User-Centric Adaptation of Structured Web Documents for Small Devices

Wai Yip Lum & Francis C.M. Lau
Department of Computer Science
The University of Hong Kong, Hong Kong
{wylum,fcmlau}@cs.hku.hk

Abstract

Content adaptation is a crucial step in making desktop-oriented web resources available to mobile, small device users. In this paper, we propose a decision engine comprising a content analysis module and a negotiation module to serve as the core of a content adaptation architecture. The content analysis module parses a structured web document originally intended for the desktop into small sections and transforms the document into a form that is best suited for rendering in a constrained mobile device. The transformation also provides the user with the best content value in an adapted web page while preserving content integrity. With the transformed document, the negotiation module selects the best rendering parameters to be used in the synthesis of an optimal adapted version of the content. The decisions made are based on the user's preference and QoS considerations. We have built a prototype to demonstrate the viability of our approach.

1 Introduction

The rapid development in mobile computing technology has made available a tremendous amount of information in the World Wide Web to mobile devices. An efficient and effective content access mechanism is most critical for the mobile device users who tend to work under many constraints. Despite constant advances in the hardware technology, mobile devices continue to be limited in their capabilities, and because of that, contents need to undergo adaptation in order to satisfy the rendering requirement of the mobile device users. The growing variety of device types would only add to the acuteness of this need. Although the computational power of recent mobile devices is always on the increase, their screen size continues to be limited as these devices need to be lightweight and fit in the pocket.This poses a challenge to any adaptation strategy aiming to fit contents intended for larger displays in small screens. This paper is on a feasible approach to meeting this challenge. A good solution to this problem could benefit millions of mobile users.

2 Related Work

To achieve good adaptation, the adaptation process needs to be fully aware of the content structure. Content understanding can be facilitated by providing "wrappers" around web sources [3]. A wrapper for a web source accepts queries about information in the available pages, fetches relevant pages from that source, extracts the requested information and returns the results in a structured form. To enable these actions, the content in a page is parsed into manageable sections, and special tokens are used to identify the beginning of a section, and other meaningful components. Similarly in [1], syntactic features like table cells, line breaks, etc. were used to split a web page into STUs (semantic textual units). Then organizational information (how the STUs are nested) can be extracted by such simple rules as the upper level units should appear in more emphatic styles, e.g., bold font.

Since there are plenty of ways by which the author would mark a section, simple token matching might not be sufficient to discover the hierarchical structure of the content. In this paper, we propose another approach to discovering the logical sections in the content, which is "relationship-aware" and has the intelligence to deal with more complicated content structures that are common in modern web sources.

The Function-based Object Model (FOM) [4] aims at capturing the author's intention by identifying the relevant objects in a document. Every object in a homepage serves a purpose, e.g., decoration, hyperlinking, or interaction with the user. Based on object analysis, an understanding of the content's structure is derived. Nevertheless, such a content adaptation is limited to only awareness of the content itself, but not the target user's preference which is another important factor in trying to achieve a higher degree of user satisfaction. In our design, we provide a level of adaptation which is based on the relationships of the elements in the

1550-445X/05 $20.00 © 2005 **IEEE**

web document rather than just the individual objects' functionalities. In addition, the design is user-centric—*i.e.*, it takes into consideration the user's preference.

Text summarization was proposed in [2] where a web page is broken into text units, each of which can be hidden, partially displayed, fully visible, or summarized. Algorithms were introduced to search for the appropriate keyword or key sentence for the text summarization. Nevertheless, the study focused on the textual elements only.

3 Relationship Discovery

In a document, some content elements are related. The special treatment to be given to these elements because of their relationships is as important as their semantic information because the original document semantics may not be presented properly if such a treatment is missing or the relationships in question are not preserved over the adaptation.

There are generally three types of elements making up a document:

- Semantic element: which presents semantic information. For example, the Anchor element and the Image element, both of which contribute semantic information to the web document.
- Container element: which encloses other elements, but does not itself present any semantic information. For example, the Table element, which acts as a container of other elements.
- Layout element: which provides information to dictate how other elements are to be presented. For example, the Bold element and the Font element.

After identifying all the elements and their types, a web document can then be mapped to its corresponding Content Relationship Tree (CRT). A CRT is composed of nodes and links. A node is either a semantic element or a container element. A link represents a relationship between two nodes. The layout elements are presented as "attributes" of a node in the CRT. Each semantic element can have multiple attributes and the combination of these attributes determines how the element is to be presented.

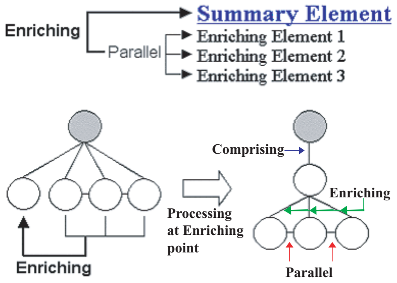
3.1 Relationship links

There are three types of relationship links in the CRT: Comprising, Parallel, and Enriching.

- Comprising: This is the relationship between a container element and each of its constituent components. In the CRT, it is represented by a vertical parent-child link between the container node and each of the component nodes.
- Parallel: In a web document, some semantic elements are complementary to each other in jointly presenting some information. These elements usually have similar outlook and functionalities. Parallel relationship is the relationship between these complementary elements. In the CRT, it is represented by a horizontal link between each pair of parallel semantic nodes/container nodes.
- Enriching: Some elements provide little and very brief information while some other elements expand on the information by providing further, more detailed information directly or through hyperlinking. Enriching relationship is the relationship between these two types of elements; the former type is called the summary element and the latter type an enriching element. In the CRT, enriching relationship is represented by a vertical parent-child link between these two semantic nodes/container nodes.

Figure 1 shows an example, where the CRT on the right hand side after the processing exhibits the three types of relationship as mentioned above. The shaded node represents the container element; the others are semantic elements.

Figure 1. CRT with three types of relationship.



3.2 Property matching

The comprising relationship can be discovered with a simple tree traversal, but the other two relationships need more intelligence. Each semantic element has a list of property attributes to describe how its content is to be presented. For a semantic element with n property attributes, we associate an n-dimensional property vector with it.

$$\text{property vector} = (p_1, p_2, \ldots, p_{n-1}, p_n)$$

To determine whether two semantic elements are parallel, their property vectors are compared for the degree of similarity. A similarity function, $S(p_i, q_i)$, is used, where p_i and

q_i are property attributes. If the difference between them is smaller than a certain threshold, $maxdiff_i$, they are considered to be similar (and the function returns 1). The parallel matching score is then found by taking a dot product between the similarity vector, s, and a parallel weighting vector, w, where w_i represents the relative importance of the i-th attribute.

$$\begin{aligned}\text{parallel matching score} &= s \bullet w \\ &= \sum s_i w_i\end{aligned}$$

If the parallel matching score is larger than a threshold, t_p, then the two semantic elements are said to have a parallel relationship.

Given a pair of container elements such as the table cells in HTML, to determine whether they are parallelly related, some more comparisons are required. The number of different types of element, e.g., image or text, etc., is first compared between the two container nodes. Then the parallel matching heuristic comparison for the semantic nodes can be applied sequentially to the elements in the container node pair.

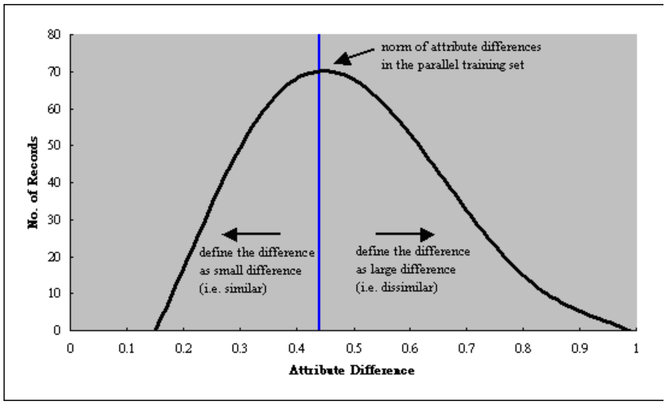
For two semantic elements having an enriching relationship, one element is identified as the summary element and the other one the enriching element. As the purpose of the summary element is to announce a subject while the enriching element is to provide more detailed descriptions, their layout properties could be quite different. In other words, to determine the enriching relationship, the same algorithm is applied but dissimilarity is considered this time. Similarly, the discovery of enriching relationship between two container nodes will use dissimilarity of types, numbers and properties of comprising elements.

3.3 Parameter Determination

For the parallel matching method just described, there are three types of parameters that need to be determined: the threshold $maxdiff_i$ for each property attribute, the parallel weighting vector w and the parallel threshold t_p.

To determine $maxdiff_i$, we use a statistical method with a training set of records. In our experiment, 500 parallel relationship records were collected and the difference of p_i and q_i for each record was computed. The distributions were plotted as a graph, as demonstrated in Figure 2. The norm ($maxdiff_i$) was applied to these differences to decide whether the attributes are similar or dissimilar. If the attribute difference is smaller than the norm, it is considered a small difference and the two attributes in question are treated as similar.

Figure 2. Distribution graph for determining attribute threshold.

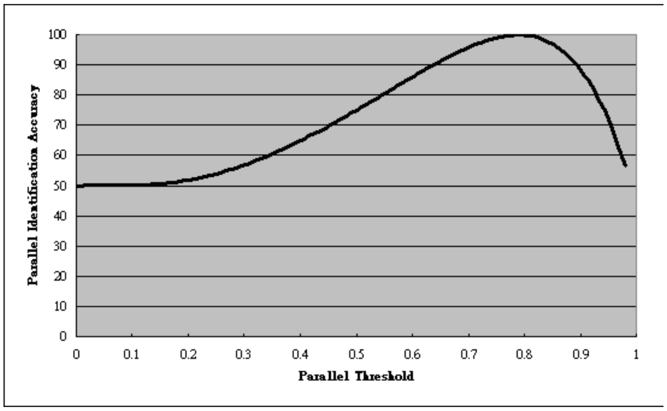


After determining the threshold $maxdiff_i$ for all the property attributes, the records in the parallel training set are input to the similarity function $S(p_i, q_i)$ to generate a set of similarity vectors. A counting vector, c, is then formed, where c_i represents the number of similar pairs (number of 1's returned) for the i-th attribute in the set of similarity vectors. The parallel weighting vector, w, is then found by:

$$w_i = \frac{c_i}{\sum c_j}.$$

The variation of identification accuracy with the value of the parallel/enriching threshold can be recorded. The identification accuracy is taken as the percentage of records identified correctly. The parallel threshold t_p is the point with the highest identification accuracy, as shown in Figure 3, which is 0.79.

Figure 3. Parallel relationship identification accuracy with training set.



With the results from the relationship extraction process, a CRT for the content (like the one shown in Figure 1) can be produced which will be used intensively in the following step where further content analysis is carried out.

4 Logical Section Discovery

A logical section is a group of semantic elements that exist as a unit. It can be separated from the original web document without affecting the semantics of both itself and other semantic elements. With the logical sections, a web document can then be split into pages and displayed in devices with a limited screen size. In other words, it can serve as an elementary building block in content adaptation and presentation. But to discover such a logical section is not an easy task since the relationships in some web documents could be quite complex. We propose a relationship-aware discovery methodology to tackle the problem.

A good logical section should have the following features:

- There is a strong coupling between the elements within the the logical section. These elements are closely related and should serve the same purpose. These elements should be presented with the same adaptation settings.
- Between logical sections, the coupling is weaker such that the sections can be presented in different styles if necessary.

Algorithm 1 Logical Section Discovery

```
LS-Discovery(testnode)
1.  if (testnode // with testnode's right siblings)
2.      LS $\leftarrow$ U(testnode, last // siblings)
3.    if (width(LS) $>$ PSA)
4.            LS $\leftarrow$ U(testnode, siblings with width(LS $<$ PSA))
5.    if (width(resultant horizontal LS's) $>$ PSA)
6.        add a line break
7.    if (LS cannot fit in $S_{remain}$ and there exists vertical // relationship)
8.        break the vertical // relationship $> S_{remain}$
9.        LS $\leftarrow$ U(testnode, last // siblings)
10. else
11.   add current node to LS
```

Algorithm 1 is the proposed logical section discovery algorithm that looks for the proper logical section to be used for negotiation in the next step. The algorithm first tries to obtain a set of parallel (indicated by // in the algorithm) siblings and puts them in the Logical Section LS. The union $U(a, b)$ is to perform node aggregation on the parallel siblings from node a to node b with all the descendants. If the current node exhibits no parallel relationship with others, then the logical section would contain only this node (line 11). If the horizontal dimension of the logical section (returned by the function $width$) exceeds the Preferred Scrollable Area (PSA) defined by the user, the logical section will be trimmed such that only those elements in the logical section with horizontal dimension smaller than PSA will be returned (line 4).

The remaining elements not in the trimmed logical section will become the target for the next iteration. There is also the case where the newly formed resultant set of logical sections (the current existing logical sections combined with the new one) exceeds the horizontal threshold; the new logical section will wrap below the preceding one (line 6).

The parallel relationships can exhibit different browsing behavior in different scenarios. For instance, in an HTML table, *td* (table column) tags are defined first, and the collections of *td*'s will form a *tr* (table row). Based on this observation, a very long table with many rows can be further adapted at the row level to suit the constrained page size. This helps to increase the page utilization. If the resultant logical sections cannot fit in the remaining spatial size, S_{remain} (spatial consumption of each page is constrained by a spatial threshold), the vertical relationship will be broken (line 8), such that the row elements can be more flexible in consuming the spatial size of the page and at the same time to increase the rendering capability of a large table in a very constrained mobile device. But simply fitting the remaining portion of a table in a new page would greatly reduce the content integrity as the other related elements (such as the caption of the table) are absent from the new content page. In our design, we try to detect any broken enriching point when a new page of content is created. The re-creation of the summary element preceding the broken enriching relationship will enhance the content integrity, as shown in Figure 4 (where the "MARKETPLACE" header re-appears in a continuing page).

5 Negotiation

The decision engine proposed in this paper executes a "negotiation" process between the data structure containing the user's preference information and a decision function of the engine.The negotiation process entails a systematic traversal of score nodes by a negotiation algorithm. A score node represents a particular rendering choice of a logical section. This is an iterative heuristic search to try to find the best score that meets the rendering constraints. The result is the most optimal score node found by the algorithm. The adaptation strategy as encoded in the node will be fed to the realization module to synthesize the actual version of the adapted content.

In our prototype, we use an Ordered Relation Score Tree (ORST) Negotiation Algorithm [5] to do such heuristic search, whereby the score node which is most preferable by a specific user will be explored first. This will lead to a good resulting score. Detailed explanation of the negotiation algorithm can be found in [5].

Algorithm 2 The Negotiation process

```
1.  Add all the nodes to the pool
2.  while (Post-order tree walk to find a node 'testnode' in the pool that is not Processed)
3.    LS = LS-Discovery(testnode)
4.    Get the score node's adaptation settings from score tree
5.    if $decision$(score-node, $P_d, P_n, LS$) != True
6.      go to 4 to get next node from the score tree
7.    else
8.      if (LS.first has broken enriching)
9.        add the summary node to LS
10.   Set LS to Processed
11.   Adapt(LS, score-node)
12.   if $decision$(score-node, $P_d, P_n, LS$) != True for all nodes
13.     Create a new page
```

With the components introduced above, the overall negotiation process can be summarized by Algorithm 2. A pool is maintained during the entire process for those nodes that have yet to be processed. A post-order tree walk is applied to search for a node in the pool (line 2). The Logical Section Discovery Algorithm is invoked and a proper logical section would result (line 3). The Score Tree Negotiation Algorithm will iteratively find a score node (including the adaptation settings) in the user-specific score tree that would give a True value at the Decision Logic. The goal of the operation of the decision logic is to find the best scoring node corresponding to a version of the content that is renderable given those real-time parameters. During the process to locate the optimal node, for each score node to be examined, the decision engine will generate a binary decision (True or False) based on the client device capability P_d, the network parameters P_n, the adaptation settings stored in the score node, and the Logical Section LS itself:

$$T \,||\, F \leftarrow decision(\text{score-node}, P_d, P_n, LS)$$

where *score-node* provides the setting of various quality axes. The binary decision True indicates that the content after transcoding according to the adaptation settings as specified in this score node is renderable in the target device with its particular capability in the current network environment. The binary decision False means otherwise. The decision function, $decision()$, interacts with the score node data structure iteratively in a negotiation until a satisfactory score node that returns True is found. If a broken enriching relationship is detected in the initial logical section, the summary node will be added to the logical section (line 9). The nodes within the logical section will then be set to 'Processed' (line 10) and the logical section will be adapted according to the adaptation settings encoded in the score node (line 11). If the logical section cannot give rise to a True score node after finishing the tree traversal, it means that the logical section may be too large to fit in the remaining space. In this case, a new page is created to render the section (line 13).

6 Web Document Adaptation System

We have implemented a prototype that hosts the discovery logic and the decision engine as discussed in the previous sections to demonstrate the practical viability of our approach. The system operates on web documents that are in HTML format.

A web page is treated by the system as follows. The whole web page is segmented into three pages according to its structure, as shown in Figure 4. After having included the logical sections for the table on the left hand side of the original web page (the 'SEARCH CENTERS' table) in the second page of the adapted web content, S_{remain} is deemed not large enough to hold all the elements from the table on the right hand side (the 'MARKETPLACE' table). The remaining space is only enough to render a subset of the table rows.In order to push the spatial consumption of each content page, the page utilization policy is to let the user get the most out of a single web page. The system breaks the table, but the broken enriching element is detected and the summary description (the summary node) will appear in the new, continuing content page again so as to maintain content integrity.

The two partitions containing an image and a text description side by side near the bottom of the original web page are identified as parallel due to the similarity in their layout. But the horizontal arrangement of the table cells exceeds the PSA value, and so the concerned logical section is wrapped.

Figure 4. Adapting a web page: (a) the original page; (b) page 1 of adapted content; (c) page 2; (d) page 3.

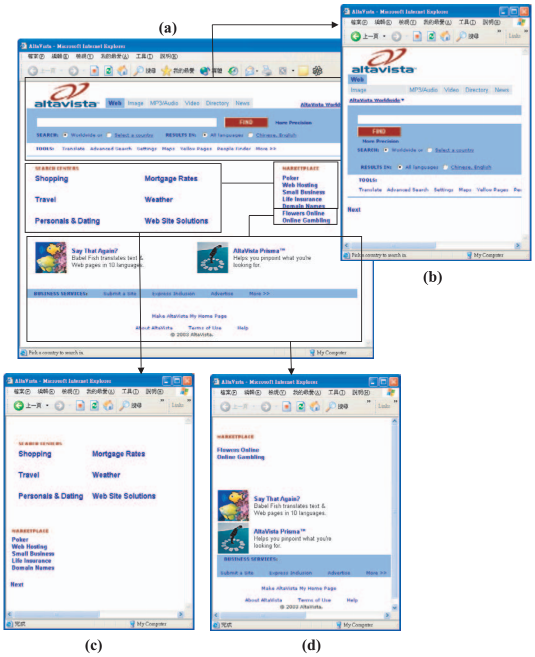
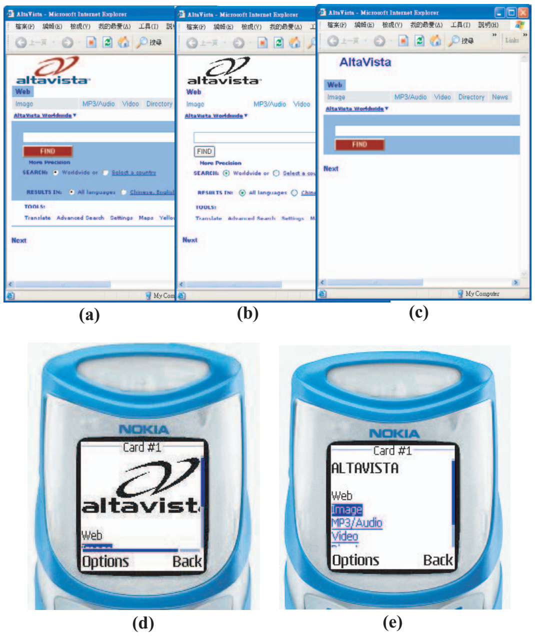


Figure 5. Context-awareness: (a) user prefers color; (b) user prefers delivery time; (c) more stringent delivery time; (d) large bandwidth on WAP phone; (e) small bandwidth.



Other than adapting the page to a small screen size, we also carried out experiments to show the adaptability of our system to different types of contexts, as shown in Figure 5.

7 Conclusion

Our system design incorporates two unique components: a relationship-aware content analysis engine and a negotiation model. We believe with these components, the system can deliver good adapted contents to the mobile clients with good user satisfaction. Our initial experiments have produced results that support our belief.

Acknowledgement

This project is supported in part by a Hong Kong RGC grant ("A holistic approach to structured web document viewing in small devices").

References

[1] O. Buyukkokten, O. Kaljuvee, H. Garcia-Molina, A. Paepcke, and T. Winograd, Efficient Web Browsing on Handheld Devices using Page and Form Summarization, ACM Transactions on Information Systems (TOIS), Volume 20, Issue 1 (January 2002), pp. 82–115.

[2] O. Buyukkokten, H. Garcia-Molina, A. Paepcke, Seeing the whole in parts: text summarization for web browsing on handheld devices, WWW10, May 2001, Hong Kong, China, pp. 652–662.

[3] N. Ashish, C. Knoblock, Wrapper Generation for Semi-structured Internet Sources, ACM SIGMOD Record, 26(4), 8–15, 1997.

[4] J. Chen, B. Zhou, J. Shi, H. Zhang, F. Qiu, Function-based object model towards website adaptation. WWW 2001: 587-596.

[5] W.Y. Lum and F.C.M. Lau, A Context-Aware Decision Engine for Content Adaptation, IEEE Pervasive Computing, Vol. 1, No. 3, July-September 2002, 41–49.

[6] W.Y. Lum and F.C.M. Lau, On Balancing Between Transcoding Overhead and Spatial Consumption in Content Adaptation, Mobicom 2002, Atlanta, USA, September 2002, 239–250.