

Personal Webs for Efficient Web Browsing on Wireless Handheld Devices.

Prashanth Pappu
prashant@arl.wustl.edu

Abstract

This paper describes a novel architecture and mechanisms for creating users' *personal webs*. A personal web is essentially a template, of a reduced and personalized version of the World Wide Web, which reflects the users' browsing routines and interests. By considerably reducing the effort expended by a user to reach desired content and significantly reducing the size of the actual content transferred to the user, personal webs prove invaluable in enabling web browsing on devices with limited capabilities, like Personal Digital Assistants (PDAs) with low bandwidth, small display and slow CPU.

We build these personal webs using HTML Page Templates (HPTs) which are essentially XML documents validated by a simple grammar, to capture the *layout* and the *static* and *dynamic content* of a webpage. The various elements of the HPTs and their attributes have been designed to automatically determine, arrange in simple hierarchies and retrieve, the right content of interest to the user from a variety of web pages. This paper while presenting the personal web architecture and the use of HPTs, also describes mechanisms for the automatic creation of these templates and their use in enabling efficient web browsing on wireless handheld devices.

1 Introduction

The proliferation of wireless handheld devices like, Personal Digital Assistants (PDAs) [4] and Pocket PCs [5], and the increasing need to enable them to act as clients in the World Wide Web, has resulted in a great demand for *wireless content*, which can be accessed and viewed on these devices.

Almost all of the existing HTML content of the web was designed to be viewed on desktop computers. Handheld devices like PDAs and Pocket PCs are fundamentally different from desktops. These devices have small screens, low processing power, bandwidth and memory and also slow text input facilities. For example, a typical PDA like *Palm VII* has only 160x160 pixels screen space, and a stylus for entering text with optical character recognition. The limited screen space in particular demands that the content presented to the user on a wireless handheld device be highly relevant and very

concise. A user cannot be expected to scroll excessively, to reach, view and traverse links provided on a small screen [8].

Several efforts [6, 7, 1, 3] have been made to enable wireless web access on handheld devices. A simple solution [6, 7] would be to just split and reformat the HTML pages to fit in the small screens of these devices. While this solution provides access to all the existing content of the web to the user, the need to scroll up and down excessively to view content, and the severe bandwidth and processing requirements to display the content, makes it problematic both for the user and the device.

Most existing content providers for wireless handheld devices, themselves create content specially formatted and optimized for these devices. This process can actually be automated by annotating existing HTML content [3], which can be interpreted by a transcoding proxy to re-render this *chosen* content on wireless handheld devices. While this *annotation approach* eliminates the problems arising due to small screen space, the user is highly restricted in the choice of content. The user has access to only a few sites from which limited content has been reformatted and made available by the content providers.

An interesting third approach [1] involves separating the navigation and viewing phases of browsing. In the navigation phase, the user is shown only the link structure of the page, to reduce the amount of information sent to the user and encourage him to explore the links to reach content of interest. When the user finally reaches desired page, the content of the page is shown to the user. The links in the navigation phase themselves are ordered using page ranking schemes [2]. Even this approach inundates the user with link information of whole pages which can easily run into hundreds for a single page, forcing the user to search for the right path of links to reach desired content. Also since the complete HTML page is shown to the user in the viewing phase, the user again has to scroll around to read the content.

While these approaches attempt to reformat existing content and their hierarchies to suit the device, it is also equally important that the links and content presented, be highly relevant for the user.

The personal web architecture introduced in this paper, is a novel architecture designed to enable users to view and navigate most of the content of their interest from any site. The architecture does not require any changes on the server side nor any special inputs from the user in creating this content. The model requires the user to download a client-side module which integrates itself with the users' browser at his desktop. This module at users' desktop, observes the users routines and actions as he browses the web and creates his own *personal web*. A personal web is a highly reduced and efficient version of the global web with templates which capture the right content of interest to the user at various sites, arranged in simple hierarchies, to eliminate undesired content and links. The user can navigate and view the updated content from his personal web anytime from his handheld device. While the user cannot view arbitrary content from a site, he can view almost all of his favourite content from any site which he has visited from his desktop. Thus the user can access through his wireless device, highly relevant and useful content formatted for handheld devices, automatically, with no changes made to existing HTML content.

This paper, while introducing the concept of personal webs for efficient wireless browsing also describes in detail the mechanisms enabling this architecture. Section 2 puts forth some arguments about the relevance of personalized and reduced versions of existing content for handheld devices. Section 3 describes the Personal Web architecture. Section 4 gives details of various algorithms used to enable the architecture.

2 What makes good wireless content ?

The limitations in the capabilities of handheld devices dictates that content directed at these devices be greatly honed. More importantly, the context in which these devices are used demands that content not only be greatly reduced to fit in the small screens, but also be highly relevant for the user.

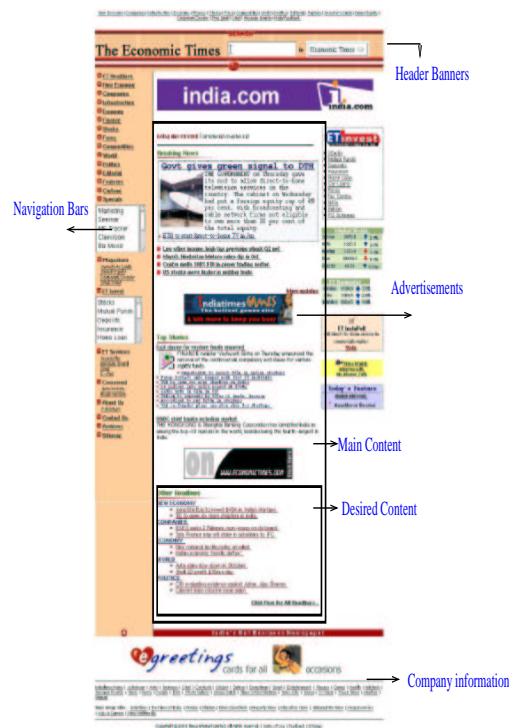


Figure 1: HTML page showing header, navigation bars, company information, advertisements, main content and content of interest to user.

Good wireless content should hence have

1. *Reduced Hierarchies*

Most web sites categorise their content in hierarchies to make navigation at their site easier for all users. For example, *www.cnn.com* uses 20 main categories, each of which has *atleast* 3 to 4 subcategories to present its content. But most users are interested in only a few of the subcategories at a site. These hierarchies force most users to visit many intermediate pages only to access more pages from them. It is important that the content presented to a user on a handheld device have very few and highly relevant categories for the user.

2. Reduced Pages

Existing HTML pages carry a lot of information apart from their *main content*. Most pages carry header banners, left and right navigation bars, advertisements, company information etc. Moreover a single user is usually interested in only some parts of the content of the page.

For example, Fig. 1 shows a snapshot of a page from a popular online business news paper. The region demarcated as content of interest to the user shows that the user is more interested in the top stories section of the page rather than in other features.

It is clear that the whole page as such is not of much use to the user and hence would be more efficient to transfer and render just the parts of the page of interest to the user.

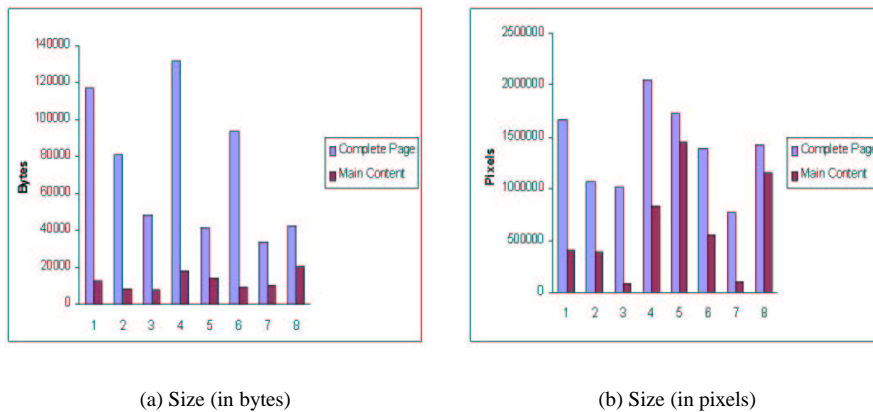


Figure 2: Comparing complete HTML pages and their main content.

To quantify these observations, consider Fig. 2(a) and Fig. 2(b) which plot the size of various webpages and their main content. The figures were obtained by annotating various popular webpages to delineate their main content and by using a simple rendering program to determine the actual size both in bytes and in terms of pixels required to show them to the user. The various webpages

were chosen from popular news, entertainment, lifestyle, magazine, university and corporate web sites like *cnn.com*, *cnet.com*, *msn.com*, *salon.com*, *time.com*, *wustl.edu*, *sun.com* etc.

These plots show that an average page has a lot of content of no interest to the user. It would greatly help the users if these pages were reduced to reflect their interests when being shown on handheld devices with small screens.

3 Personal Web Architecture

The personal web architecture is a novel architecture to enable efficient web browsing on wireless handheld devices.

A personal web is essentially a reduced, relevant and efficient version of the global web created according to a users' personal browsing routines and preferences. These personal webs reflect the fact that

- Users are interested in the content only in a few categories at a given site.
- Users visit many intermediate pages only to reach other pages of interest from them.
- Most webpages carry along with their *main content*, a lot of adjoining navigation menus, company information, advertisements, header banners etc, which the user is not interested in.
- Users follow only a few links of interest in the *main content* of a web page.

Personal webs are essentially *templates* designed according to users' routines which effectively capture the content of interest and relevance to the user, arranged in appropriate hierarchies while eliminating most of the undesired content. Fig. 3 shows the personal web architecture. Users download a client side module which resides on their desktops and observes their browsing routines and actions. This information is then used to create and update the users' personal web and communicated to a central server. Users with handheld devices connect to a wireless proxy server which uses information in their personal webs to retrieve updated content from global web and present the same to the users.

4 Personal Webs

4.1 HTML Page Templates

The personal web of a user is not built with HTML pages but with their *templates* called HTML Page Templates (HPT). HPTs are documents which capture the *layout* of a page, the *static* and *dynamic* content of each of the *regions* of the page and can also record and retrieve particular regions of the page according to the users' interests.

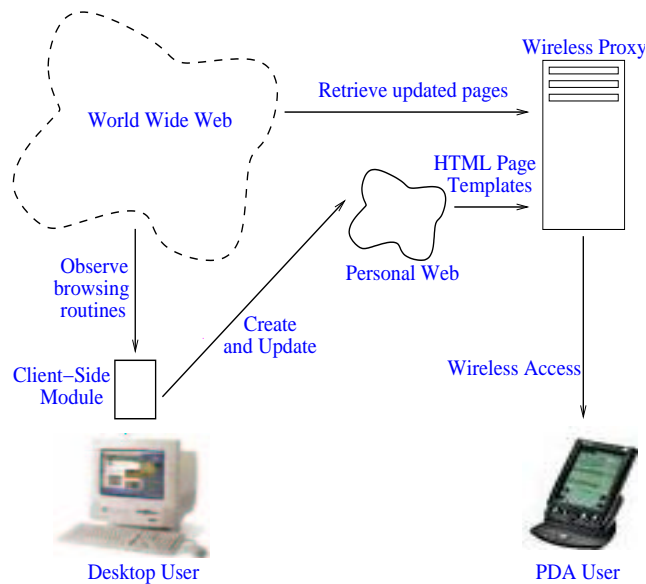


Figure 3: The Personal Web Architecture.

A HTML Page Template is not just an annotation page, like (REF maf, RSS) and is a *template* which can automatically annotate a corresponding HTML page even in the presence of any changes to the content and layout of the page.

While a static annotation page like RSS can be a simple RDF (ref) document, HTML Page Templates also need to capture the layout of the page and are essentially XML (ref) documents validated by their own grammar resulting in HTML Page Template Language (HPTL). The regions of various templates are hierarchially linked to each other to create a users' own personal web of templates.

4.1.1 HTML Page Layout

HTML page templates capture the layout of a HTML page by dividing it up into *regions*. Each of these regions is further divided into more smaller regions. For example, the page in Fig. 1 can be divided into 3 top regions each being a HTML table as shown in Fig. 4. Region table 2 is again divided into regions table4 and table5 and so on.

In general, a region of a template is either a table, its rows, columns or even cells, any text-block like a paragraph, preformatted text or block quotes or an ordered list or just one of the items of the list of the corresponding HTML page. The complete hierarchy of the regions (*a tree*) defines the layout of the page.

All the regions of the HPT end up holding *content*. The various regions and their content are together referred to as the *elements* of the HPT. Each of the content elements held by a region can be *static* or *dynamic*.

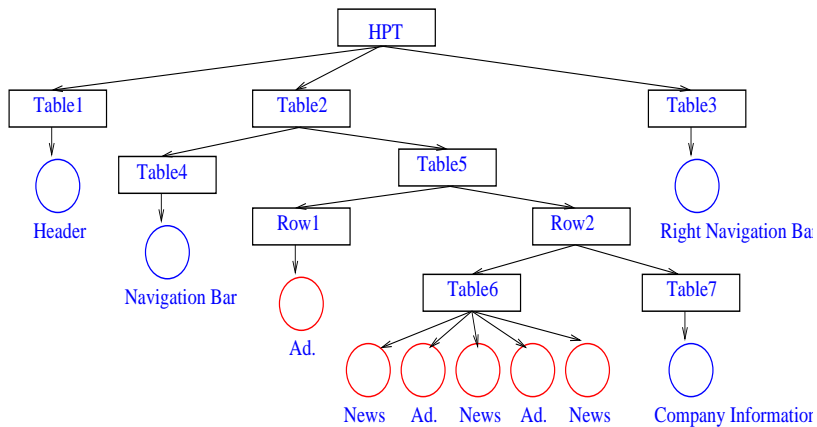


Figure 4: HTML Page Templates

4.1.2 Static and Dynamic Content

Consider, for example, the links in the navigation bars in Fig. 1. Most of these links like *Economy*, *Finance*, *Stocks* etc are static content in the sense that these links will most probably still be available even if the page is updated, whereas the links in the *top stories* section of the page are dynamic content, replaced by new stories and links the next day.

Each of the regions of a HPT can hold content elements which can be static or dynamic. In a HPT, while simple attributes like `text`, `links`, `images` etc can be used with static elements to classify and represent the actual content of the page itself, dynamic content is represented by holding *properties* of the content in the attributes of the *dynamic* element. These attributes can hold simple *statistical properties* like

```

numLines;    /* Number of lines in the region */
numImages;   /* Number of images in the region */
numChars;    /* Number of characters in the region */
numLinks;    /* Number of links in the region*/

```

etc, or they could be more sophisticated properties like *semantic signatures* capturing the ideas and concepts in the content.

Static and dynamic content of a region, together help in determining, *what to expect in an updated version of the page*.

4.1.3 Attributes of regions and content

In addition to the attributes of the *static* and *dynamic* elements discussed in section 4.1.2, the various elements of a HPT have the following attributes.

- *Identifier Attribute*
All regions and content elements have an identifier attribute `id` whose value is an

integer, to identify them uniquely within a template. Also each HPT in the users' personal web is also assigned a unique integer identifier. Hence any element can hierarchially be addressed by first using, the template identifier and then the element identifier. For example, an identifier of *12.14* refers to the element with identifier *14* in template *12*.

- *HTML Attributes*

All regions (tables, blocktext, lists etc) also carry their HTML attributes. For example, a table is still characterised by its *width*, *height*, *rowspan* etc.

- *Element Retrieval Attributes*

Also the elements (regions and content) have two additional common attributes.

1. *contains* - Space separated string of elements.
2. *isDestination* - Boolean, true if element is of interest to user.

The attribute *contains* represents the regions of interest to the user which are *contained* in a region. For example, if the user is interested in the content in a particular cell of a table, the corresponding table will have its *contain* attribute point to this particular cell. The *contain* attribute helps a retrieval algorithm in reaching the right regions of interest to the user in a HPT.

Also all elements which are of interest to the user have their *isDestination* set to *true*. The content (both static and dynamic) elements have only the *isDestination* attribute. Regions or content which have their *isDestination* attribute set can either be *links* or *text content* of interest to the user.

- *User Interest Attributes*

Each element of a HPT which has its *isDestination* set to *true*, also has additional attributes which represent the users' interest in the element. These include

- *timeSpent* - A low-pass filtered estimate of time spent by the user on the element.
- *frequency* - A normalised fraction denoting the number of times the user has visited this element compared to other elements of the page.
- *recency* - A normalised fraction denoting how recently, the user has visited this element compared to other elements in the web site.

These attributes are used in ranking the various templates to eliminate intermediate pages and also to present the content in the right order of interest to the user.

- *Linking Attributes*

Elements with *isDestination* attribute set to *true*, use *nextElement* attribute to link to elements of other templates pointed to by the links in the element.

- *Element Equivalence Attribute*

This attribute is used to indicate that a Element a , of a template HPT_b , is equivalentTo another element x in a template HPT_y .

$$(b.a) = (y.x)$$

The equivalence of two elements is used to indicate that the two elements are *completely* similar. For example, complete navigation bars appearing in two different pages of a web site are usually equivalent regions and so are the header banners, company information bars etc. The equivalence of two elements also works at a finer granularity, showing for example, the equivalence of two links appearing in two different pages.

This attribute plays a very important role in the creation of the personal webs for two reasons

1. The use of equivalentTo attribute potentially reduces the amount of information stored in a user's personal web. Every occurrence of an element a equivalent to another element x in any HPT, is referred to the *original* (first encountered) element x using this attribute.

Hence, any user interests in this region, are recorded and updated in a single template, even though the user might have visited the same element in different HTML pages.

2. More importantly, the use of the equivalentTo attribute aids in the creation of a *tree* of HPTs reflecting the user's interests from a general directed and connected graph of HTML pages.

For example, consider Fig. 5 showing three HTML pages A , B and C , with page A having links to page B and C , and page B also pointing to C . The links in these pages will potentially be placed in their own content elements to record the user interest properties for each of the links separately. If the user visits page A first and then later, visits C from B , the element carrying the link pointing to C in B just refers to the original element encountered in A , and updates this element with the recorded user interest properties.

In a *tree* of HPTs, all equivalent elements of various templates point to a single instance of that element referred to as the *primary element*. All other equivalent elements are called *secondary elements*. In Fig. 5, the element C in HPT A is the primary element, while the element C in HPT B is a secondary element.

4.1.4 HTML Page Template Language

HTML Page Template documents are written in HTML Page Template Language (HPTL) and are essentially XML documents validated by a simple grammar with rules for capturing the layout of the page and assigning the right attributes to various elements of the page.

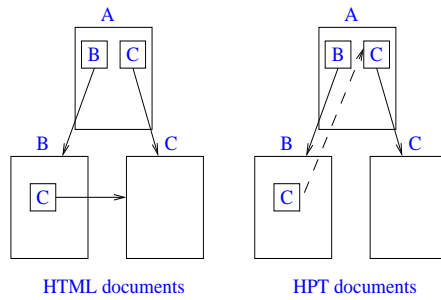


Figure 5: Use of equivalentTo attribute

Consider the HTML page template shown in Fig. 7. This template of a fictional page has two elements of interest to the user. Both are held in the cells of a table. The first one is a set of dynamic news links. These links are retrieved from the updated web page and presented to the user. When the wireless user traverses these links, the nextElement 10.33, element numbered 33 in template 10, is retrieved and shown to the user. The second is a static link to Football, which when followed shows content using element 15.6.

4.2 Creating Personal Webs

4.2.1 Client-side Module

As mentioned before, the personal web of a user is automatically created by a client-side browser module, by observing the routines and actions of the user at the desktop.

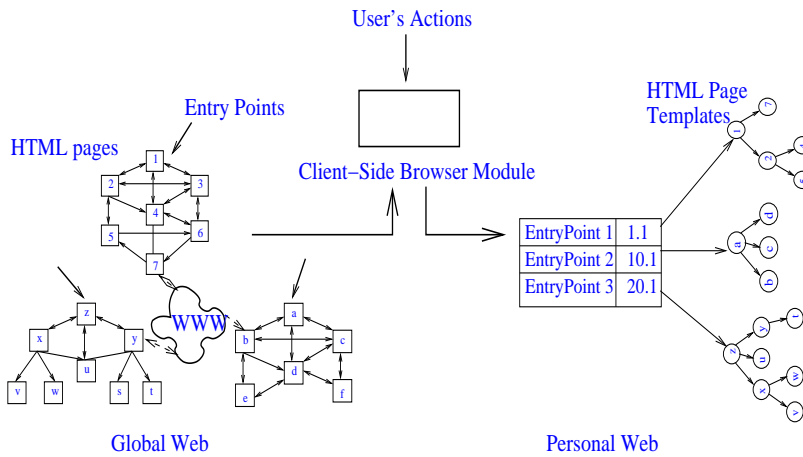


Figure 6: The client-side module.

Fig. 6 shows the function of the client-side module. When the user initiates browsing by following a bookmark, or by entering a url in the browser, the module sees the content available for the user to browse as a connected graph of urls. Each node of the graph corresponds to a distinct url and is characterised by its *content* and *links*. Since most webpages at a site can be reached from a number of other pages, using for example the navigation bars provided in the pages, the module views a directed graph.

The user visits various pages starting from the entry point. These pages can be

1. Content pages, carrying content of interest to the user.
2. Intermediate pages, carrying links to other pages of interest to the user.
3. Both content and intermediate pages, carrying some content and pointing to other pages of interest.

For each entry-point of the user, the client-side module creates a *tree* of HPTs reflecting the user's interests. Each HPT carries several elements containing content and/or links of interest to the user. Though, the user might traverse arbitrary paths through the graph of available HTML pages, the use of `equivalentTo` attribute leads to the creation of a simple tree of these templates.

The personal web created by the module for the user, is a set of entry points, each pointing to a tree of HPT documents. Different websites are potentially represented by different entry points in the users' personal web and occasional movement between completely different websites is not reflected in the personal web due to the use of *user interest attributes*.

4.2.2 Creation of HPT documents

When the user visits a HTML page which does not have a corresponding template page in the users' personal web, a new HTP document is created for that page. The creation of this template page is straightforward. The module first, uses a parser to identify all the regions of the page and attempts to establish the equivalence relation between the various elements of this page and the elements of other HPTs corresponding to this entry point. It then treats all unmapped content as being static, records user actions (links traversed!) to identify elements of interest and assigns the relevant `nextElement` and user interest attributes to the corresponding *primary* element. Finally, the `contains` and `isDestination` attributes are set for various elements, for later use in retrieving elements of interest.

When the user visits a HTML page which has a corresponding HPT in the users' personal web, a temporary HPT (HPT_{tmp}) is first created for that page and used to update the existing HPT (HPT_{old}), using a `templateUpdate` algorithm shown below.

```
templateUpdate(p1, p2)
HPT p1, p2;
{
```

```

    Set[Elements] e;
    e = templateMatch(p1,p2);
    updateCommonRegions(p1,r);
    updateUserInterestAttrs(p1,p2);
}

```

The algorithm first attempts to match the templates, to find their common elements using a `templateMatch` algorithm discussed in further sections. This algorithm does not expect the elements of the two pages to be completely similar and matches the elements in the best way possible. It then updates the common elements of HPT_{tmp} , with new static and dynamic content, by comparing these elements with the corresponding ones in HPT_{old} . Finally, it updates the user interest attributes using,

$$t_{new} = t_{old} * \beta + \tau * (1 - \beta)$$

where, t_{new} is the new `timeSpent` attribute, t_{old} is the `timeSpent` attribute from HPT_{old} , β is the `recency` of the region, and τ is the recorded time spent by the user before going to the next page. The `recency` attribute itself is updated for all the HTPs corresponding to an entrypoint, when the user follows another entrypoint. While all the elements visited in the session by the user, are given a `recency` factor of 1, the `recency` of all the other elements marked as *destinations* but not visited by the user are decremented linearly and all the `isDestination` attribute of all elements with a `recency` of 0 is set to *false*. The use of the above equation in calculating `timeSpent` ensures that a higher `recency` factor gives greater weight to the history of the user at the page. The `timeSpent` is an important attribute in determining the importance of the page to the user.

4.3 Using HPT documents

The HTML Page Templates are of limited use if the updated content from a corresponding HTML page cannot be retrieved in the presence of changes to the layout of the page. The content retrieval algorithms described in this section are designed to use the unique attributes of the elements in HTP documents to effectively retrieve updated content. To match and retrieve desired content, a temporary HPT is created for the HTML page and matched with the existing template.

4.3.1 Matching Elements

Matching two elements $e1$ and $e2$ forms a basic block of the other algorithms described in this section. All the matching functions described, return a *confidence* value between 0 and 1, with a value closer to 1, representing a higher probability that $e1 = e2$.

1. *Matching static content elements*

The confidence value increase with the *common strings* found in the `text` and `links` attributes of the elements. A match in the `images` attributes significantly increase the confidence value.

2. *Matching dynamic content elements*

To match two dynamic content values, the various properties of the elements are compared, and the confidence value decreases with the percentage increase in the difference between various corresponding attributes of the two elements.

3. *Matching Regions*

To match two regions

- First, their HTML attributes are compared. A close match in the attributes (like equal width in two tables) gives higher confidence value.
- Then, all the content appearing in the region (including its child regions) is aggregated into a single *sequence* of static and dynamic content elements. These *sequences* of content elements are then compared using a simple *approximate sequence comparison* algorithm which allows two continuous elements of one sequence to match , two other elements in the other sequence which are within a *window* of error. The confidence value is determined by the fraction of matched elements.

4.3.2 Retrieving Elements

Element matching algorithms are used in creation and update of HPTs (in `templateMatch`) and also in retrieving them. There are two different cases involved in the problem of retrieving new content - retrieving dynamic *links* and retrieving dynamic *text*. The problem of retrieving the right links of interest to the user is guided by the use of `contains` and `isDestination` attributes, but the problem of retrieving text is more difficult due to the lack of any hints about the location of the elements containing this text.

1. *Retrieving Links*

When the user desires to receive content corresponding to a dynamic content element containing links, an *eliminate and match* algorithm is used to retrieve the right links from the temporary template of the HTML page using the existing template. The *eliminate and match* algorithm *recursively* tries to find a corresponding element for the element with `contains` attribute in the existing template. On finding such an element, the algorithm first eliminates all its child elements which match elements without the `contains` attribute and then again searches for a match for the desired element in the remaining elements. The algorithm recurses till it reaches the right `destination` element, or till it has eliminated all the elements it can match to other regions. In the case that, the algorithm does not find a right match, the elements left after the elimination process are aggregated and returned as the desired region. This algorithm while *always* presenting the desired links to the user, potentially presents some *other* links also, when the layout of the page changes. While pure matching algorithms can retrieve links from a HTML page if there are no changes to the layout, the

complete information contained in HPTs combined with the elimination methods used, ensures a high probability of finding the desired content even in the presence of changes to the HTML page.

2. Retrieving text

The bigger problem of retrieving the just the right text from an article in a HTML page is handled using an approach similar to the *eliminate and match* algorithm with a difference that the *matching* is replaced by a search for *longest text segment*. The algorithm first, eliminates all regions that it knows are *not* the desired regions. The algorithm considers any region with `equivalentTo` attribute set, as undesired regions, as it does not expect to find text of interest to the user in regions appearing in multiple pages. The algorithm returns the longest text segment from the elements left after the elimination process.

5 Wireless Browsing

When the users access their personal webs by connecting to a wireless proxy server, the users are presented all their favourite sites to browse. When the user follows a link corresponding to an entry point, the related tree of HPTs is used to retrieve the content for the user. The simple hierarchy of the HPTs makes it possible to present highly reduced and relevant links to the user.

Also,

- At any level of the hierarchy, all links presented to the user are from *static or dynamic* content whose `isDestination` is set to true.
- A link presented from *static content* leads to more links and/or just *pure* content.
- A link presented from *dynamic content* leads to *pure* content (without any links).
- The links from various destination regions of the HPTs corresponding to an entry point are themselves presented in order of their corresponding HPTs in the tree corresponding to the entry point.

All the links are named after the text appearing in the HTML page. For a small set of stop-gap links like *full story*, *click for more* etc, the user is also presented the text in the immediate parent region to put the links in the right context.

The personal webs, are absolutely invaluable and users' can browse exactly the desired content from any of their own favourite sites on wireless handheld devices.

6 Conclusions

In this paper, we presented a novel architecture for enabling efficient web browsing on wireless handheld devices like PDAs. We introduced the concept of personal webs built with HTML Page Templates. The HPTs with their layout and content elements

use a variety of attributes to reduce and capture a users' regions of interest in various websites. We also presented algorithms to effectively use these HPTs to retrieve the right content for the user.

The personal web architecture together with the enabling HTML Page Templates, the update and retrieval algorithms provide highly relevant and reduced versions of the users' favourite sites automatically on their handheld devices.

References

- [1] Buyukkokten, O., Molina, H. G., Paepcke, A., Winograd, T. "Power Browser: Efficient Web Browsing for PDAs".
- [2] Cho, J., Garcia-Molina H., Page L., "Efficient Crawling Through URL ordering", in *Proceedings of the 7th International WWW Conference, 1998*
- [3] Hori, M., Kondoh, G., Hirose, S., Singhal, S. "Annotation-Based Web Content Truncoding"
- [4] Palm Inc. <http://www.palm.com>
- [5] Pocket PC. <http://www.microsoft.com/mobile/pocketpc/default.asp>
- [6] Pocket Internet Explorer.
<http://www.microsoft.com/mobile/pocketpc/bgide/features/ppc/pie.asp>
- [7] ProxiNet, Inc., Proxiweb. <http://www.proxinet.com>
- [8] Kawachiya, K. and Ishikawa, H. "Improving Web interaction on Small Displays" in *Proceedings of the 8th International WWW Conference, 51-59, 1999*

```

<HPT id="8">
  <TABLE table_attrs contains="4 32" id="1" >
    <TR row_attrs contains="4 32" id="2" >
      <TD column_attrs contains="4" id="3">
        <TABLE table_attrs contains="4" id="4">
          <TR row_attrs contains="4" id="5">
            <TD column_attrs contains="4" id="6">
              <STATIC content="HeadLines" id="7">
                </DYNAMIC numLinks="5" numLines="6" numChars="270"
                  numImages="0" isDestination="true" nextRegion="10.33" id="8"
                  frequency="0.33" recency="0.27" timeSpent="0.22">
            </TD>
          </TR>
        </TABLE>
      <TD column_attrs contains="32" id="9">
        <STATIC content="Baseball Boxing Chess Cricket" id="10" >
        <STATIC content="Football" isDestination="true" nextRegion="15.6" id="11"
          frequency="0.12" recency="0.19" timeSpent="0.03">
      </TD>
      <TD column_attrs id="12">
        <STATIC content="Hockey Racing Soccer" id="13">
      </TD>
    </TR>
  </TABLE>
</HPT>

```

Figure 7: HTML Page Templates - Example.