SmartWeb – Web Content Adaptation for Mobile Devices

Péter Cserkúti, Zoltán Szabó, Tamás Eppel, János Pál

Budapest University of Technology and Economics
Műegyetem rkp. 3-9, H-1111 Budapest, Hungary

Abstract: This paper introduces SmartWeb, a proxy-based content re-authoring system for HTML pages. The purpose of this process is to adapt the original content to the capabilities of the client device. To achieve this goal SmartWeb performs some structural analyses and applies some transformations accordingly. SmartWeb is a robust and easily extendable system for content adaptation. It uses many kinds of algorithms that can cooperate with and rely on each other.

Keywords: content adaptation, content re-authoring, web page conversion, mobile devices, transformation, conversion, syntactical and semantic analysis

1 Introduction

The Internet contains a huge amount of information concerning all kinds of domains, and its size is growing rapidly. Today Google has got more than 8 billion indexed web pages [1], and it is estimated to cover just about 10% of the whole web [2]. A research shows that Internet – excluding the Deep Web – is growing by 10 million new, static pages every day. Exploiting the facilities provided by the development of desktop computers and the improvement of available network bandwidth web documents are getting more and more complex regarding both style and content. Thus rendering these documents may require a robust client device and large bandwidth. Besides, for the past few years mobile devices (mobile phone, smartphone, PDA, palmtop) have rapidly become an integral part of our lives. We can talk and send messages or video streams by them, and with an internet connection we can have access to any kind of web content. Accessing the Internet from mobile devices is becoming increasingly popular. Today there are over 1.1 billion web capable handsets in use worldwide – which is 63% of the total handsets of the world [3]. However, as can be read in [3] mobile web access is hindered by the fact that the majority of web contents are tailored for desktop computers. Compared to desktop computers mobile devices have got some evidential drawbacks. Their performance is usually much lower (processor,

memory), their screens are smaller, and generally their internet connections are much slower and/or more expensive than their peers'. Hence in the majority of the cases it is impossible to display the original web content, that is customized to a desktop computer and a 17" monitor, on the display of a handheld device in a usable and efficient way. Somehow we should adapt the original content to the capabilities of the client. There are several different existing techniques of this process that will be introduced later.

This paper introduces SmartWeb, a robust and extensible system for web content adaptation. SmartWeb provides a web page for the client where it can set the URL of the desired content. On the basis of the capabilities of the client SmartWeb then automatically re-authors the original web page to fit for the client device, and displays it. SmartWeb itself is a framework which uses different kinds of re-authoring techniques that can quite easily be extended. Basically the process of adaptation consists of two phases: content analyses and content transformation. Content analyses are used to determine the syntactically and semantically coherent parts of a page while content transformation aims to generate the adequate version of a web page for a client. Evidentially the results of the analysis process leverage the transformations applied to the original page.

The rest of the paper is organized as fallows. Section 2 describes the basic types and approaches of displaying web content in small screen devices. It then reviews the existing techniques and systems for content adaptation. Section 3 introduces SmartWeb and explains the main methods it uses to transform HTML pages to fit for mobile devices. Section 4 explains the algorithms of SmartWeb in detail: transformation algorithms, analytical algorithms – which we call block recognition algorithms – and conversion algorithms. Section 5 demonstrates the architecture of the system. Finally Section 6 traces the plans for the future.

2 Related Works

Document [4] gives a fairly good classification of techniques for displaying web contents in small screen devices. It forms the following groups: device-specific authoring, multiple-device authoring, client-side navigation, automatic re-authoring and web page filtering. In this paper we will introduce a new group named client-independent authoring and handle page filtering as part of the automatic re-authoring techniques. Now let us shortly describe these groups.

Device-specific authoring

Device-specific authoring means authoring a web page with a concrete client device in mind. Pages authored this way will perfectly be displayed in the client but the client can only view a certain set of pages in the adequate version. When applying this method adaptation is accomplished during the authoring phase.

Multiple-device authoring

Multiple-device authoring is relatively similar to device-specific authoring, but in this case the target client type is not a single device but a well defined set of devices.

Client-independent authoring

Client-independent authoring means a creation of a document in a way that it can properly be rendered on different kinds of client devices with different capabilities. It especially means placing comments and notes in the documents that basically control the process of rendering the page on the client side. The most typical means of this technique is RIML (Renderer Independent Markup Language) [11] [12]. In this case the adaptation is also accomplished during the authoring phase.

Client-side navigation

In client-side navigation the user has got the ability to change the portion of the web page that he wants to observe. A typical means for this is the scrollbar. Another solution is to let the user zoom in and out in the page. In this case the adaptation is accomplished on the client side during rendering time.

Automatic re-authoring

Automatic re-authoring is the most mature method for content adaptation. It takes the capabilities of the client and automatically, on-the-fly re-authors the requested page to fit for the client. On the basis of the place where the adaptation is fulfilled three classes can be formed: client-side, server-side or intermediary proxy-based adaptation.

The rest of this section covers some existing techniques and systems for content adaptation. Digestor is a web page filtering and re-authoring system [4]. It applies syntactic transformations (outlining transforms, first sentence elision transform, table transform, image reduction and elision transform and image-map transform) to reduce the size of the original content. To determine the adequate set of transformations for a page Digestor performs a depth-first-search in the document transformation space using some heuristics until it finds a "good enough" version of the document. [8] introduces the elISA project. It reveals a rule-based structural analysis technique for web pages. It defines an XML based rule langue which is capable of formulating patterns and transformations for them. elISA produces an XSLT document, which than can be applied to the original HTML page. A different technique is used by HearSay [7]. It performs both structural and semantic analysis on a web page and creates a partition tree accordingly. The partitioning is based on the fact that "semantically related items in an HTML document normally exhibit consistency in presentation style and special locality". During structural analysis it searches for structural recurrences, and for semantic analyses it uses lexical associations and ontologies. Function-based Object Model

is explained in [9]. It takes a very different point of view. Through an analytical process it tries to determine the functions of the objects in a page on the grounds of their visual preferences. For first it determines the set of basic objects then builds composite objects consisting of some basic objects and/or some other composite objects. Finally it classifies the objects into different categories.

3 Introducing SmartWeb

3.1 Overview

SmartWeb is a proxy-based automatic re-authoring system. Figure 1 depicts the whole basic process how SmartWeb works. It provides a web page for clients where they can set a URL for the desired content. In response to a client query SmartWeb downloads the original web page from the given web server and converts it at the client's leisure. The conversion process is driven by the client's *user profile*. The user profile contains information about the capabilities of the client device (display parameters, processor, memory), the available bandwidth and the user preferences. User preferences mean explicit intervention into the conversion process, for example users can explicitly define that they want no images on the resulting page or they can also set the desired font styles.

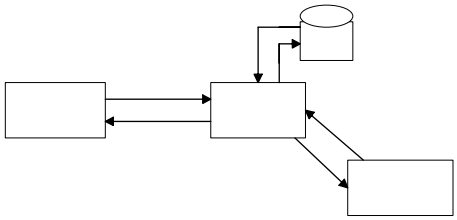

Figure 1
Adaptation process of SmartWeb

3.2 Steps of the Adaptation Process

The aim of the content adaptation made by SmartWeb is to simplify the style of the page and to get rid of the unimportant parts of it, thus it can be displayed easier on a small screen, will require less resources and lower bandwidth. The main steps of the adaptation process are as follows:

1 Build a specific tree from the HTML document that we call *transformation tree*.

2 Perform some *analyses* and *modifications* on the transformation tree.

3 *Assemble* the resulting HTML page on the basis of the transformation tree.

When building the transformation tree SmartWeb first converts the HTML document into XHTML since HTML is not well-formed and harder to handle. It then parses the XHTML document and builds the transformation tree accordingly. Every HTML tag in the document will become a node in the tree. A node has got many parameters. The *label* parameter holds the name of the tag and the *attributes* parameter holds the attributes of the tag that it represents. Nevertheless it also contains parameters that unburden the adaptation process. For example algorithms can attach notes to them which then can be used by other algorithms.

During the process of analysis SmartWeb tries to determine the semantically coherent parts of a page by structural examinations. The goal of the analysis is to recognize particular block types in a web page defined by specific patterns. That is why we call our analysis process *block recognition*. During the first phase of the project we manually examined several popular pages in order to be able to identify the most common block types and their particulars. The set of these pages was determined by statistics of webaudits.

During the modification of the transformation tree we commit changes on it in order to produce a version of the document that best matches the user profile. These modification techniques can freely rely on the results of the analysis. We group these modification techniques as follows:

Transformations

Transformation means any kind of simple editing of the transformation tree. A transformation can change the attributes of a node, and can perform predefined atomic transformations. There can be two types of transformations. There are ones that do not rely on the results of the block recognition and ones that do. The first group is called *automatic transformations* and the second is called *block recognition based transformations*. It is vital to distinguish these two groups as automatic transformations take place before the block recognition and block recognition based transformations take place just after that. The purpose of transformations is also dual. There are transformations that's main purpose is just to promote block recognition (for example by removing unnecessary parts of a page) and there are ones that's main purpose is to commit changes to get closer to the desired version of the document (e.g. image reduction).

Elisions

Elision means eliding a part of a document which is equal to deleting a node or a whole subtree from the transformation tree. Elisions may also rely on the results of the block recognition. Ones that do not are technically realized by transformations

and ones the do are realized at page assembling time by not including the respective block in the resulting page.

Conversions

Conversions are always supported by block recognition. Conversion means replacing an identified block of the document with a simplified version of it, thus making the document smaller and easier to display. Conversions are controlled by rules that will soon be introduced.

The following table summarizes the transformation tree modification techniques of SmartWeb with some examples.

	Transformation	Elision	Conversion
Does not need block recognition	- image reduction	- image elision	None
Needs block recognition	- applying a CSS for the identified elements of different block types	- advertisement block elision	- replacing a block with a block that only contains its essential elements

4 Algorithms

4.1 Block Recognitions

The purpose of block recognition is to determine the semantically coherent parts of a web page, that we call *block*s. For a human it is relatively easy to separate the different elements of a page, but it is difficult to formalize for a computer. To recognize blocks SmartWeb fulfills a structural analysis on the transformation tree. In our approach blocks can basically be recognized in two ways. A block can be a subtree in the transformation tree with a specific structure, and a block can also be a recurring part of the tree. In the first case we are searching for a *pattern*, in the second case we are searching for recurring subtrees. There is also a third algorithm for block recognition. There are block types which can not easily be recognized in a formalized way. The recognition rules for these block types are hard coded (e.g. LongText block type which is a simple text node with more than 50 letters in it.). The framework is designed in a way that these kinds of informal rules can very easily be added as well.

4.1.1 Pattern Recognition

The pattern recognition algorithm is capable of recognizing patterns in the transformation tree. Patterns are grouped into *block types*. Every block type has its own peculiar patterns. After identifying a block the algorithm marks its root node in the transformation tree. What do we call a pattern? A pattern determines the most typical elements of a block type and their structure as well. A pattern can be represented with a tree (Figure 2).

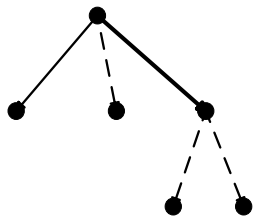

Figure 2
A pattern of a block type

A label of a pattern tree node can be of tree types. It can refer to a transformation tree node label or a block type name of a previously identified block (Headline, LongText) or it can be Any that covers all the nodes of the transformation tree. The weight of an edge in the pattern tree represents the maximum depth of a child node with respect to its parent. Edges has also got a style (weak, normal or strong) that determines weather two sibling nodes of the pattern tree can be in the same subtree under the parent node or not. Every pattern node has got a unique id. It is necessary for the conversion algorithms to be able to refer to an identified element of a block. These patterns become really powerful by the fact that a pattern can freely refer to an other pattern. Patterns can be stored in XML.

```
<Any id="1">
    <Headline id="2" maxdepth="2" linktype="normal"/>
    <LongText id="3" maxdepth="3" linktype="weak"/>
    <table id="4" maxdepth="1" linktype="strong">
        <img id="5" maxdepth="2" linktype="weak"/>
        <a id="6" maxdepth="1" linktype="weak"/>
    </table>
</Any>
```

4.1.2 Recurrence Analyzer

Recurrence analyzer searches the transformation tree for recurring subtrees. It must be given a threshold for the maximum depth of the subtrees that it should check. SmartWeb defines patterns also for recurring subtrees. These patterns also belong to block types called *recurring block types*. When the algorithm finds a recurring subtree it tries to match it with a pattern of a recurring block type. If it succeeds it marks the parent node of the recurring subtree.

4.2 Transformations

The purpose of transformations is to simplify the transformation tree before the block recognition (e.g. removing all formatting tags) and to apply changes on the transformation tree that will simplify the original document for the user (e.g. reducing images). Transformations usually do two types of things. They either change the attributes of a node or perform a predefined atomic transformation on it such as merging a node with one of its children or with its parent, deleting a node or inserting a new node between the node and its parent. Every transformation has got a *condition* and an *action* part. The transformation walks through all the nodes of the transformation tree and evaluates the condition for every one of them. When a node meets the condition it applies the action on it. Automatic transformations do not rely on block recognition. They have got a great importance as when block recognition does not work well for a certain page adaptation can still produce good results due to them.

The current version of SmartWeb uses the following transformations:

- Link and URL transformations
- Removing all the formatting tags of a page
- Image reduction and image elision
- Removing the styles of the nodes
- Applying a new CSS for the document
- Applying a new CSS for the identified block types

4.3 Conversions

The function of a conversion is to replace a whole identified block with a simpler version of it. It defers from transformations as it does not handle the transformation tree at the level of nodes but at the level of blocks. Every block was once recognized as a pattern. Conversions are driven by *rules*. A rule consists of two parts. First it defines the pattern that it should be applied to, second it

defines the new block that the old one should be replaced with. While the new block can have a very different structure it can also contain elements from the original one. The elements of the original block can be referenced by the id-s of the pattern tree nodes. The new block is a piece of XHTML code that is extended by references to the elements of the original block. For example the following definition replaces the blocks that were recognized by the pattern depicted in Figure 2 with a block that only contains the Headline subpattern and the image from the original block with a new layout.

```
<table>
<tr><td><FromPattern id=”2”></td></tr>
<tr><td>This block was replaced.</td></tr>
<tr><td><FromPattern id=”5”></td></tr>
</table>
```

5 Architecture

This section describes the architecture of the core of SmartWeb. It consists of two main separate modules: the Adaptation Engine module and the Concrete Algorithms module. The Adaptation Engine is just a framework for the whole adaptation process that is capable of handling the transformation tree and running different algorithms. The Concrete Algorithms module contains the algorithms that can be run by the Adaptation Engine. It includes transformations, block recognition algorithms, conversion algorithms, patterns and rules. This separation makes SmartWeb a robust system that can easily be tuned and modified when practices of web page development change.

The Adaptation Engine contains pipelines for every phase of the adaptation process. The suitable algorithms can be added to these pipelines respectively and they can also be removed from them. Basically these algorithms are responsible for the structural analysis and the transformation of a page. For block recognitions and conversions we always use the same methods: Pattern Recognition, Recurrence Analyzer and Pattern Conversion. These algorithms are always run for every adaptation request. However the adequate set of transformations is determined on the basis of the client’s user profile. It is the Negotiation Engine (not represented in Figure 3) that matches user profiles with transformation sets and parameters. Users can also request transformations explicitly. Once the transformation set is determined its elements are added to the appropriate pipeline and then the adaptation can be started. The point of this architecture is that it presents an easily extendable framework. By this approach new page processing or analyzing algorithms can easily be added to the system.

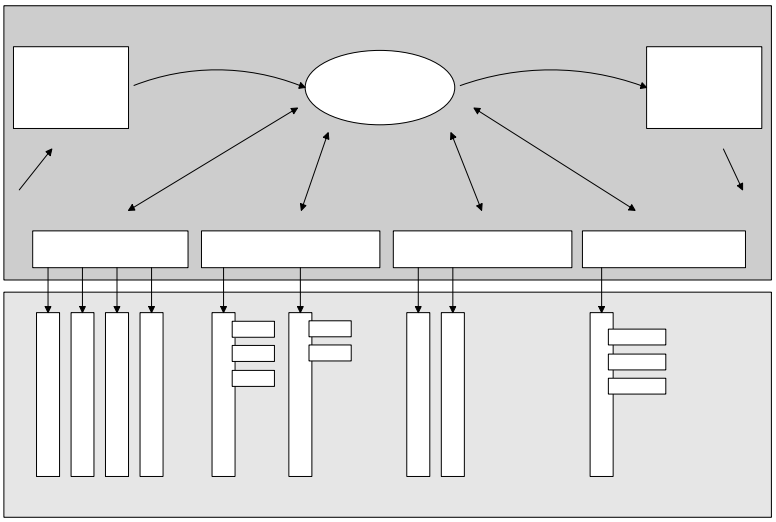

Figure 3
The architecture of the core of SmartWeb

6 Future Works

The current version of SmartWeb contains just a few transformations. In the future the range of transformations should be extended, new ones should be defined to achieve more efficient adaptation that results more useful pages on the client side. Conversion rules should also be extended. In the future conversions must be able to hide an identified block (or just some parts of it) by a link that points to a new page that contains the whole block (or a simplified version of it). Now SmartWeb does not handle client side scripting separately. An emulator should be applied that emulates the run of a script on the proxy side. This version is suitable for HTML page processing only. In the future we would like to improve it to be able to process the most common web content formats (e.g. pdf). Finally the process of searching in the document's transformation space (finding the adequate transformations and their parameters for a user profile) should be refined.

Conclusions

In this paper we introduced SmartWeb, a proxy-based content re-authoring system for HTML pages. We presented the main algorithms that it uses such as transformations, block recognition algorithms and conversion algorithms. We also presented the architecture of its core. SmartWeb is a robust and easily extendable

system for content adaptation. It uses many kinds of algorithms that can cooperate with and rely on each other.

References

[1] Metamend, www.metamend.hu

[2] Peter Faber: Google Increases number of indexed pages, November 10, 2004

[3] Philipp Hoschka: Vision and Ambition for the Mobile Web, in Proceedings of 'Mobile Web Initiative' Event, London, November 15, 2005

[4] Timothy Bickmore, Andreas Girgenshon and Joseph W. Sullivan: Web Page Filtering and Re-Authoring for Mobile Users, Computer J., 1999, Volume 42, Number 6, pp. 534-546

[5] Timothy W. Bickmore, Bill N.: Digestor: Device-independent Access to the World Wide Web, Computer Networks and ISDN Systems, 1997, Volume 29, Number 8-13, pp. 1075-1082

[6] Deng Cai, Shipeng Yu, Ji-Rong Wen and Wei-Ying Ma: Extracting Content Structure for Web Pages based on Visual Representation, The Fifth Asia Pacific Web Conference (APWeb2003), Xi'an, China, 2003

[7] Ramakrishnan, I.V and Stent, Amanda and Yang, Guizhen: HearSay: Enabling Audio Browsing on Hypertxt Content, In Proceedings International WWW Conference, New York, USA, 2004

[8] Fabio Vitali, Angelo Di Iorio, Elisa Ventura Campori: Rule-based structural analyses of web pages, in Proceedings of 6th International Workshop, DAS 2004, Florence, Italy, September 8-10, 2004, pp. 425-437

[9] Jinlin Chen, Baoyao Zhou, Jin Shi, HongJiang Zhang, Qiu Fengwu: Function-based Object Model Towards Website Adaptation, World Wide Web, 2001, pp. 587-596

[10] Wang Zhulong, Yu Hao, NISHINO Fumihito: Automatic Special type Website Detection Based on Webpage Type Classification, in Proceedings of International Workshop on Web Engineering, August 10, 2004

[11] Jari Näveri: Framework for device independent web application development, 2004

[12] Guido Grassel: An XHTML language profile for single source authoring to enhance the access from mobile devices to Web enterprise applications, in Proceedings of WWW 2003 Developers Day, Budapest, Hungary, 2003