# Structure-Aware Web Transcoding for Mobile Devices

Transcoding heuristics that consider a Web page's structure and the relative importance of Web components make complex Web pages accessible from handheld devices.

**Yonghyun Hwang, Jihong Kim, and Eunkyong Seo**
*Seoul National University*

As both the number of mobile users and users' reliance on the Web grows, so does the need for Web access from handheld devices.[1] The current disparity between such devices' available computing resources and the resources required for smooth Web browsing makes it difficult and unpleasant to access Web pages with them. To navigate complex Web pages with a handheld device, a user must scroll down and across the page many times. Furthermore, although handhelds are becoming more powerful, few can satisfactorily handle multimedia data.

Two general solutions to this disparity exist: *manual* and *automatic* reauthoring. In manual reauthoring, Web authors prepare multiple versions of a Web page targeted to resource profiles of various platforms, including the Wireless Application Platform (www.wapforum.org).[2,3] Although this approach can produce high-quality pages for specific devices, it assumes a Web author will both be available to reauthor the pages and will know what pages users will want to access. Because no one can predict how Web surfing will progress (during a mobile Web search session, for example), this approach severely limits the number of Web pages accessible via handheld devices.

In automatic reauthoring,[4,5] a transcoding module transparently converts individual pages, making all Web pages accessible. Although attractive in theory, this approach is not yet widely used because the transcoding quality is poor. Systems such as Pixo (www.pixo.com/products/products002.htm) generate almost unusable pages from complex Web pages with large nested table structures. Existing automatic reauthoring techniques ignore the relative importance of Web page *components* — basic units for transcoding — which can be extracted through syntactic analysis of the page's HTML source code. Furthermore, because most existing transcoding techniques are local transforms, they do not consider the

 Published by the IEEE Computer Society 1089-7801/03/$17.00©2003 IEEE 

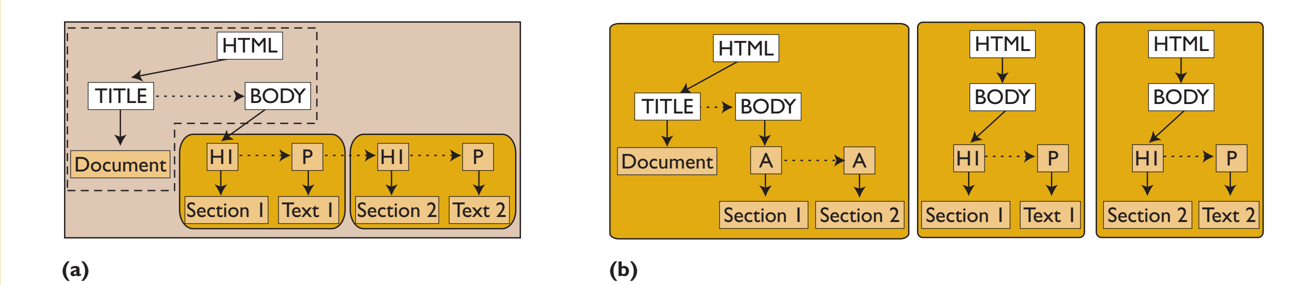


*Figure 1. Tree-based representation of a Web page. (a) The outlining transform's grouping function divides Web components into subtrees; (b) the summarizing function chooses subtrees to be elided and modifies the tree to reflect these decisions.*

Web page's overall structure when making decisions. (For a description of current techniques, see the sidebar, "Existing Transcoding Heuristics.") Recent work by Jinlin and colleagues[6] considers a Web page's overall layout during the transcoding process, but does not reflect the relative importance of Web components.

Our long-term goal is to develop a high-quality syntax-based Web transcoding system that allows universal access to Web pages without manual reauthoring. We focus on *structure-aware* transcoding heuristics, which preserve the original Web page's underlying layout as much as possible. The proposed heuristics extract the relative importance of Web components from an intelligent syntactic analysis. Like other transcoding approaches, ours does not support executable content such as Java, Javascript, and Flash. To evaluate our heuristics' effectiveness, we performed subjective quality evaluations in Web-based experiments.

## Transcoding Framework

We first describe a transcoding framework, which is useful in understanding the strengths and weaknesses of transcoding heuristics in a unified setting.

### Web Page Representation

We represent each Web page using a modified tree structure. This lets us efficiently analyze pages because it represents various page layouts (for example, nested table structures) and expresses the interrelationship among components. Figure 1a shows a tree-based internal representation using a sample Web page.

The tree-based data structure has two node types and two edge types. *Context nodes* (white boxes in Figure 1a) contain attributes of the corresponding HTML structure, such as the HTML tag, the estimated screen real estate the HTML structure requires, and other heuristic-specific data. *Terminal nodes* (shaded boxes in Figure 1a) include Web content to be displayed, such as text blocks and images. The tree's solid edges point to the (possibly nested) substructures, and the dashed edges represent the sibling relationships between connected nodes.

### Problem Formulation

A Web transcoding technique $H$ generally consists of two functions: a *grouping function* and a *summarizing function.*

The grouping function $H_g$ divides a Web page into several subgroups, forming Web components. Given a tree representation $T_w$ of a Web page $W$, $H_g$ partitions $T_w$ into a set of subtrees.

The summarizing function $H_s$ includes three subtasks:

- deciding which subgroups will be reduced to hyperlinks in the transcoded pages;
- choosing representative phrases for the elided subgroups; and
- modifying the tree-based representation to reflect decisions made in the first two steps.

Consider, for example, the outlining transform $O$,[4] which separates section headers from subsequent text, as Figure 1 shows. The grouping function $O_g$ identifies sections based on section headers in the Web page (Figure 1a). The summarizing function $O_s$ leaves the section headers as hyperlinks, hiding the text blocks behind them (Figure 1b).

Given a tree representation $T_w$ of a Web page $W$ and a set of $n$ transcoding heuristics $\{H_1, ..., H_n\}$, the Web transcoding problem aims to find a sequence $S = H_{i_n} \circ H_{i_{n-1}} \circ ... \circ H_{i_3} \circ H_i$ of transcoding heuristics such that the transformed pages most efficiently satisfy a given quality metric. (For existing transcoding heuristics, see the sidebar on p. 18.)

## Structure-Aware Transcoding Heuristics

To improve the presentation of transcoded Web pages on a mobile device, transcoding heuristics

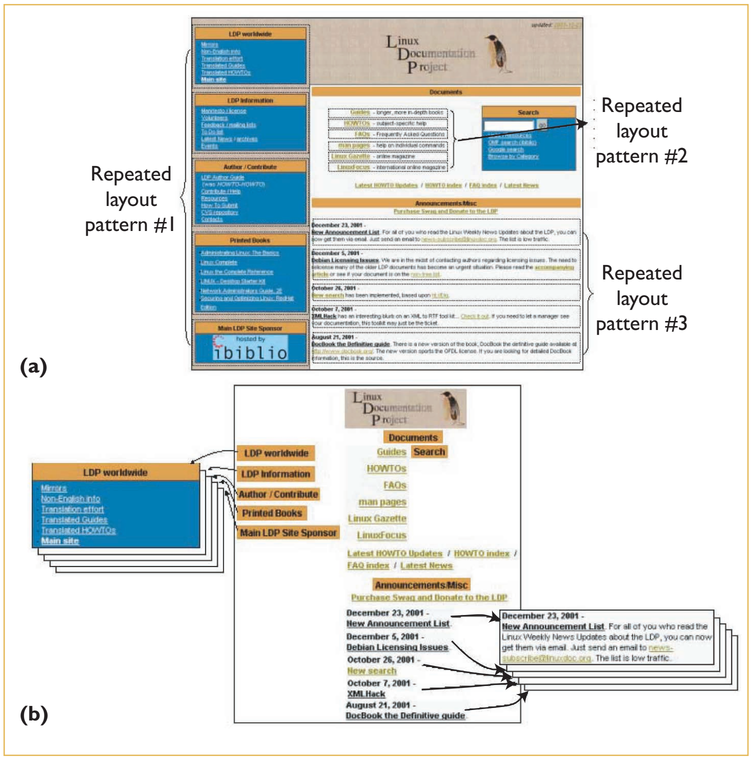


*Figure 2. An example of the generalized outlining transform. (a) The transform identifies three repeated layout patterns in a Web page and (b) transcodes the page into a sequence of smaller pages.*

should preserve the layout of the original Web pages as much as possible. We introduce two new heuristics, the *generalized outlining transform* and the *selective elision transform*, which preserve the structure of Web pages during transcoding. Both exploit common layout characteristics of complex Web pages. They use more general grouping functions than other transcoding heuristics while considering the page's overall layout. (For a more detailed description, see our earlier work.[7])

## Generalized Outlining Transform

Complex Web pages generally use multiple repeated layout patterns. This is a characteristic that other art mediums share. For example, most graphic artists know that large posters should not use more than four colors or patterns; to deliver the message more effectively they should use a few patterns repeatedly.

Figure 2a shows three repeated layout patterns in a sample Web page. If we can identify these patterns, we can group them like the outlining transform groups header and text components. To detect repeated layout patterns, we need a general grouping condition. For example, we cannot use the outlining transform to group the repeated patterns in the sample page because the patterns do not use section headers. The generalized outlining transform, however, can identify these patterns using a general grouping function that transcodes the Web page to a sequence of smaller pages, as Figure 2b shows.

**Grouping function.** The generalized outlining transform's grouping function depends on repeated layout patterns, rather than specific tag combinations. Although repeated layout patterns are easy to identify visually, they are difficult to detect in a tree-based representation. Successive Web components often have the same layout but different tree representations because they contain significantly different content. Consider, for example, the five leftmost table cells in Figure 2a. As Figure 2b shows, the cells form a repeated layout pattern although the bottom cell in Figure 2a contains an image while the top four cells contain hyperlinks only.

To find all the repeated layout patterns in a Web page $W$ and assign them to groups, the grouping function traverses $W$'s tree-based representation $T_W$ in postorder. Whenever it meets a context node representing a structural tag (such as `<ul>` or `<table>`), it executes the prefix pattern-matching algorithm to find a repeated layout pattern from a subtree rooted with the current context node, or *structural context node.*

To find a repeated layout pattern for each subtree $T'$ rooted with a structural context node, we solve the prefix pattern-matching problem on the string representation of subtree $T'$. Given the tree-based representation $T'$, we convert $T'$ to a string representation $S_{T'}$, consisting of symbols representing various HTML tags and text blocks in the Web page. For example, we would convert the source code `<a>WebAlchemist</a><br> is a structure-aware transcoding system` to the string, $T_A$ *Text* $E_{tag}$ $T_{BR}$ *Text*. $T_A$ corresponds to `<a>`, $T_{BR}$ represents `<br>`, $E_{tag}$ represents all closing tags such as `</a>`, and *Text* indicates a text block.

We search for frequently occurring substrings of $S_{T'}$, which are candidates for a repeated layout pattern. Because we match prefixes only, the different content types do not affect matching results. The suffixes in a string representation denote the Web component's content type.

To find a repeated layout pattern, we construct a string tree for a given string representation. Each string tree edge represents a string element, which symbolizes an HTML tag or text block. Each node records the number of times the corresponding substring occurs in the given string representation. We construct the corresponding substring by following the edges from the root to the node.

Figure 3 illustrates how we construct a string tree, using repeated layout pattern #2 in Figure 2 as an example. When we insert the string $T_A$ *Text* $E_{tag}$ *Text* $T_{BR}$ into a string tree, the string follows an edge $T_A$, if it exists, under the root. If not, we create a new node and set its node counter to 1. This node is linked with the root, and we associate the edge between the root and the new node with $T_A$. We similarly insert *Text*, followed by $E_{tag}$, *Text*, and $T_{BR}$. As we insert each character of this string, the corresponding node counter increments by 1. The string tree in Figure 3 shows that the string $T_A$ *Text* $E_{tag}$ *Text* $T_{BR}$ repeats six times.

After building the string tree, we can easily find repeated substrings through tree traversal. We collect all substrings appearing more frequently than a given threshold as candidate repeat substrings. We then select the final repeated substring using two rules:

- *Rule 1*. Select substrings starting with a structural tag or a hyperlink.
- *Rule 2*. Select the longest of the selected substrings.

Finding a repeated substring takes $O(n^2)$ operations, where $n$ is the number of nodes in a subtree rooted by a given node.

We compute the repeated layout pattern from the selected repeated substring. Because we only match the repeated substring's prefix, the selected substring defines the beginning portion of a repeated layout in a Web component. The grouping function combines all the components located between the beginning of the current pattern and the beginning of the next.

**Summarizing function.** After the grouping function identifies the repeated layout patterns, the summarizing function decides which repeated layout patterns to elide. It converts the matched prefixes of the selected layout patterns into hyperlinks, hiding the rest of the content behind them. For example, it replaces repeated layout pattern #1 in Figure 2a with five hyperlinks using the matched prefixes (category names such as "LDP worldwide") in Figure 2b.

Because complex Web pages typically have several repeated layout patterns, the summarizing function must select the appropriate patterns from among many candidates. We introduce two new parameters, *shrinking factor* and *information density,* to assist the selection process. The shrinking factor is the ratio of a Web component's display size after it is transcoded to its size before it is transcoded. The summarizing function generally favors repeated layout patterns with high shrinking factors.

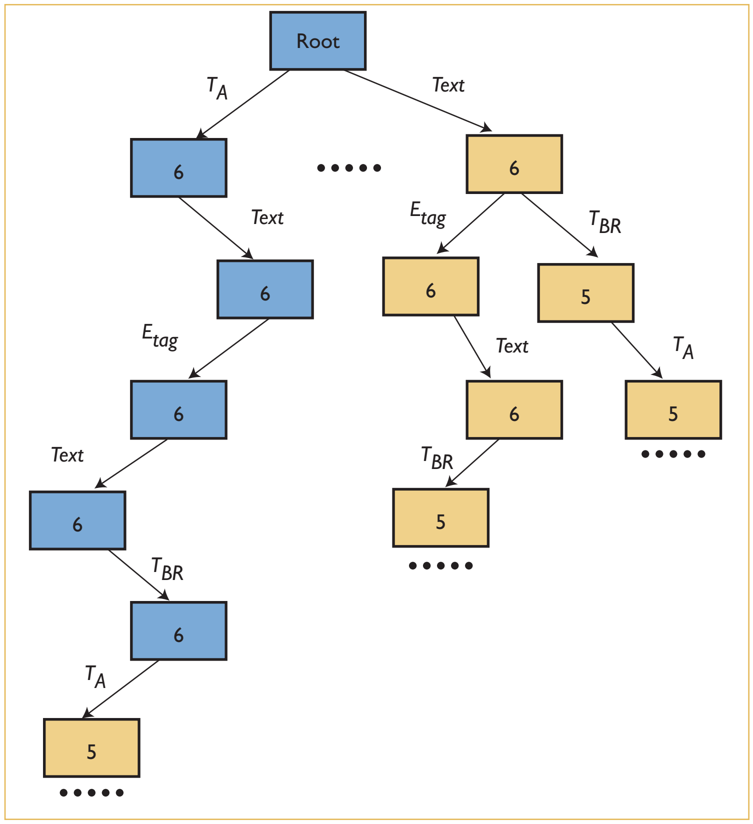


*Figure 3. A simple example of string tree construction. Each edge represents a string element; inside each node is the number of times the substring is repeated.*

Information density refers to the amount of content accumulated in a Web component, indirectly representing the component's importance. The summarizing function does not select repeated layout patterns with high information density because eliding such semantically important portions will degrade the transcoding quality.

Each Web component starts with an information density of 1. In the transform's current version, a Web component's importance is determined by syntactic attributes such as font size, table cell width, cascading style sheets (CSS), and the number of already elided subcomponents. The information density increases by 1 if the font size or the table cell width is larger than a predefined threshold value. (This is based on the assumption that HTML authors commonly use larger fonts or wider table cells to emphasize more important components.) If a Web component includes previously elided subcomponents, its information density also increases by 1, which prevents the same component from

## Existing Transcoding Heuristics

The existing literature describes seven transcoding heuristics.[1,2]

- Bickmore and colleagues proposed the *outlining transform*[1] for paragraphs that begin with section headers. The outlining transform replaces the section headers with hyperlinks pointing to the corresponding text blocks. This transform effectively preserves a Web page's itemized structure while significantly reducing the display size.
- The *improved outlining transform*[2] adds a more general grouping function and supports more syntactic combinations than the original outlining transform. Transcoding system developers can use the improved outlining transform where conceptually higher (more abstract) and lower (more detailed) pairs exist.
- Transcoding system developers use the *first sentence elision transform*[1] when a Web page's text blocks are too large to be displayed on a handheld device. A hyperlink hides all but the first sentence of the text block. This transform works well when the first sentence summarizes the entire block.
- The *restricted first sentence elision transform*[2] has a more limited grouping function than the first sentence elision transform. If a long text block is within a table structure, or a text block includes a table structure, the restricted transform suppresses the first sentence elision transform to maintain the Web page's table structure. By not applying the first sentence elision transform where table structures exist, this transform lets table-specific heuristics handle tables.
- The *image reduction* and *elision transforms*[1] are useful in dealing with images in Web pages. They scale down images with a predefined scaling factor and create hyperlinks pointing to the reduced images.
- The *indexed segmentation transform*[1] divides a long Web page into a sequence of small subpages that fit a handheld device's display. The transform tries to find logical elements, such as text blocks or lists, by analyzing syntactic information on the Web page. It sequentially arranges the identified elements in the transcoded page until it can properly display the new page on the handheld device. The transform then creates a sequence of subpages, each connected via hyperlinks.
- The *table transform*[1] identifies a table in a Web page and checks that a handheld device can display it properly. If the table is too wide or too long, the transform unrolls it and creates one subpage per cell in a top-down, left-right order.

Existing transcoding heuristics employ grouping functions that are effective only for small combinations of HTML tags. Furthermore, because the grouping functions consider only the local syntactic attributes without weighing a Web page's overall layout, they cannot properly reflect the developer's intention, especially for complex Web pages. Our structure-aware transcoding heuristics improve on these by considering both the importance of Web components and the Web page structure when making transcoding decisions.

### References


1. T. Bickmore, A. Girgensohn, and J.W. Sullivan, "Web Page Filtering and Reauthoring for Mobile Users," *Computer J.*, vol. 42, no. 6, 1999, pp. 534–546.
2. Y. Hwang et al., "WebAlchemist: A Web Transcoding System for Mobile Web Access in Handheld Devices," *Proc. ITCom*, SPIE — The Int'l Soc. for Optical Eng., 2001, pp. 37–46.


being elided repeatedly, which makes it difficult to preserve the Web page's original layout. The Web components in the lower right corner of Figure 2a have an information density of 3 because they use boldface text in a wide table cell.

After assigning a shrinking factor and an information density to each Web component, the summarizing function classifies the components into five categories based on these numbers.[7] For example, if a component has a high shrinking factor but low information density, the summarizing function elides it. On the other hand, if the component has a high information density, the summarizing function does not elide it because it is important.

### Selective Elision Transform

Many popular Web sites use complex tables to organize their content structurally. Properly transcoding such complex Web pages for handheld devices is a challenge. Existing transcoding heuristics such as the table transform (see the sidebar) often destroy the original table structures. Once the table structure is broken, it is difficult to understand the author's intent.

The selective elision transform attempts to preserve table structure as much as possible: in addition to table attributes such as cell width, this transform uses syntactic attributes such as font size to decide whether to elide a table cell.

Figure 4 shows an example of the selective elision transform. The transform's grouping function selects all table cells whose width is larger than a given threshold (Figure 4a). When table cells are nested, it examines the inner cells first.

As with the generalized outlining transform, the selective elision transform's summarizing function uses shrinking factor and information density to decide which subgroups to elide. It cal-

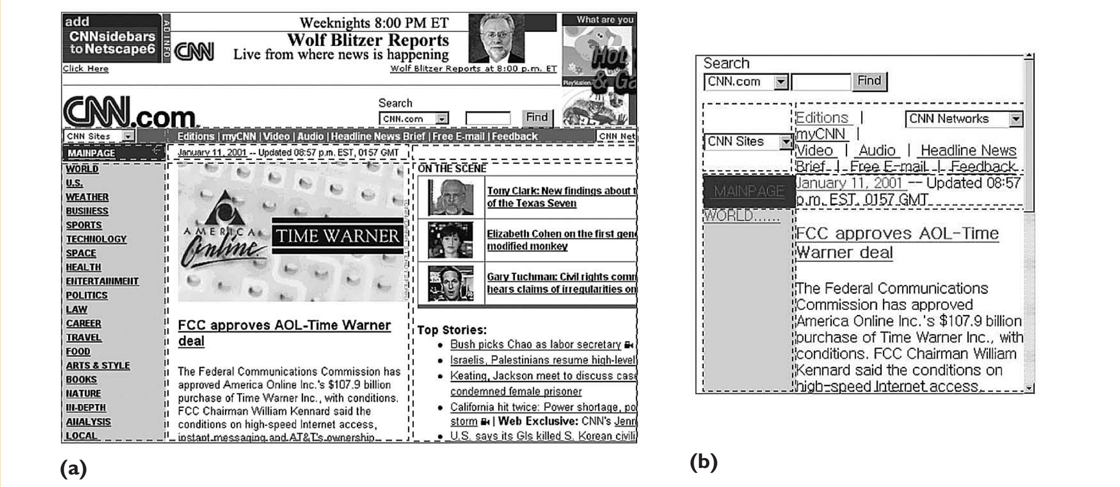


*Figure 4. An example of the selective elision transform. (a) The CNN homepage and (b) the transcoded CNN homepage after applying the selective elision transform.*

culates these parameters similarly to the generalized outlining transform.

The center cell in Figure 4a has higher information density because it is wider than the others, and is therefore not elided. As Figure 4b shows, the selective elision transform preserves the important table structure of the original CNN homepage.

## Experiments

To evaluate the effectiveness of the proposed transcoding heuristics, we extended the prototype of the WebAchemist transcoding system, an in-house testbed for Web transcoding research.[8] We implemented our new heuristics and modified WebAlchemist's transcoding manager module to reflect the new transcoding parameters, information density and shrinking factor.

### Transcoding Manager

The extended WebAlchemist system supports six transcoding heuristics: the image reduction and elision transforms, the restricted first sentence elision transform, the indexed segmentation transform, the improved outlining transform, the generalized outlining transform, and the selective elision transform. We also modified the first four transcoding heuristics to work with information density and the shrinking factor.

The transcoding manager's main role is to decide how to use the six heuristics. Because transcoding heuristics require several parameters for proper operation (the display size of a handheld device and predefined threshold values, for example), the transcoding manager also decides which value to use for each parameter. We empirically determined the following sequence of steps as a default sequence for WebAlchemist:

1. improved outlining transform
2. generalized outlining transform
3. selective elision transform
4. restricted first sentence elision transform
5. image reduction and elision transforms
6. indexed segmentation transform

This sequence follows our intuition that preserving the overall layout (or structure) is an important requirement for high-quality transcoding of complex Web pages. The first three transforms effectively reduce display size while keeping the original page structure.

### Subjective Evaluation

To evaluate WebAlchemist's effectiveness in converting complex Web pages, we chose a subjective evaluation. Initial experiments showed that similar objective characteristics can exhibit striking differences when evaluated subjectively.

Forty-three college students and engineers living in Seoul participated in the evaluation. We asked each participant, all of whom are active Web users, to judge the transcoded pages' quality using one of five grades: *fair, good,* and *excellent* are satisfactory ratings while *unusable* and *poor* are unsatisfactory. We used 13 Web sites as test pages,[7] four of which were well-known news sites (for example, www.cnn.com) and four were popular portals/search engines (such as www.yahoo.com). A simulated $320 \times 240$ display (typical for PDAs) showed both transcoded and original pages.

Figure 5 summarizes the results of our subjective evaluation. Figure 6 shows a transcoded ver-

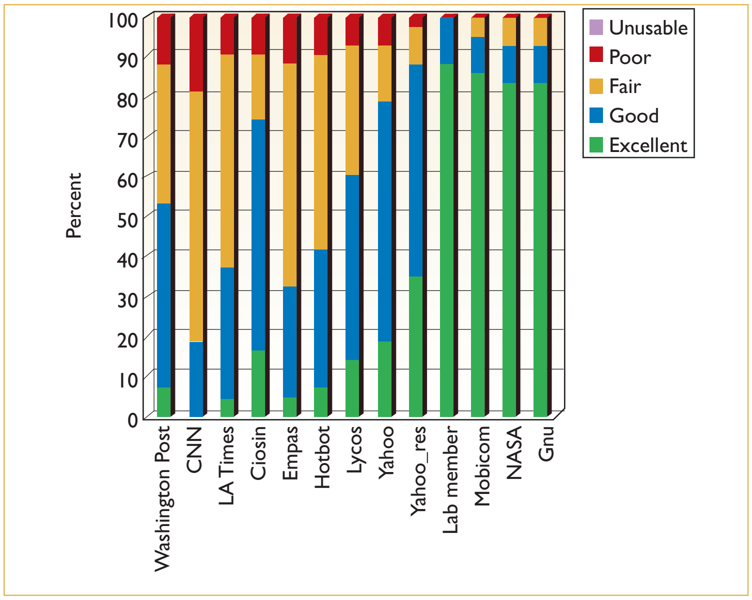


*Figure 5. Subjective evaluation results. At least 80 percent of the participants rated the test pages as satisfactory or better.*

sion of the Yahoo homepage. (All transcoded Web pages are available at http://davinci.snu.ac.kr/WebAlchemist/experiments/.)

As Figure 5 shows, at least 80 percent of the evaluators gave the test pages satisfactory ratings. Four of the pages — Lab_member, Mobicom, Nasa.gov, and Gnu.org — received *excellent* ratings from more than 80 percent of the evaluators. Furthermore, no evaluator rated any test Web page *unusable*.

One interesting observation from this evaluation is that most mobile users avoid horizontal scrolls as much as possible, a fact that contributed to the CNN homepage's relatively poor rating. Considering the complexity of the tested Web pages, the evaluation results demonstrate WebAlchemist's overall effectiveness.

## Conclusion

Although the current version of WebAlchemist already produces usable Web pages for handheld devices, we can further improve it within the automatic reauthoring framework. For example, knowing a Web component's access profile can help determine its relative importance. We plan to extend WebAlchemist by incorporating user access profile into the existing transcoding heuristics.


### Acknowledgments

A Korea Research Foundation grant (KRF-2001-041-E00243) supported this work.


### References


1. K.H. Britton et al., "Transcoding: Extending E-Business to New Environments," *IBM Systems J.*, vol. 40, no. 1, 2001, pp. 153-178.
2. E. Kaasinen et al., "Two Approaches to Bringing Internet Services to WAP Devices," *Proc. 9th WWW Conf.*, ACM Press, 2000, pp. 231-246.
3. J. Freire, B. Kumar, and D.F. Lieuwen, "Webviews: Accessing Personalized Web Content and Services," *Proc. 10th WWW Conf.*, ACM Press, 2001, pp. 576-586.
4. T. Bickmore, A. Girgensohn, and J.W. Sullivan, "Web Page Filtering and Reauthoring for Mobile Users," *Computer J.*,


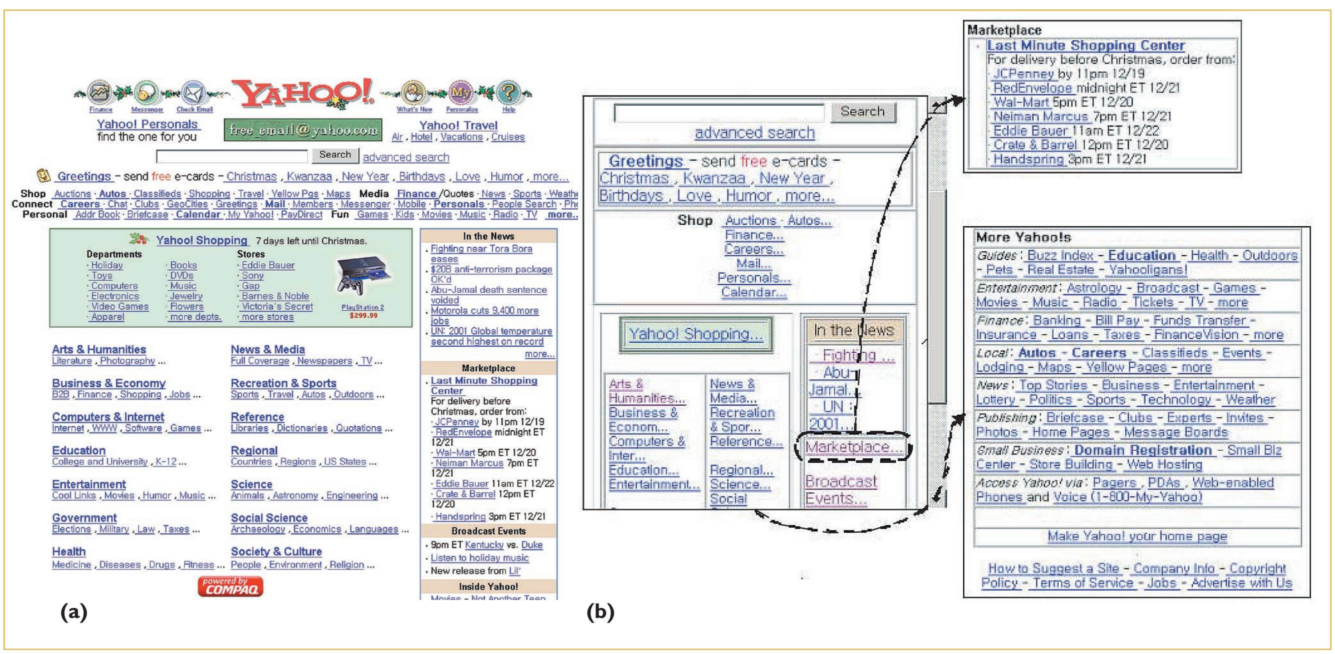


*Figure 6. Transcoding example. (a) Original Yahoo.com homepage and (b) WebAlchemist-transcoded version of the page.*

vol. 42, no. 6, 1999, pp. 534-546.

5. T. Bickmore and W. Schilit, "Digestor: Device-Independent Access to the World Wide Web," *Computer Networks and ISDN Systems*, vol. 29, no. 8, 1997, pp. 1075-1082.
6. J. Chen et al., "Function-Based Object Model Towards Website Adaptation," *Proc. 10th WWW Conf.*, ACM Press, 2001, pp. 587-596.
7. Y. Hwang, J. Kim, and E. Seo, *Structure-Based Transcoding Heuristics for Mobile Devices*, tech. report, School of Computer Science and Eng., Seoul National Univ., 2002; http://davinci.snu.ac.kr/Download/hwang_techrep.pdf.
8. Y. Hwang et al., "WebAlchemist: A Web Transcoding System for Mobile Web Access in Handheld Devices," *Proc. ITCom*, SPIE — The Int'l Soc. for Optical Eng., 2001, pp. 37-46.

**Yonghyun Hwang** is a full-time instructor at the Naval Academy, Korea. His research interests include Web transcoding for mobile systems, mobile communications, and embedded systems. Hwang received an MS in computer science and engineering from Seoul National University, Korea. Contact him at semiaion@hotmail.com.

**Jihong Kim** is an associate professor in the School of Computer Science and Engineering, Seoul National University, Korea. His research interests include embedded systems, computer architecture, real-time systems, and Java computing. Kim received a PhD in computer science and engineering from the University of Washington. For questions and comments about this article, contact him at School of Computer Science and Engineering, San 56-1 Shilim-Dong, Kwanak-Ku, Seoul National University, Seoul, Korea 151-742; jihong@davinci.snu.ac.kr.

**Eunkyong Seo** is a senior student in the School of Computer Science and Engineering, Seoul National University, Korea. Her research interests include optimizations, algorithm design and analysis, and embedded systems. Contact her at ilekseo@hotmail.com.