over a link to see the links on the referenced page. This new set of links is listed indented on the display, just like the file browser on Windows or Macintosh computers can expand folders in their file browsers. The Power Browser also provides site-specific search and word-completion facilities [4]. As the user begins to enter search keywords with the pen, possible keyword completions are offered. The keywords on this completion list are all guaranteed to occur within the site being accessed. In [3,4] we showed that these techniques substantially reduce the time and number of pen motions required in the navigation phase.

In this paper we turn to end-game browsing, where the user wishes to examine pages in detail. We summarize the information on these pages in various ways. We still do not show all of the textual information at once, unless requested to do so. Instead, we use syntax, and sometimes user directed mechanisms for conveying high-level information, while making the details accessible at a stroke of the pen. We call our summarization strategy *accordion summarization* because a page can be shrunk or expanded much like an accordion.

METHOD

We use three main techniques in accordion summarization:

- Page Summarization
- Keyword-Driven Summarization
- Automated View Transitions

Page Summarization

To illustrate, Figure 1 shows the image of an actual Web page. For purposes of explanation in this paper, we have marked the image with numbers and rectangles. Figures 2 and 3 show how we summarize this example page on a PDA and on a cell phone, respectively. We do not discuss the controls in the top toolbar, but one of them lets us switch between the shown end-game view and our previously-developed navigation view.

Notice that the hand-drawn rectangles identify coherent blocks of text on the page. We call these blocks semantic textual units (STUs). Later in this section we will explain how the Power Browser identifies these units.

The Power Browser summarizes each STU into a single line (that may be expanded) and then presents the summary lines as a hierarchical tree (that can also be shrunk and expanded). For example, STU-7 is summarized by its first 5 words in line 7 of Figure 2 and by its first 3 words in line 7 of Figure 3. (The line numbers in Figures 2 and 3 are just to illustrate; they do not appear on the display.)

Each summary line has a line marker icon (small black/white circle) to the left, telling the user whether the corresponding STU is longer than the summary line. Figure 4 illustrates how users interact with line markers to progressively expand STU-6. The marker can be in three states: one line displayed (black circle), three lines displayed (half black circle), or all lines displayed (white circle). By repeatedly tapping the line marker, the user can rotate among the states. If the STU fits in a single line then there is a single state (white circle), and if it fits in three or fewer lines there are only two states (black and white circles). The scheme lets us use screen real estate wisely, displaying larger STUs only when the user is interested. If the text contains hyperlinks, they are underlined in the display and can be followed by tapping.

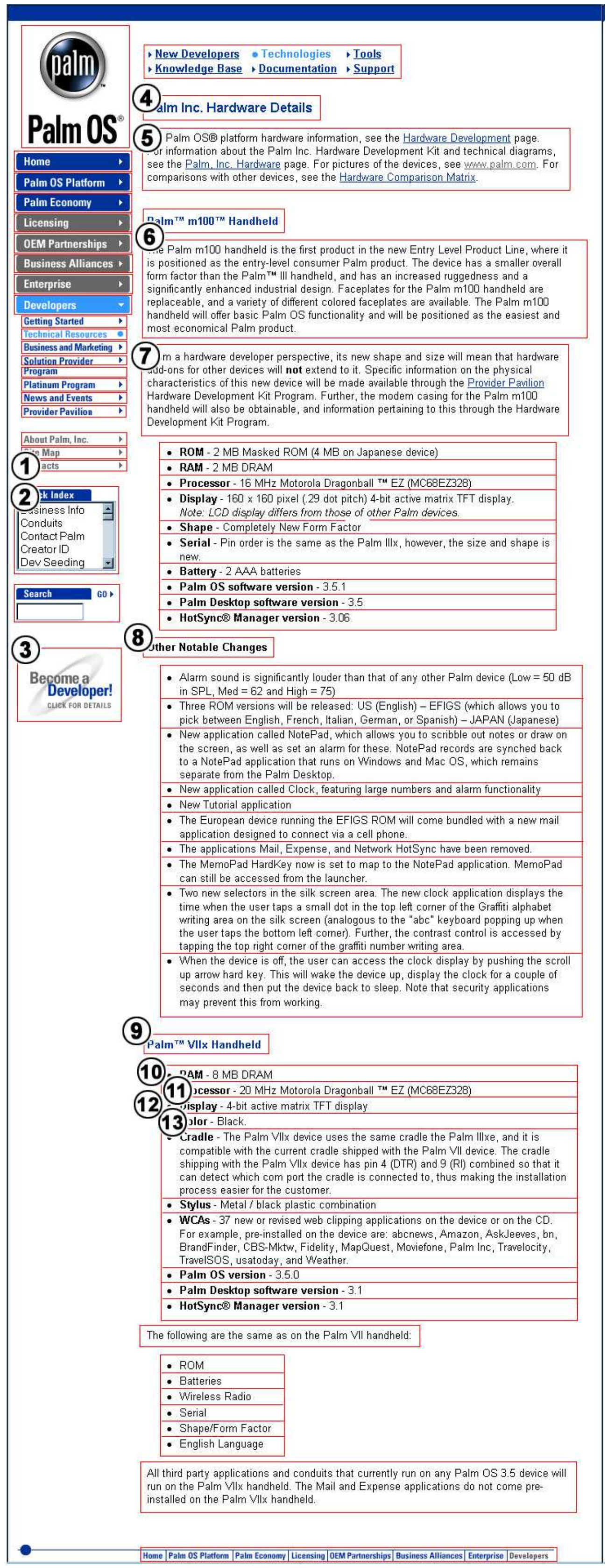


Figure 1: STUs on the PalmOS Web page

Figure 2: Page summary on the Palm Pilot

Figure 3: Page summary on the Nokia

In our current implementation, a semantic unit containing text is summarized by its first line. However, we could also use text summarization or fact extraction algorithms [7, 14] and display their output.

For non-text units, we use a variety of heuristics. For instance, in STU-3 the ALT property of the image ("Become a Developer") is used as the summary. On the other hand, STU-1 contains an image with the word "Contacts," but there is no ALT property. Fortunately the URL for the image ends with "/contact/". Thus we use the string "Contact" as the summary of the image (Line 1 of Figure 2).

STU-2 illustrates an STU that is a select menu containing link choices. In this case we represent the menu as the concatenation of the menu items, and we display as many as fit in one line (Line 2 of Figure 2). This summary line can be expanded in the usual form.

Since screens only have a few lines (e.g., 13 on the PDA, 8 on the cellular phone), the user can only see a limited number of STU summaries. To make it easier to browse through the STUs, we organize the STUs hierarchically into a tree structure. Every tree node is an STU summary, and nodes can be nested within others. The user can progressively expand and collapse the tree nodes to find the relevant STUs. The user can also use the scroll bar control on the right to move the tree structure up and down.

STU-7, for example, consists of an introductory paragraph, followed by a bulleted list. STU summaries reflect such nesting. Thus, the STUs of the list items are hidden under the STU-7 summary.

The *nesting controls* ('+' and '-' characters) allow users to expand and collapse nested STUs. Figure 2 shows the state after the STU-9 summary has been expanded. Hence STUs

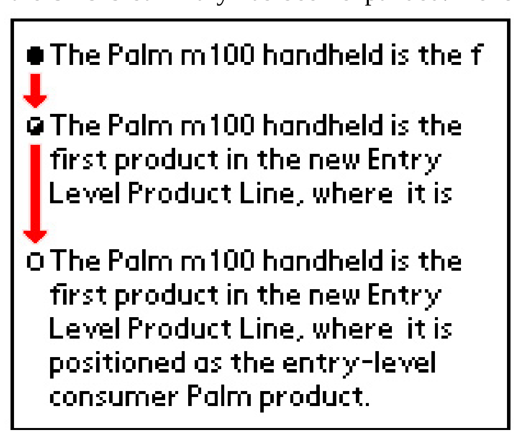


Figure 4: The line markers

(The actual display shows all of STU-6 for the bottom state)

10 through 13, the list items that follow STU-9, are displayed, and nested under STU-9. (Notice that indentation is from the left-most icon, not the first text character.)

The proper partitioning of Web pages into STUs and their organization into a hierarchy are, of course, central to effective accordion browsing. This partitioning and organization is a heuristic process.

Partitioning: The Power Browser uses syntactic features to split a Web page into STUs. For example STU-6 and STU-7 are separated by a new paragraph token. STU-7 is separated from the STUs that follow by a list token. Similarly, table cells and frames are considered separate STUs. Line breaks within a table cell do no indicate a new STU.

Organization: Syntactic properties also help us to determine how STUs are related, and must therefore be nested. The basic rule is that a decrease in emphasis (e.g., smaller font size) increases the depth in the tree, while an increase in emphasis (e.g., bold font, larger font) decreases the level. For example STU-5 is considered a child of STU-4 because the font size is decreased at the partition point.

The beginning of a list is considered a decrease in emphasis. For example, suppose that STU-*X*, at level *i*, is followed by the first element of a list, STU-*L1*. Then STU-*L1* is at level *i+1*. All the list items that follow will also be at level *i+1*, and STU-*Y* that follows the last list item reverts to depth *i*. A change of emphasis within a list item can change the depth (e.g., a nested list, different font). However, no STU between STU-*X* and STU-*Y* can have a depth smaller than *i+1*. In Figure 1, for example a list follows STU-9, and hence the list items are nested under it. Frames and tables are handled in an analogous fashion.

We realized that supporting Cascading Style Sheets (CSS) is crucial for organizing STUs, since emphasis directives are sometimes not in the HTML page itself. For example, the style sheet can change the font size of anchors or header tags (e.g., H1, H2). The Power Browser processes style sheets to determine all style changes in order to derive a correct hierarchical representation of the page.

After the partitioning and organization steps we introduce a final cleanup step to simplify the hierarchy when possible. For example, If STU-*Y* begins with a repetition of an immediately preceding STU-*X*, then STU-*X* is removed. See, for example, STU-6 in Figure 1, where the header is removed.

We tried several different approaches for displaying the tree structure. Initially, we represented different structural features with separate icons (e.g., for frames, tables, lists). When we tested the system, we realized that all the different icons put too much of a cognitive burden on the user while operating the tree structure. We decided instead simply to use plus or minus signs like a conventional tree widget. We noticed that this representation made the tree look more user friendly and much easier to operate. In order to minimize the space taken by the node icons, we first used three pixels for the icons on our PDA implementation. However, the users had a hard time tapping on the small icons on the tree widget. As a result, we decided to increase the icon size to five pixels.

Keyword-Driven Summarization

When the browsing process is goal oriented, the user has a good idea of where to locate information. For instance, if the user is interested in a phone number, STUs that contain the keywords "contact" or "phone" will be valuable. We use this concept to improve browsing speed. If the user enters keywords (by tapping on the magnifying glass icon on the toolbar in Figure 2), the summarization is modified to take these keywords into account. All occurrences of the keywords in STUs are highlighted and revealed to the user immediately. All the nodes on the tree structure containing keywords are expanded automatically. Instead of displaying the first line of such STUs, the line where the keyword appears is displayed and highlighted. If the user taps on the line marker, the line before or after the highlighted keyword is revealed. If the user taps again, the entire STU is shown. Figure 5 illustrates how a summary is generated when the keyword "ryman" is given.

Using keywords for information retrieval and specialized indices is a well-known technique (e.g., Keyword In Context [18]). However, in our *Keyword-Driven Summarization*, the keywords entered by the user are used to form a series of extractions each containing at least one of the keywords. The keywords are active across browsing boundaries and are used to distinguish parts of the Web page summary.

Automated View Transitions

On a small screen, the user can easily get disoriented when the contents of the page change abruptly. In order to reduce this cognitive load, each content change is made through a smooth transition so that the user can easily adapt to the new screen. This is done in three ways:

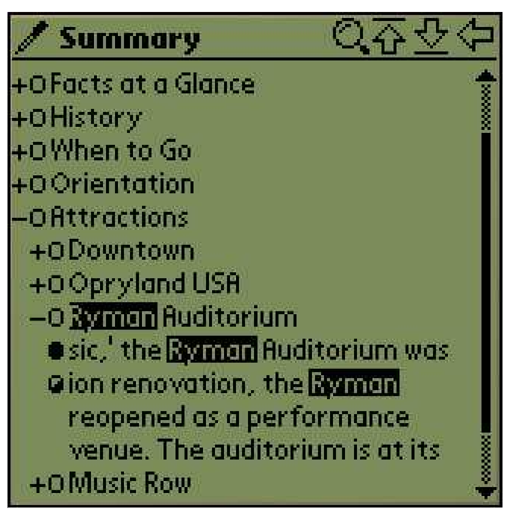


Figure 5: Keyword-Driven Summarization

- *Smooth scrolling:* When the user manually scrolls the screen, instead of changing the display abruptly, the summary is scrolled down or up. The smooth transition helps the user keep track of the summary page with less effort.
- *Automated page scrolling:* Often times, when the contents of the display change, the user would need to scroll up or down to adjust to the new screen. We avoid this inconvenience by scrolling the display up or down automatically. For instance when a tree node is expanded, the node is scrolled up to make as many children visible as possible. Similarly when a node is collapsed and there is empty space left, the nodes are scrolled down to use up the space. Smooth scrolling is used here as well, in order to maintain visual continuity.
- *Automated single-line reading:* The user also has the option of reading longer STUs using a single line, without disturbing the remainder of the display. For this, the user makes a left to right gesture on the line s/he wants to read. This action causes the line to periodically replace its content in-place with the next line in the STU. The user can thus read the STU one line at a time without issuing any further commands. We observed that on the average 90 milliseconds per line for the PDA was sufficient for the average reader. The time interval between each line can be changed on the PDA on demand. The user can stop the automatic scrolling by tapping anywhere on the display.

We experimented with several transition effects to mark the progression from one line to the next. For example, in separate experiments, we scrolled consecutive lines left, up or down. We also tried briefly highlighting each new line as it appeared. When we used these methods, users felt that highlighting was too distracting and scrolling left took too much time. Users preferred for each line to disappear by scrolling up and for the new line to appear without transition.

IMPLEMENTATION

The processor on a small device such as the Palm Pilot has the power of desktop machines in the mid 80's. Thus, in the Power Browser we use proxy technologies to improve performance in two ways. First, the proxy performs computation-intensive operations on behalf of the browser. Second, the proxy decreases the response time over the wireless connection by filtering irrelevant content (e.g., multimedia, tags, comments) and by using compression.

Figure 6 shows the architecture of the system. The arrows represent the data flow between the components. After a client initiates a browsing request to the server, the Page Parser fetches the requested Web page. (If the user entered keywords, these are transmitted with the request.) The parser module decomposes the HTML page into its tokens. If there is an associated style sheet, it is downloaded and parsed as well. The Partition Manager receives the tokens (i.e., tags and text elements), analyzes them and generates an STU Table. Each entry in this table represents one STU and its associated properties, such as the text characteristics (e.g., text size, font type) and structural information (e.g., table, list, frame). The Organization Manager extends the STU Table to include the depth of each STU in the corresponding tree structure.

The Representation Manager transforms the STU Table into the Display Table, using the appropriate Device Profile. The profile contains device characteristics such as display resolution, font type and text size. Each entry in the display table represents one line of text that may be displayed on the device. That is, each STU is broken into the multiple lines that may be necessary for displaying it. The final decision as to what lines to actually display is made at the browser based on user input. If the user specified keywords, the display table includes extra lines with highlighted keywords that the browser may display.

Finally, the display table is sent to the client to drive the display. Note that user actions on that page such as expanding the tree structure do not generate more traffic to the proxy. The browser manages those locally. The proxy communicates with the client via HTTP, and a variety of devices can share the same proxy. Each client simply identifies its hardware when it makes its initial request, and the proxy generates the device-dependent response.

In order to experiment with accordion summarization, we first implemented a device emulator for the desktop. The PalmOS emulator [20] was not generic enough since we wanted to test our system on separate devices with different characteristics. Our emulator can mimic the operation of the Power Browser on different devices such as the Nokia cellular phone (Figure 3) and the Palm Pilot (Figure 2). A

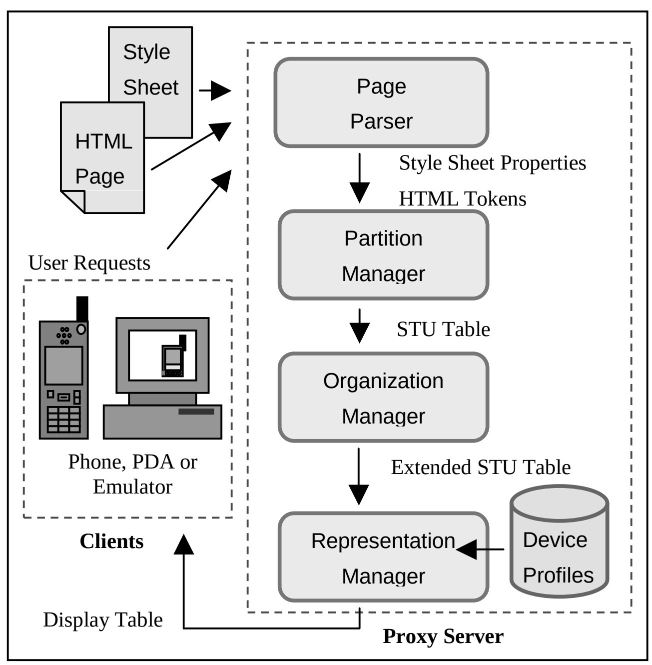


Figure 6: The architecture

new device can be added with minimum effort by modifying the device profiles database. An emulator cannot, however, replace a device, no matter how good it is. Therefore, we also implemented the system on an actual device, the 3Com Palm Pilot.

Even though we have not yet implemented the client on a cellular phone, we have completed an initial design. The main challenge is the different input modality: Input is entered through push buttons instead of a pen. In order to accommodate that difference, we plan to add line numbers to the display. Pressing a digit N will cause line N to expand or collapse. Pressing *N will be equivalent to tapping on the line marker for line N. Finally, #N will select and follow an anchor on line N. We also plan to experiment with a time-sensitive solution. Here the functionality is selected by the amount of time the key is pressed.

PERFORMANCE STUDIES

User Performance

In our user evaluation, we used a Palm III with a wireless Ricochet modem instead of the emulator, to ensure realistic results. (We also have a Palm VII implementation.) The subjects were 15 individuals aged 12 to 45. They all knew how to use a Web browser. Most of them never used a PDA before, and a few of them had previously used ProxiWeb.

Each user performed fifteen tasks (See Table 1). Since we are evaluating end-game browsing, each task involved searching through a single Web page. The pages differed in length; the shortest one could be fully viewed with a desktop browser on a single 1024x768 screen, while the largest required 15 screens.

We evaluated three different browsing methods:

- *ProxiWeb:* ProxiWeb is one of the popular commercially available browsers on the Palm platform. It also uses a proxy server. As mentioned earlier, ProxiWeb tries to render a Web page as faithfully as possible, like a desktop browser.
- *Power Browser:* Our browser using accordion summarization but without keyword support.
- *Keyword-Driven Power Browser:* Our browser using accordion summarization and keyword support. Here, we pre-selected the keywords to use. We did not let the users choose the keywords so that the results would not depend on the particular choice. The keywords we provided are listed in the right-most column of Table 1. We also pre-entered the keywords on behalf of the users to eliminate the differences in graffiti proficiency. (Graffiti is the Palm Pilot's character set for pen input.)

Of the fifteen tasks each user performed, sets of five were performed with each method. Each user therefore performed all the tasks, and used all the methods. We varied the order in which users were exposed to the methods. We video-taped the interaction of each user with the device. Later we analyzed the tapes to measure the completion time and input effort.

Graph 1 summarizes the average task completion time for each method. For all the tasks, the Power Browser, with or without keyword support, performed better than ProxiWeb. For most tasks the Keyword-Driven Power Browser was more effective than the Power Browser, but surprisingly this was not the case for Tasks 3 and 14. In Task 3 the keyword was a single character and the highlighted character was hard to see. In Task 14, the keyword occurred multiple times and users tried expanding the summary without checking the context around the keyword. Keep in mind that the time to enter the keywords must be added to the times shown in Graph 1. Thus, depending on user proficiency, keyword support may or may not be advantageous.

Similarly, Graph 2 shows the average input effort for each

Task	***Keywords***
1) On the Matrox Graphics Product Specification page find out how much memory is supported for the Millenium G400 graphics card.	memory
2) From Jian Liu's personal home page find out what kind of sports she likes.	sports
3) On the Museum of Flight page find the museum admission price.	$
4) On the UCSF directory Web page find the link to the directory of the UCSF emergency telephone system.	emergency
5) On Prof. Julie A. Jacko's home page, find where she received her degrees.	degrees
6) From Arturo Crespo's home page find out his office room number.	office
7) On the Encyclopedia Britannica Web page for Franz Schubert find the names of his operas in his early life and career.	opera
8) On the National Wildlife Federation's Grizzly Bear page find information about 'primary threats' to the bears.	threat
9) On Senator Joseph Lieberman's home page find the senator's phone number.	phone
10) From the XML.com page find out the date of the "XML DevCon 2000" conference.	devcon
11) Find the attractions at Ryman Auditorium from the Lonely Planet – Nashville Web page.	ryman
12) On the Bureau of Census home page find the link to "News on minorities".	minority
13) Find the duration of internships at the United States Holocaust Museum.	duration
14) On the Chi 2001 home page find the link that gives information about the location of the conference.	location
15) On the Palm Inc. Hardware details page find the processor for the PalmVII.	processor

Table 1: Tasks for user experiment

task. We measured input effort by the number of pen taps on the display. For all but Task 6, the Power Browser, with or without keyword support, required less input effort. Pressing on a scrollbar button also counted as a pen tap. On ProxiWeb, all the pen taps are for scrolling. On the Power Browser, only 35% were for scrolling and the remainder expanded or collapsed the tree structure. On the Keyword-Driven Power Browser, 50% of the pen taps were for scrolling.

When we compare the Power Browser to Proxi-Web, we observe a 57% improvement in browsing speed and a 75% reduction in input effort. For the Keyword-Driven Power Browser, we get a 69% improvement in browsing speed compared to ProxiWeb. Finally, the Keyword-Driven Power Browser yields a 31% improvement in task completion time and 29% reduction in input effort relative to the Power Browser (again, not taking into account the time to enter keywords).

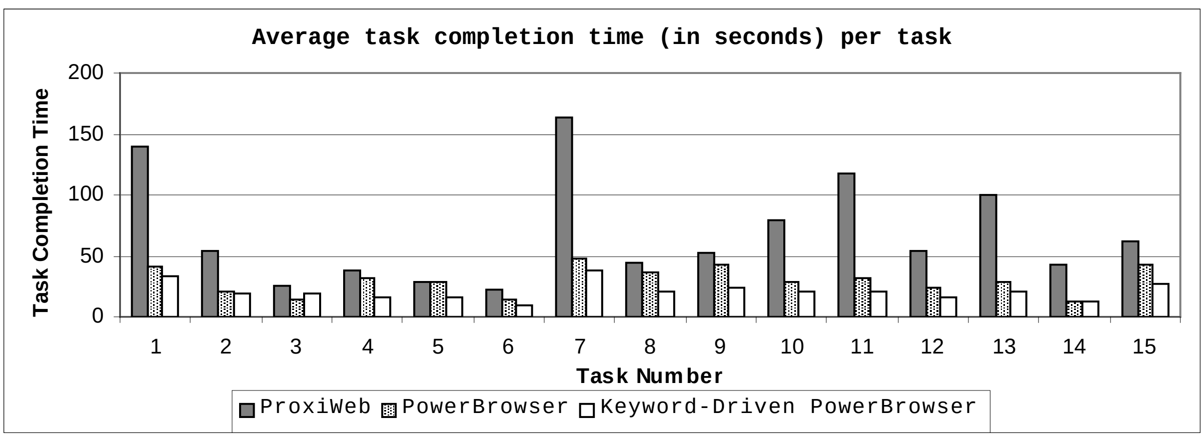
In summary, our results show a marked end-game improvement with the Power Browser. Incidentally, we believe that accordion summarization could also play a role during the navigation phase. For example, note that in Tasks 4, 12 and 14, accordion summarization also helped us find links quite effectively. We plan to conduct experiments in the future comparing the Power Browser's navigation support with its accordion summarization facility.

System Performance

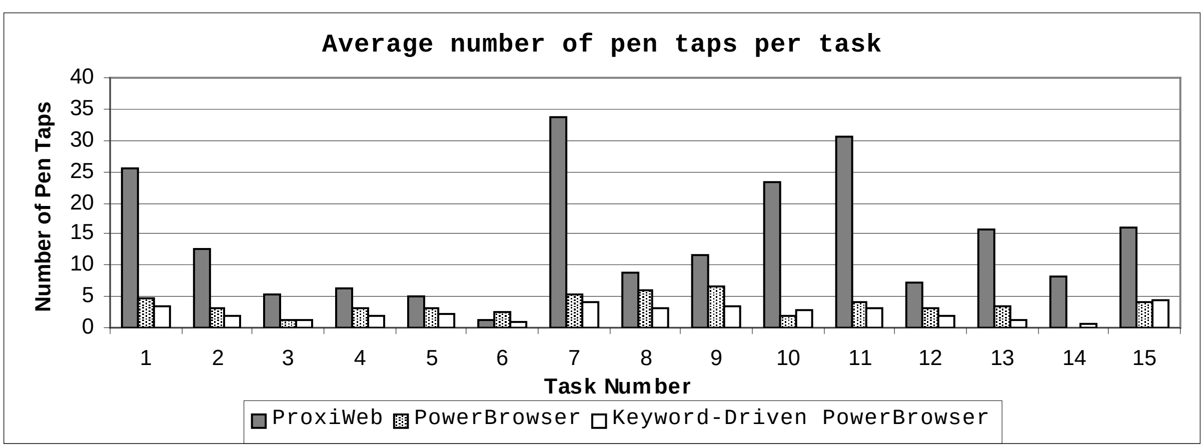
The client requires 90 kilobytes for code and 30 kilobytes for memory. The size of a display table sent by the proxy server is significantly smaller than the actual Web page. For the Web pages of our experiments, on average pages shrink 82% (for every 100 bytes in the page, the transmitted display table contains only 18 bytes). If we consider only the text in the page (i.e., the HTML and CSS), than the shrinkage is 54%.

Accordion summarization also significantly reduces the amount of information that the user must see. For example, to see the full text of each of the fifteen task pages of our experiments requires between 3 and 88 screens (18.4 screens on the average) on the Palm Pilot. For the Nokia between 6 and 324 screens (66.1 screens on the average) are required. However, with accordion summarization each initial summary (no nodes expanded) only takes between 1 to 4 screens on the Palm Pilot (between 1 to 6 on the Nokia). Thus the user can very quickly get a high-level summary of the Web page, and then drill down into the details without too much scrolling and reading effort.

Our approach adds the most value for large pages. Well organized content yields the best accordion views.



Graph 1: Task completion time



Graph 2: Input effort

RELATED WORK

The strategy of accordion summarization is in the tradition of Fisheye Views [12] where the user sees places nearby in great detail while the larger context is displayed in successively less detail. Hypertext systems have explored tree structures for multiple documents [9] and large tables of contents [6]. Structured browsing systems have been used widely and been shown to be effective for documents [8] and search results [5, 23]. WebToc [19] uses tree navigation for browsing; however, they browse over the whole site, not individual pages. Expanding and contracting text in place is sometimes called "holophrasting". The term and practice is introduced in [13]. Search word support has been incorporated in summary browsing [8], but Superbook's table of content summary was not adjusted when keywords occurred in the text body.

Keyword highlighting is a common technique for visual search tasks [2, 22]. Many techniques have been developed to make interaction easier for small displays, such as overlapping widgets through transparency [17] and graceful image degradation [10].

Digestor [1] provides device-independent access to Web pages. It also uses structural page transformations and sentence elision. However, Digestor links deleted text, instead of hiding it as in accordion summarization. Page partitioning occurs in [16], which converts Web pages to decks and cards for WAP devices.

CONCLUSION

In this paper we proposed accordion summarization, a new approach for end-game browsing of Web pages on small devices. We showed how this technique significantly reduces the browsing effort. This improvement is accomplished by providing syntactic page summarization and a progressive browsing mechanism that allows the user to gradually view the contents of a Web page.

REFERENCES

1. Bickmore, T.W. & Schilit, B. N. Digestor: Device-independent Access to the World-Wide Web. *Proc. WWW6*, 1997.
2. Brown, T.J. Visual Display Highlighting and Information Extraction. *Proc. The Human Factors Society*, 35th Annual Meeting, 1991, 1427-1431.
3. Buyukkokten, O., Garcia-Molina, H., Paepcke, A., & Winograd, T. Power Browser: Efficient Web Browsing for PDAs. *Proc. CHI'2000*, 430-437.
4. Buyukkokten, O., Garcia-Molina, H. & Paepcke, A. Focused Web Searching with PDAs. *Proc. WWW9*, 2000, 213-230.
5. Chen, M., Hearst, M., Hong, J., & Lin, J. Cha-Cha: A System for Organizing Intranet Search Results. *Proc. the 2nd USENIX Symposium on Internet Technologies and SYSTEMS (USITS)*, 1999.
6. Chimera, R., Wolman, K., Mark, S. & Shneiderman, B. An Exploratory Evaluation of Three Interfaces for Browsing Large Hierarchical Tables of Contents. *ACM Transactions on Information Systems*, 12, 4, Oct. 94, 383-406.
7. Edmundson, H. P. New Methods in Automatic Extracting. *Journal of the Association for Computing Machinery* 16(2), 1968, 264-285.
8. Egan, D., E., Remde, J. R., Landauer, T. K., Lochbaum, C. C. & Gomez, L. M. Behavioral Evaluation and Analysis of a Hypertext Browser. *Proc. CHI'89*, 205-210.
9. Feiner, S. Seeing the Forest for the Trees: Hierarchical Display of Hypertext Structure. *Conference on Office Information Systems*, New York: ACM, 1988. 205-212.
10. Fox, A. & Brewer, E. A. Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation. *Proc. WWW5*, 1996.
11. Fox, A., Goldberg, I., Gribble, S. D., Lee, D. C., Polito, A. & Brewer, E. A. Experience With Top Gun Wingman: A Proxy-Based Graphical Web Browser for the 3Com PalmPilot. *Conference Reports of Middleware*, 1998.
12. Furnas, G. W. Generalized Fisheye Views. *Proc. CHI'86*, 16-23.
13. Hans, W. J. User Engineering Principles for Interactive Systems. *Proc. of Fall Joint Computer Conference*, Nov. 1971, AFIPS Press, 523-532.
14. Hovy, E., & Lin., C.-Y. Automated Text Summarization in SUMMARIST. *Proc. Workshop on Intelligent Scalable Text Summarization*, Aug. 1997, 18-24.
15. Jones, M., Marsden, G., Mohd-Nasir, N., Boone K. & Buchanan, G. Improving Web Interaction on Small Displays. *Proc. WWW8*, 1999, 51-59.
16. Kaasinen, E., Aaltonen, M., Kolari, J., Melakoski, S. & Laakko, T. Two Approaches to Bringing Internet Services to WAP devices. *Proc. WWW9*, 2000, 231-246.
17. Kamba, T., Elson, S. A., Harpold, T., Stamper, T. & Sukaviriya, P. N. Using Small Screen Space more Efficiently. *Proc. CHI'96*.
18. Luhn, H.P. Keyword-In-Context Index for Technical Literature (KWIC Index). *American Documentation* XI, 4, 1960, 288-295.
19. Nation, D. A., Plaisant, C., Marchionini, G. & Komlodi, A. Visualizing Web Sites using a Hierarchical Table of Contents Browser: WebToc. *Proc. the 3rd Conf. on Human Factors and the Web*, 1997.
20. Palm, Inc., Palm OS Emulator: http://www.palmos.com/dev/tech/tools/emulator/.
21. Palm, Inc., Web Clipping Development: http://www.palmos.com/dev/tech/webclipping/.
22. Philipsen, G. Effects of Six Different Highlighting Modes on Visual Search Performance in Menu Options. *Intl. Journal of Human-Computer Interaction* 6, 3, 1994, 319-334.
23. Pratt, W., Hearst, M. A., & Fagan, L. M. A Knowledge-Based Approach to Organizing Retrieved Documents. AAAI-99: *Proc. of the 16th National Conf. on AI*, 1999.
24. ProxiNet, Inc., ProxiWeb: http://www.proxinet.com/.