Available online at www.sciencedirect.com
ScienceDirect
Journal of Network and Computer Applications 30 (2007) 563–585
Journal of NETWORK and COMPUTER APPLICATIONS
www.elsevier.com/locate/jnca

A web content manipulation technique based on page Fragmentation

Bouras Christos[a,b,*], Kounenis Giorgos[a,b], Misedakis Ioannis[a,b]

[a]*Research Academic Computer Technology Institute, Riga Feraiou 61, 26221 Patras, Greece*
[b]*Department of Computer Engineering and Informatics, University of Patras, 26500 Rion, Patras, Greece*

Received 28 September 2005; received in revised form 12 January 2006; accepted 24 January 2006

Abstract

Web portals today offer a variety of content and services to their users. This content can be split into various categories and usually content semantically related is placed in the same area. In this paper, a software technique is presented that allows the viewers of web sites to build their own personalized portals, using specific areas of their preferred sites. This technique saves users' time and reduces the cost of browsing the web by minimizing the volume of data that has to be downloaded. It is based on an algorithm, which fragments a web page in discrete fragments using the page's internal structure. Users utilize a web interface to define which parts of selected web pages they desire to appear in their personalized portal. No additional software needs to be installed on the users' personal computers, since this technique is designed to function centrally as a data source for a Web Server. In addition, usage of this technique reduces user perceived latency during browsing sessions, since less data must be transferred to users' personal computers.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Web page fragmentation; Web data manipulation; Information search; Web scraping; Web page transcoding

1. Introduction

Most web pages have a stable, static structure for the view (presentation) of their content. This structure rarely changes, even if the content of the web page changes very

*Corresponding author. Research Academic Computer Technology Institute, Riga Feraiou 61, GR 26221, Patras, Greece. Tel.: +30 2610 960375; fax: +30 2610 996314.
E-mail address: bouras@cti.gr (B. Christos).

1084-8045/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2006.01.005

often. The layout of web pages remains the same, regardless of the frequency of the content's change (for example, personal sites of Internet's users change rarely, while portals' content changes many times per day). In content-rich web sites (such as general interest or news portals) this structure comprises of areas of content (usually text with links) of common thematic areas (for instance, news about politics, economy, sports, etc.). These areas are called 'Web Page Components' or 'Web Page Fragments', because those web pages can be split entirely in such discrete areas.

Most users of the Web visit some specific web sites every time they are engaged in a browsing session. They usually show interest for some specific parts of the home page. For example, some users only visit the sports section, while others prefer to view news about politics and economy. Their browsing habits are almost the same every time they use the Web: they sequentially visit a list of preferred web sites (portals) and view specific areas of these sites.

In this paper, we present a technique that could assist the users of the web in their browsing sessions. In particular, the technique presented can be used for building a web service that allows its users to construct 'personal home pages' containing areas from their favorite sites. By using this service, a user could have in a single web page, all the areas of the sites he/she prefers. The presented technique premises the usage of a software tool that works centrally (as a data source for the web server), analyzes selected web pages and fragments them in the thematic areas they are composed of. Updated sections of the web pages are stored centrally and are used for the construction of the home pages of the users. Every web page that is to be available to the users for building their personalized pages must be first analyzed by the software tool, which extracts the page's structure and defines the areas (Web components (WC)) that compose the page. These WC are used for building the users' personalized pages. The functionality of this tool is transparent to the users, since they only use a web interface. Users select from this web interface those areas of the home page of the web portals they wish to include in their 'personal home page' and the system automatically constructs their own home page using the latest version of the content every time they wish to access it.

This technique offers several advantages to users. The most important is that they will not need to browse all of the web sites that have some information of their interest. All this content can be found gathered in their personal home page. In addition, the volume of the data that has to be downloaded to the users' personal computers is significantly reduced. Given that most users are usually interested in only few of the available areas of portals, all other areas (text and images) will not be downloaded to their PCs, since the technique is server-based. This fact reduces the user perceived latency and the cost of their browsing sessions. In addition, a modified extended version of this technique will allow users to view fragments of their favorite portals in small screen devices.

This paper starts with a presentation of related work in Section 2 and an introduction to the concept of WC in Section 3. In this section, a definition of the term 'WC' is provided. In Section 4, an algorithm is presented, which is used for fragmenting a web page and extracting the WC that compose it. In Section 5, the technique that has been implemented for constructing personalized pages based on popular portals' WC is presented. Section 6 examines the performance of the proposed technique in terms of avoided data transfers and also presents an evaluation of the training procedure. Finally, in Section 7, our future work plans are presented and some conclusions are drawn in Section 8.

2. Related work

‘WC’ was introduced as a concept in Bouras and Konidaris, (2001). In Bouras and Konidaris(2002, 2003) the concept of ‘WC’ is utilized for improving web performance and reducing redundant data transfers and user perceived latency. The fragmentation algorithm that is used in the system was presented in Bouras et al. (2004).

In Sugiura and Koseki (1997) a system named Internet Scrapbook is presented, which aims to automatically create personalized pages using fragments from other web sites. The users select a part of a web page and the system creates a matching pattern. To update a personal page, the system extracts the target data from source pages and re-constructs the personal page. In Internet Scrapbook, extracted data are determined by line patterns, which are the previous/first/next lines of the portion that the user had selected on an original source page. In addition to the line patterns, Scrapbook also uses HTML tag patterns, which are the order of the tags in a source page. However, the system uses the line patterns prior to the tag patterns. The tag patterns are only used when no candidate portion can be found with the line patterns or when multiple candidates exist.

In (Ramaswamy et al. (2005)) a technique to fragment a web page is presented which uses multiple instances of the same page and also multiple pages from the same web site. However, the fragmentation aims at maximizing cache efficiency, whereas ours is aimed at making a fragmentation that looks logical to the users.

The problem of providing specialized content to diverse devices like PDAs, mobile phones or though voice interfaces is closely related to the problem of fragmenting a web page. Web content providers can design device specific interfaces, but this solution is expensive. Alternatively, transcoding proxies can be used, in order to transform the web content as it exists on the server into a form capable of being displayed to each respective device, a technique with unsatisfactory results. Finally, specialized wrappers can be used, which can simulate human interaction with the web site and provide shortcuts to the information needed, excluding display of unwanted material. Wrappers can be used in a variety of ways. They can be provided by the ISP, by the content provider or they can be used by the user himself. They can also be used by enterprises to provide device-dependent views of services. The two main problems of the use of wrappers are scalability, which is the ability to provide access to a large number of web sites, and robustness, which is the wrapper’s ability to adjust to changes of the web pages on which it operates.

Wrappers are used by the system described in Freire et al., (2001). This system focuses on the problem of identifying a particular part of a web page in different time points, besides fragmenting a page and it is the most relevant to our own objectives among the systems we found in our research. It uses the XPath query language to specify extraction expressions for its flexibility and ease of use. In (Ramaswamy et al. (2005)) the WebVCR system is presented, which records a user’s interaction with the web site, so that the next time the user wants the same information, the interaction will automatically be performed by the system.

Several transcoding systems have been presented that aim to provide users of small-screen devices, such as PDAs or WAP-phones, an alternative, enhanced way of browsing the Web (see e.g. Bickmore and Schilit, 1997; Britton et al., 2001; Buyukkokten et al., 2001; Chen et al., 2001, 2003; Hwang et al., 2001, 2003; Kaasinen et al., 2000). An interesting feature of some of these systems is that they offer a way to create ‘summaries’ or ‘indexes’ of the full content of web pages. An annotation-based transcoding system (not fully automatic) is presented in Hori et al. (2000).

In Liu et al. (2004) a system is proposed, which aims to create logical structures of web pages using a software agent. To achieve this objective the WICCAP Data Model is defined, which maps a pages physical structure to logical views. A visual tool is also presented, which eases the construction of logical views.

A 'Web surfing assistant' is presented in Hwang and Lee (2003), which utilizes a similar fragmentation technique as the one presented in this paper for splitting a web page in semantic regions. Also, the work presented in Challenger et al. (2000) and Wills and Mikhailov (2000) investigates fragmentation's impact on Web performance. Several interesting heuristics for fragmenting a web page have been presented in Butler et al. (2001), Caverlee et al. (2004), Chen et al. (2001) and Yu et al. (2003) amongst others.

To conclude, we identified several research efforts similar to ours. Indeed, some heuristics that have been proposed in the past could also be used to extend our technique. However, almost none of them offered the functionality we wanted to implement in our technique. In particular, the systems we found in the bibliography do not offer a way to identify a portion of web page in different time points (with the exceptions of Freire et al., (2001) and Sugiura and Koseki (1997)).

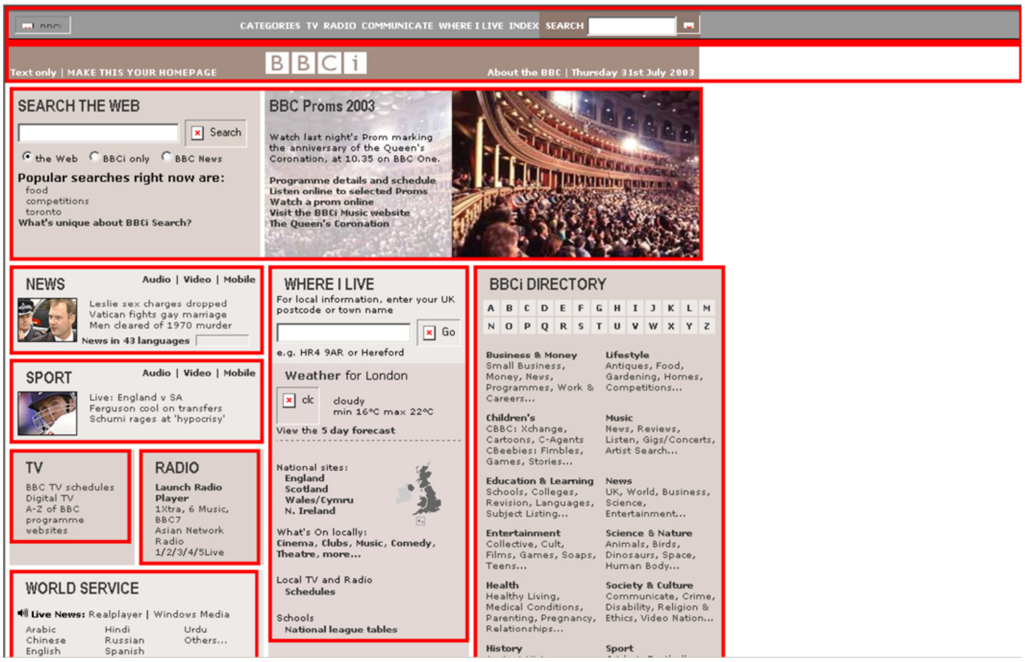
3. Web components

While authoring a simple web page is easy, authoring a portal or a content-rich web page is quite challenging. The problem lies in the fact that a content-rich web page contains information, navigation links, images, text, etc making it impossible to simply add content in a top-to-bottom approach (i.e., like writing a text file). There are various *layout* techniques and practices that can be used to make the presentation of content appearing in a web page more consistent to its semantic meaning and purpose. For example, one of the first techniques that was used in the WWW that differentiated a web page from a simple text file was the addition of a navigation bar in the header of the page or in the left side of the page. Today's popular sites use much more complex techniques and HTML code than this to build functional and appealing layouts for their content. In every portal (or content-rich web site in general) one can identify discrete content areas that contain text or links that belong to the same thematic area. For example, these areas could contain news about sports, economy, politics, etc. We call these areas 'WC' (or '*Web Fragments*'). Every web page can be fragmented in such areas. The number of the discrete WC depends on the complexity of the page layout and content differentiation. Usually, portals consist of 15–25 WC. You can see an example of this fragmentation of web pages in Fig. 1, where the first page of the BBC web site is presented (www.bbc.co.uk), with its WC marked with bold border.

4. Fragmentation algorithm

A browser renders a web page based on the HTML file that represents the page. This file is nothing more than a text file with markup tags that instruct the browser how to render the page. In addition, external files may exist (for example, files for CSS, javascript code, images, etc.). However, only the HTML file must be handled to extract the WC that compose a page.

The tags inside the HTML file are nested. This means that the code of the page can be represented as a tree (HTML tree). The root of this tree is always the HTML tag, which



CATEGORIES TV RADIO COMMUNICATE WHERE I LIVE INDEX SEARCH

Text only | MAKE THIS YOUR HOMEPAGE

BBCi

About the BBC | Thursday 31st July 2003

SEARCH THE WEB

Search

the Web BBCi only BBC News

Popular searches right now are:
food
competitions
toronto
What's unique about BBCi Search?

BBC Proms 2003

Watch last night's Prom marking the anniversary of the Queen's Coronation, at 10.35 on BBC One.

Programme details and schedule
Listen online to selected Proms
Watch a prom online
Visit the BBCi Music website
The Queen's Coronation

NEWS Audio | Video | Mobile

Leslie sex charges dropped
Vatican fights gay marriage
Men cleared of 1970 murder
News in 43 languages

SPORT Audio | Video | Mobile

Live: England v SA
Ferguson cool on transfers
Schumi rages at 'hypocrisy'

TV

BBC TV schedules
Digital TV
A-Z of BBC programme websites

RADIO

Launch Radio Player
1Xtra, 6 Music, BBC7
Asian Network Radio
1/2/3/4/5Live

WORLD SERVICE

Live News: Realplayer | Windows Media

Arabic	Hindi	Urdu
Chinese	Russian	Others...
English	Spanish	

WHERE I LIVE

For local information, enter your UK postcode or town name

Go

e.g. HR4 9AR or Hereford

Weather for London

clk cloudy
min 16°C max 22°C

View the 5 day forecast

National sites:
England
Scotland
Wales/Cymru
N. Ireland

What's On locally:
Cinema, Clubs, Music, Comedy, Theatre, more...

Local TV and Radio
Schedules

Schools
National league tables

BBCi DIRECTORY

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

Business & Money
Small Business, Money, News, Programmes, Work & Careers...

Lifestyle
Antiques, Food, Gardening, Homes, Competitions...

Children's
CBBC: Xchange, Cartoons, C-Agents
CBeebies: Fimbles, Games, Stories...

Music
News, Reviews, Listen, Gigs/Concerts, Artist Search...

Education & Learning
Schools, Colleges, Revision, Languages, Subject Listing...

News
UK, World, Business, Science, Entertainment...

Entertainment
Collective, Cult, Films, Games, Soaps, Teens...

Science & Nature
Animals, Birds, Dinosaurs, Space, Human Body...

Health
Healthy Living, Medical Conditions, Parenting, Pregnancy, Relationships...

Society & Culture
Communicate, Crime, Disability, Religion & Ethics, Video Nation...

History

Sport

Fig. 1. BBC home page fragmented.

has two children (HEAD and BODY). The building code of the Web page is inside the BODY tag. Therefore, we can extract the parts of the page that represent the different WC of the page just by extracting some particular nodes of the HTML tree.

The algorithm that fragments the web page must be able to determine which nodes of the HTML tree represent WC. This is not a straightforward procedure. The main difficulty is the 'subjective' nature of the procedure of identifying a WC. An area that one person comprehends as an area with semantically similar content could be considered by another person as too complex to be represented by a single WC. This problem could be solved by considering every node of the HTML tree as a WC. For instance, we could consider an image (contained in an IMG tag) or a paragraph of text (contained in a P tag) as separate, discrete WC. The user could combine some of these components in order to construct his own components. But this would make the service provided too complex and could also lead to huge problems when the site content changes and the software mechanism would have to recognize the new version (instance) of the selected components.

In order to overcome this problem, the implemented fragmentation algorithm was solely based on the structure that the web site's author uses to construct the web page. Most of the web sites use tables for building their layout. This led to the decision to use the table structure of a web page as the leading criterion for fragmenting the page in discrete components. If we ignore all the tags (nodes) of the HTML tree except the TABLE tags, the HTML tree is significantly reduced in complexity and depth. Some of the TABLE tags of this tree are only used for layout purposes, while some others contain content. Then, based on the amount of content (text) that corresponds to each node, the algorithm chooses which nodes must be considered as building components of the web page. Each node of this reduced tree contains a link to the corresponding node in the HTML tree. The algorithm uses the reduced tree (index tree) to fragment the web page and afterwards it can retrieve the actual content of each component by following the links to the HTML tree.

The fragmentation algorithm is used for the *analysis and fragmentation of the web page*, which includes two phases: training phase and update phase. Both in the training phase and the update phase, a software mechanism (which implements the fragmentation algorithm) downloads the web page, parses it and fragments it to WC. In the training phase, the fragmentation algorithm calculates the areas that will be candidate WC, while in the update phase the fragmentation algorithm updates the information that are stored about the presentation of the selected WC. If the fragmentation algorithm finds a change in the structure of the page or in the number of the WC during the update phase it tries to resolve the issue by recalculating the WC of the page.

The steps of the fragmentation algorithm are presented in the procedure below:

Algorithm 1: Fragmentation Algorithm

```
Note: Steps 1-4 are used both in the 'Training' and the 'Update' phase.
(1) Fetch the latest instance of the web page from its respective URL.
(2) Parse the web page and construct the HTML tree.
(3) Analyze the HTML tree and produce the index tree.
(4) Analyze the index tree and calculate which nodes must be marked as Web components.
Note: Steps 5 and 6 are used only in the 'Update' phase.
(5) Check if there are differences in the structure of the index tree from the index tree of
    the'training' phase or if there are differences in the number of the web components
    selected. In case there are differences, recalculate the web components.
(6) Extract the Web Components from the HTML tree and store them.
```

Step 1 is fairly simple. The fragmentation algorithm requests the html file of the web page from the respective URL and downloads it locally.

For the needs of step 2 we use a simple HTML parser. It takes as input the text of the HTML file that was downloaded from the page's web server (step 1) and builds the HTML tree. All the necessary information for reconstructing the HTML file in its initial form are stored in this structure. But the purpose of this transformation (flat text file to tree data structure) is to have the data of the web page in a form that allows the use of algorithms that work easily, effectively and efficiently. The fragmentation algorithm also starts a background procedure, which downloads locally all the other files that are used for the presentation of the web page (images, javascript files, CSS files, flash, etc.).

Step 3 of the fragmentation algorithm takes the HTML tree as input and constructs another tree structure that is used in step 4 for recognizing which areas of the HTML file (or nodes of the HTML tree) will be extracted as WC. This tree is also used as an index for the HTML tree. Therefore, throughout this paper it is referred to as the 'index tree'. It is significantly smaller in size than the HTML tree since it only contains the TABLE nodes.

The algorithm starts from the root of the HTML tree and recursively traverses all of it. For every TABLE node it finds, a new node is added to the newly constructed index tree. This way the produced index tree has the structure the HTML tree would have if all nodes except TABLE nodes were removed. An ID is assigned to each node of the index tree, which depends on the position of the node in the tree. In particular, the ID is a representation of the path from the root of the index tree to this node. All nodes except from the root have a number indicating their position relatively to the other nodes with the same parent. Starting from the root, following the nodes on the path to a node and combining their numbers an ID for each node is constructed. For instance, the ID of the third child of the second child of the root is '2-3'. Each node of this new structure has a link to its corresponding node in the HTML tree and also some information that will be of later use. This information includes the length of the text of this node in the HTML file (with and without the tags), the ID of the node, the number of images that are included under this node and finally the number of links that can be found in the text (content) of the node.

The actual decisions about how a web page will be fragmented are made in step 4. The fragmentation algorithm uses the index tree that was produced in the previous step of the fragmentation procedure. It starts by recursively parsing the index tree trying to find nodes that meet some particular criteria. When a node meeting those criteria is found, no children of this node are visited and the node is marked as a WC. This means that the whole sub-tree beneath this node is considered as a single entity that can be used by the users of the service for building their personalized page. The children of this node are part of the component and cannot be used standalone, since the algorithm has decided that their content is of minor importance (or too small) in order to be used as a component. We have to note here that a node in the index tree that has been selected as a WC cannot be used directly to get the actual content of the component. In order to achieve this we have to follow a link from the node in the index tree to its respective node in the HTML tree and from there acquire the HTML code of the component in text format.

The criteria used to decide if a node of the index tree (i.e. a part of the HTML file) is suitable for being used as a WC are related to the size of the content of this node and its internal structure (i.e. the number of children and descendants of this node). In its current form, the algorithm calculates the 'size of the content' of a node by calculating the length of the *pure text* (i.e. without the tags) that is found inside the node (future plans include to use the area occupied by the component in the web page instead of the length of the text). If node p meets the following criterion then it is marked as a WC without even examining its internal structure:

$$\text{Average Ratio} \leqslant \text{Ratio}_{\text{p}} \leqslant 2 * \text{Average Ratio} \tag{1}$$

or

$$1 \leqslant \text{Ratio}_p * (\text{Number of content nodes}) \leqslant 2, \tag{2}$$

Where

$$\text{Ratio}_p = \frac{\text{Pure text length in the node } p}{\text{Pure text length of the root node}} \tag{3}$$

and

$$\text{Average Ratio} = \frac{1}{\text{Number of content nodes}} \tag{4}$$

Relation (1) (or its equivalent relation (2)) expresses the intuitive criterion that a WC must be 'medium'-sized, neither too small, neither too large in comparison with the whole page size. Ratiop is calculated by dividing the pure text length included in node p by the pure text length of the entire page, giving the percentage of the node's content to the content of the whole page. By the term 'pure text' we denote the text without the HTML markup. This metric expresses the relative size of the WC (regarding the size of the whole page). Average Ratio is the percentage of the text of the whole page that a node would have if all nodes that contain content (i.e. nodes that contain text and are not used for layout purposes) were equally sized. This metric is used to approximate the size of a 'medium-sized' component. It is defined as the inverse of the number of the content nodes of the index tree. If the size of the content (text) of a node is greater than the average size (i.e. the Average Ratio) or smaller than the double of the average size, then that node is considered 'medium-sized' and is selected as a WC.

Relation (2) could be rewritten in a more abstract form as

$$l \leqslant \text{Ratio}_p * (\text{Number of content nodes}) \leqslant u, \tag{5}$$

where

$$0 \leqslant l \leqslant u \leqslant u_{\max} (= \text{Number of content nodes}).$$

The values of l and u express the lower and upper bound for the length of a node's text in order to consider the node as 'medium-sized'. Relation 5 means that if a node's text length is greater than or equal to $l/u_{\max}$ and smaller than or equal to $u/u_{\max}$, then this node is considered 'medium-sized' and is selected as a WC. By substituting $l = 1$ and $u = 2$ in (5) we get the criterion expressed in (2). The values of $l = 1$ and $u = 2$ were arbitrarily chosen, since they resulted in good fragmentation of web pages. In case we had set a value for l that was smaller than 1, then the algorithm would select nodes with text length smaller than the text length of the average node, which is already small. We chose $u = 2$ after experimenting with several web sites and examining the fragmentation's results. However, future work plans include further testing with more web sites in order to find the 'ideal' values for the constants l and u.

The size of the content contained inside a TABLE tag is not the only criterion used for fragmenting a web page (although it is the most important). The other major criterion that is used for selecting a node as a WC is based on the structure of the index tree. We noticed that areas that are intuitively perceived as WC are frequently composed of more then one TABLE tags. From these TABLE tags, one contains the main body of the Component's content, while the rest are used for layout purposes or contain an insignificant amount of content (for instance, a title of an article). Thus, when the fragmentation algorithm finds one node of the index tree, which is not a leaf, and which contains *less than four children and less than five (in total) descendants* (not including layout nodes) it selects this node as a WC. The WC that have either been selected because of their content size or because of their structure and contain more than one table tags (i.e. they are not leaf nodes in the index tree) are called '*complex* WC'.

When the fragmentation algorithm finishes the traversal of the index tree, it makes some last refinements of the WC selections. More specifically, if it finds a WC that is the single child of its father it selects the father as a WC. This is done because the father of the previously selected component is probably a layout table tag or contains some content related to its child node (such as a title or the author of an article).

When step 4 of the fragmentation algorithm finishes, the whole index tree has been traversed and some nodes have been marked as WC. If the algorithm is in the update phase, two more steps, which are described in the respective section of the paper, are performed. For the training phase, an index tree with the selected WC suffices.

5. Transcoding technique

Next we will present the methodology for constructing personalized web pages based on WC. There are three phases: *Web pages' analysis and fragmentation*, *components selection (by the user) and personalized page synthesis for presentation to the user*.

6. Web pages' analysis and fragmentation

The first phase involves a software tool whose role is to continuously download and analyze selected web pages and update the HTML code of each component and the meta-information stored about it. This tool ('WCs Creator') is not installed on the users' personal computers, but functions centrally, as a data source for the web server of the service provider. It keeps a list of web pages defined by the service administrator. The WC that are available for the creation of personalized pages are extracted from these sites. These WC are not created dynamically upon users' requests, but are extracted and stored by this software mechanism. The building code of each WC is updated in frequent time intervals. The reason behind this choice is *performance*: In the case that we had selected the dynamic creation of WC upon users' requests, the user should have to wait for a significant amount of time, since all the web pages that constitute in the creation of his/her personalized page would have to be downloaded on the server of this service, they would have to be parsed and analyzed and after that, the personalized page of the user would be created by the extracted components. This would also lead to huge waste of bandwidth (every page that has a component in a user's personalized page is downloaded in every request) and server resources. Instead, by having the 'WCs Creator' continuously analyze the web pages that are contained in its list of interesting sites, only the last step of the procedure described above (creation of a web page from the HTML source of the selected components) is performed upon a user's request.

Web pages' *analysis and fragmentation* is a multi-step procedure. This procedure includes two main phases. The first one is the *training phase*, where each web page is examined for a given period of time and areas that can be treated as WC are detected. In this procedure, the training algorithm parses the web page many times, fragments it and stores some data after each parsing. Later, when enough data have been gathered, the algorithm analyzes them and calculates which areas of the page will be extracted as WC. The training algorithm stores information about each WC that will be used as the *identifier* of this component in the update phase, which follows the training phase.

When the training procedure of the system for a specific web page has been completed, the *update* procedure begins. This procedure lasts for as long as this specific web page is chosen to be available to the users. The update procedure fragments the web page and based on the information collected from the training phase, updates the latest instances of the WC that are stored in the system.

In the following sections, the training and the update phases are examined more thoroughly.

7. Training phase

The training phase of a web page analysis must be performed before this page's WC are made available to the users. The WC of this specific page can be used for building personalized pages, after the training phase has been successfully completed. Its role is to analyze the web page and decide how the Web page will be fragmented and how its WC will be selected during the update phase. In the end of the training phase for a specific web page, the training algorithm's output is the number of the WC of this page, and a *unique identifier* for each one of these components. This unique identifier can be used for identifying a WC in a web page instance that has changes in the page structure or changes in the number of the WC. This training phase would not be required if changes never happened in the web page's structure and the relative size of its content areas. Although changes of these kinds are rare, it is almost sure that they will happen. The training phase allows the system to have pre-built knowledge about how to handle these changes. In the training phase we try to find a unique identifier for each WC. During this phase we use the ID of each WC as an identifier, which depends on the position of the WC inside the index tree.

Although the goal of the training phase is to allow the creation of a mechanism that could overcome problems caused by changes of the structure of web pages or in the number of the WC, one basic assumption for its correct functioning is that changes like these *do not happen during the training procedure*. If changes like the ones described above happen during the training procedure, the algorithm must become too complicated in order to overcome them. There are various ways for the solution of this problem. The most simple is to reject the samples that present changes (this is done in the current implementation). It is also possible to use both the structure and the content of the WC in order to readjust the structure of the index trees, but this is a much more complicated procedure.

A WC can be characterized by many factors: Its position inside the index-tree, its relative position to the other WC, its ID inside the index tree, its content (in terms of text or images), its content size (in terms of text length or number of images) and others. However, it is quite difficult to find a criterion that can be used to *uniquely* identify a WC from the others that are contained inside a Web page. We have to note here that this is necessary for the proper functioning of the system, since users must be able to select which components they wish to see in their personalized page and the system must be able to recognize them from the list of the WC extracted from the fragmentation algorithm. The training phase, which lasts from 1 day to several days, aims at providing a unique identifier for every WC of a Web page.

Before continuing with the analysis of the training phase, one important fact must be mentioned about the content of the WC. Depending on the web page we can classify WC in three categories, using the criterion of the *changes of their content*: There are some components whose content remains constant (for example, an area with links to categories of news), there are some others whose entire content changes (for example, an area with news headlines) and there is a third category of components where part of their content changes, while the rest remains the same (for example, an area with news headlines that has a 'NEWS HEADLINES' title on top). During training phase the constant part of the components' content is used for components of the first and third categories in order to assign a unique identifier for them, while the relative position of the components is used for components of the second category.

The training phase for a web page can be split in four sub-phases:

(1) Data gathering phase.
(2) Comparison of *Content Vectors* (CVs) of instances of the same WC and extraction of a single Constant *Content Vector* (CCV) for each WC.
(3) Comparison of the Constant Content Vectors of all the WC of the web page and extraction of the Identifier *Content Vector* (ICV) of each WC. The ICV of a WC uniquely identifies it between all other WCs in the web page, with the exception of *weak* ICVs of WC that are empty.
(4) Assignment of a *signature* for each WC of the page.

In the data gathering phase, the training algorithm uses the fragmentation algorithm in order to gather information about the usual structure of the web page. In fixed time intervals the fragmentation algorithm is activated and the index tree for the specific page instance in that point of time is stored. The goal is to have enough specimens of the index tree for a time interval in which all the content changes that happen regularly in the web page have taken place. In the most popular news web sites or portals this time period is 24 h, while for sites that change less frequently this time interval can last more (it depends on the frequency of the content's change).

When this predefined monitoring time period has passed, k specimens of the index tree have been collected, where $k = \lfloor \text{Monitoring period/sampling interval} \rfloor$. At this point the training algorithm has acquired all the data it needs for its analysis.

In the second phase of the training procedure, the algorithm computes which part of each WC stays *constant* during the monitoring period. For each WC of the index trees, the fragmentation algorithm constructs a data structure that contains its content, i.e. every word of the text inside the WC and the filenames of the image files contained in the component. This data structure is named 'CV' and is a characteristic of each WC *instance* (this means that the CV can be different for different instances of the same WC). The representation of documents as *term vector*s is a very popular technique in information retrieval. The CV is a pair of two vectors, one containing the terms inside each WC instance and one containing the filenames of the image files inside the components. These are symbolized as T_p and I_p, respectively:

$$T_p = \{w|w \text{ is a word inside the pure text of WC } p\}, \tag{6}$$

$$I_p = \{\mathrm{z}|\mathrm{z} \text{ is an image contained in the code of WC } p\}, \tag{7}$$

$$\mathrm{CV}_p = (T_p, I_p). \tag{8}$$

We assume that for the k specimens of the index trees the number of the WC that have been selected in each fragmentation and the index tree's structure remain the same. Using the ID of each WC, the training algorithm acquires this WC's instances and its CVs from the collection of the index trees. Next, it compares the k CVs of each WC and keeps only the content (text and image filenames) that exist in ALL the CV. At the end of this procedure the algorithm has constructed a data structure that keeps the content of each WC that remained constant during the whole training procedure. This structure is named 'CCV' and is a characteristic of a WC independent of its instances in different time points. Eq. (9) expresses the construction of CCV in mathematical terms.

$$\mathrm{CCV}_p = \bigcap_{t=1}^{k} \mathrm{CV}_{p,t}, \tag{9}$$

where $\mathrm{CV}_{p,t}$ is the Content Vector of the tth instance of WC p.

Although step 2 of the training procedure produces a structure that could identify a WC with great accuracy, there are some cases where the CCV of a WC is not enough. The CCV of WCs is constructed considering only the content of this specific WC (in all instances of this WC derived during the training period). But the goal of the training procedure is to produce a unique identifier for all WCs of a Web page. Therefore, in step 3 of the training procedure the CCVs of all the WCs are compared mutually. There are two modifications that are made in the content of the CCVs: The first is that the text or images that exist in all the CCVs are removed. The second action is that if the content of a CCV is contained completely inside the content of another CCV, then the first WC and its ICV (which is the output of step 3) are marked as *weak*. This means that its ICV (reduced CCV after the removal of all the common content elements) cannot uniquely identify it and that its relative position in the index tree should be used for identifying it. In the end of step 3 of the training procedure, each WC has a reduced CCV that uniquely identifies it in the Web Page, with the exception of WCs that are marked as weak or that have CCVs that are empty. These WCs get a unique identifier in step 4 of the training procedure. The reduced CCVs that uniquely identify WCs after the completion of the 3rd step of the training procedure are named 'ICVs' of their respective WCs. This procedure is shown in Eq. (10).

$$\mathrm{ICV}_p = \mathrm{CCV}_p - \bigcap_{i=1}^{p_{\max}} \mathrm{CCV}_i, \tag{10}$$

$$\mathrm{WC}_p \text{ is weak if } \exists\ \mathrm{WC}_q:\ \mathrm{ICV}_p \subseteq \mathrm{ICV}_q. \tag{11}$$

Step 4 is the final phase of the training procedure. Until this point, almost all the WCs that are contained inside a Web page have been assigned an ICV that can be used as a signature for them. Step 4 assigns this ICV as the signature of the WCs that have one. In almost all the web pages, the vast majority of the WCs have an ICV that uniquely identifies it in all page instances. This ICV is set as the *signature* of the WC. However, it is possible that some components have an empty or weak ICV (an empty ICV is a also weak, since it is a subset of all other ICVs). As explained before, this ICV cannot be used as a signature. Therefore, in step 4 the training algorithm detects the WCs with weak or empty ICVs and

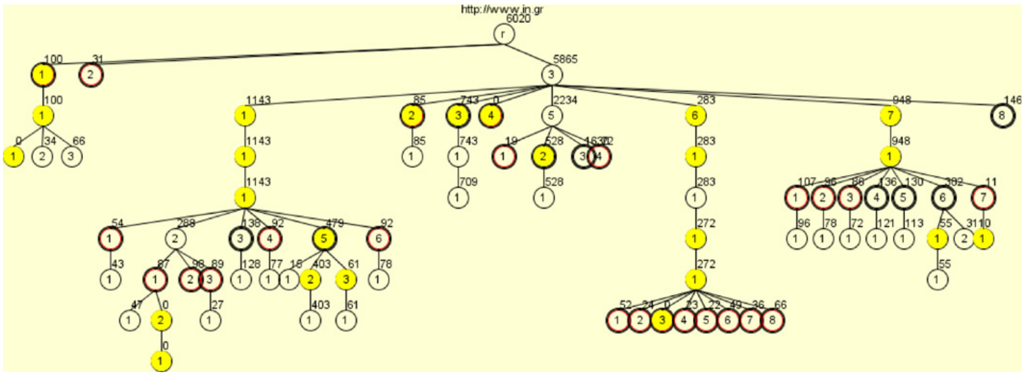


Fig. 2. Index tree for www.in.gr.

Previous Component	Next Component	Position	Text Size	Number of Images

Fig. 3. Identifier for components with no constant content.

assigns another kind of data structure, which is based in their relative position in the web page and in the content size, as a signature for them. The way this signature is constructed is explained with an example:

Fig. 2 shows the index tree graph of www.in.gr, a popular Greek portal. While almost all of its WCs (which have been marked with bold in the index graph) have an ICV as a signature, the leaf nodes, which are descendants of node with the ID 3-6, have been chosen as WCs and all of them have empty ICVs. Therefore, the algorithm must assign a separate signature to every one of them. The WCs with IDs 3-5-4 and 3-7-1-1 are the first components before and after the series of the components with empty ICVs. These two components are included in the signature data structure. In addition, the position of each one of these components with empty ICVs inside their list is stored in the signature data structure. These three fields denote the relative position of the components regarding the other components in the index tree. Additionally, two more fields are used to store the text size and the number of images included in each component. These are used in order to identify a component in cases where the relative position is not enough (However, there are cases where even these two additional fields are not enough. These are extreme and complicated cases and we will make no further reference to them in this paper). Therefore, the identifier data structure for components with no constant content or constant content that cannot be used as identifier (empty or weak ICVs) has the structure shown in Fig. 3. In this identifier the previous and the next component (1st and 2nd field) are marked with their *sequence number* in the *page index*, which is the final output of the training procedure. The page index is a matrix, which contains the ID and the signature of every WC of the page.

8. Update phase

Following the training phase, an estimation can be made about what output to expect from a later download and fragmentation of the web page. The full structure of the web page is available, shown in any of the k index trees collected during the training phase (note: the entirety of this information is not needed and only the last index tree is kept), and also the page index which stores all the WC identifiers. The page index contains both the *ID* of each WC, which is related to its position in the index tree, and its *signature*, which diversifies it from all other WCs (as has been explained in the previous section, the ID of a component is possible to change if there are changes in the page structure, while the signature remains constant).

The role of the update phase is to update the stored data with the latest instances of the WCs (HTML code, images, etc.). The time interval between each fetching and update of a web page depends on how often its content changes. This time interval is fixed, but in future versions it will vary depending on how often the algorithm finds the page modified.

The whole procedure of the update phase has many similarities with the training phase. It continuously fetches the web page, parses it and calculates (using the fragmentation algorithm) the WCs of the web page. Following this, it stores the latest instances of the WCs in the Web Server of the system, in order to be used by the users for their personalized portal's creation.

The fragmentation algorithm, as discussed in the respective section, produces in step 4 the index tree of the web page instance that was fetched and marks some nodes as WCs. The training procedure uses a collection of the index trees of different instances of the same page, in order to decide how the web page must be fragmented. We assumed that during the training phase no changes happen in the page structure or in the number of the WCs that was calculated by the fragmentation. In general though, despite being a rare situation, changes may come about during the update phase. In this case the fragmentation algorithm has to by pass the problems caused by the changes using the data gathered in the training phase.

Step 5 of the fragmentation algorithm is used for this purpose (steps 5 and 6 are not used in the training phase). It takes the output of step 4, i.e. the index tree and the page index and checks if there are differences in the structure of the page or in the number of the calculated WCs. Although it is possible to implement this check directly on the index trees, it is done by checking for differences in the ID field between the page index of the latest page instance and the page index that was produced in the training procedure (the page index of page instances contains the WC's CV in the placeholder of the signature, since the signature is a characteristic of all instances of a WC and a CV is a characteristic of each single instance). If no changes show up (which is the rule for most cases) then the fragmentation algorithm continues with step 6. Otherwise, a special procedure takes place, which aims at fixing the problems created by the differences.

The algorithm, which corrects the calculation of the WCs, is presented in the following paragraphs. It is the most complex algorithm implemented for this system, since many situations may trigger it. Although it is probably impossible to create an algorithm that could function flawlessly even at the most complex situations, the algorithm presented below shows a very good behavior for most common cases.

Algorithm 2: Fragmentation Correction Algorithm

```
(1) If (page index of the page latest instance page index) {
        (2) If (WC_count in the page index from training== WCcount in the instance page index){
                Check the signature scontained in the page index with the Content Vectors contained
                in the instance page index
                 (3) If (signatures match) {
                          Extract (mark for extraction) the Web components based on their signatures
                }
                 (3) else {
                          Extract all the Web components that their CVs match with signatures in the
                          page index. Extract all the rest WCs based on their order of appearance in
                          the page index.
                } (3)
        (2) } else {
                If (index tree structure from training matches with the instance index tree) {
                Extract Web components based on their IDs
       } (2)
(1) } else {
        Counter++;

         (4) If (Counter<4){
                 (5) If (WC_count in the instance > WCcount from training){
                          Run the fragmentation algorithm with its parameters set to produce larger
                          (and less) Web Components
                }else{
                          Run the fragmentation algorithm with its parameters set to produce smaller
                          (and more) Web Components
                }(5)
         (4) } else {
                Get the initial fragmentation (with the default value of the uparameter). Extract all
                the Web Components that can be extracted based on their Content Vectors. Extract
                all the remaining Web Components based on their order of appearance and their
                content size (closest match).
        }(4)
}(1)
```

We have to note here that the algorithm presented above uses the Content Vectors of the WC instances for the comparisons with the signatures of the selected WCs from the training phase. However, it gets more complicated if the page contains components with no constant content or weak ICVs. We assigned a different kind of signature for these components during the training phase (Fig. 3). The Content Vector of a WC instance cannot be used for comparing it with a Component with such a signature. Whenever such a situation occurs, the algorithm compares all the components that have an ICV as a signature and after this comparison has been done it tries to calculate the rest based on their relative position and their content size.

Another issue is how to make the comparison between the CV of WCs instances and the signatures (ICVs) of WCs contained in the page index. Remember that the ICV is subset of the CCV of a WC, which is calculated by comparing several instances of the same WC and removing the content that does not exist in all the instances. It also uniquely identifies a WC, i.e. there are no two WCs (WC) with CV that are supersets of the same ICV. The fragmentation correction algorithm uses this property of the ICVs for the comparisons it has to make. Specifically, it checks an ICV against all the CVs of the WC instances. The first matching instance, is the WC it tries to detect. There is one extreme case where this check can produce more than one results. It occurs if in an instance of a WC, the content that is changed contains all the content of the ICV of another component. However, this situation is extremely rare and no special measures to deal with it are considered.

When the fragmentation correction algorithm finishes (step 5 of the fragmentation algorithm) all WC instances have been marked for extraction. Then, in step 6 they are extracted and materialized in the Web Server. The index tree is traversed and for each marked node the algorithm follows the link to the respective node in the HTML tree and retrieves the HTML code of the WC. Afterwards, it makes some transformations to the HTML code and stores the derived code in the Web Server. This code is used for presenting the WC to the users. Each WC is identified in the Web Server by its sequence number in the page index of its respective page and the web page's name.

9. Personalized Portal Creation

Web page analysis and fragmentation aims at always having the latest instances of the WCs that comprise the web pages that are offered to the user for his/her personalized page creation. The software that implements the fragmentation technique stores the HTML code, the images, the javascript files and all the other related files in the hard disk of the web server that provides the service to the users. It also stores a modified version of each fragmented web page that is used for showing to the users the WCs that comprise it.

The next phase is interaction with the user. It aims at creating a list of the WCs that a user wants to include in his/her personalized page or altering this list by adding or removing components. Using a special web interface to interact with the system, the user is asked to select one of the sites that have been analyzed by the system. When the user makes a choice, he is transferred to a page where all the WCs of the selected site are shown in their initial position inside this site's first page, marked with a red border. When the user hovers his/her mouse pointer over the area occupied by a component, a button appears over this component asking the user to store this WC in his/her list. If the user selects to store the WC, then this component is recorded in his/her profile as a new entry in the selected components list. The user can now continue selecting other components. The components that have already been selected are marked with a yellow border. When the mouse pointer is hovering over them a button appears calling the user to remove them from the selected components list. When the user finishes with the selected web page, he/she can be transferred back in the first page where he/she is asked again to select one of the available sites.

The outcome of the described procedure is the creation of a list of the desired WCs for a user in the system database. This list is used when a user accesses his/her personalized web page.

10. Personalized Page Synthesis

Personalized page synthesis is executed by a script in the web server of the service provider. This script checks the database for the user's record and retrieves the list of the selected WCs. It then retrieves the source code of each selected WC from the filesystem of the web server and uses it for constructing the user's personalized page.

The first time a user enters his/her personalized page, all WCs are presented sequentially, top to bottom inside the page. However, the WCs are placed inside layers, which can be moved by the user using his/her mouse (there is a tab in the upper left corner of each layer for this purpose). Each WC's position is stored in the database when the user releases the mouse button. The next time the user enters his/her personalized page the WCs are placed in the same position that they were left. A special button that appears in the top (header) of the page along with the user's name, allows him/her to enter the 'positioning mode' again.

We have to note here that during the personalized page synthesis a special procedure must be followed for WCs that originate from pages using CSS. In this procedure, the CSS files of the originating pages are appended in the CSS file of the personalized page. When the page synthesis finishes, the CSS file that is created contains all the styles that are defined inside the CSS files of the originating pages of the components. In addition, during the extraction of the WCs in the update phase of the web page analysis and fragmentation, a transformation is performed in the source code of the WCs and the CSS files. In particular, all the styles are named based on the page they originate from. A style name 'X' which is used in the 'Y' page is named in the final CSS file of the personalized page as: 'Y_X'. This way no conflicts occur between styles with the same name that come from different pages. A similar procedure is performed for javascript files.

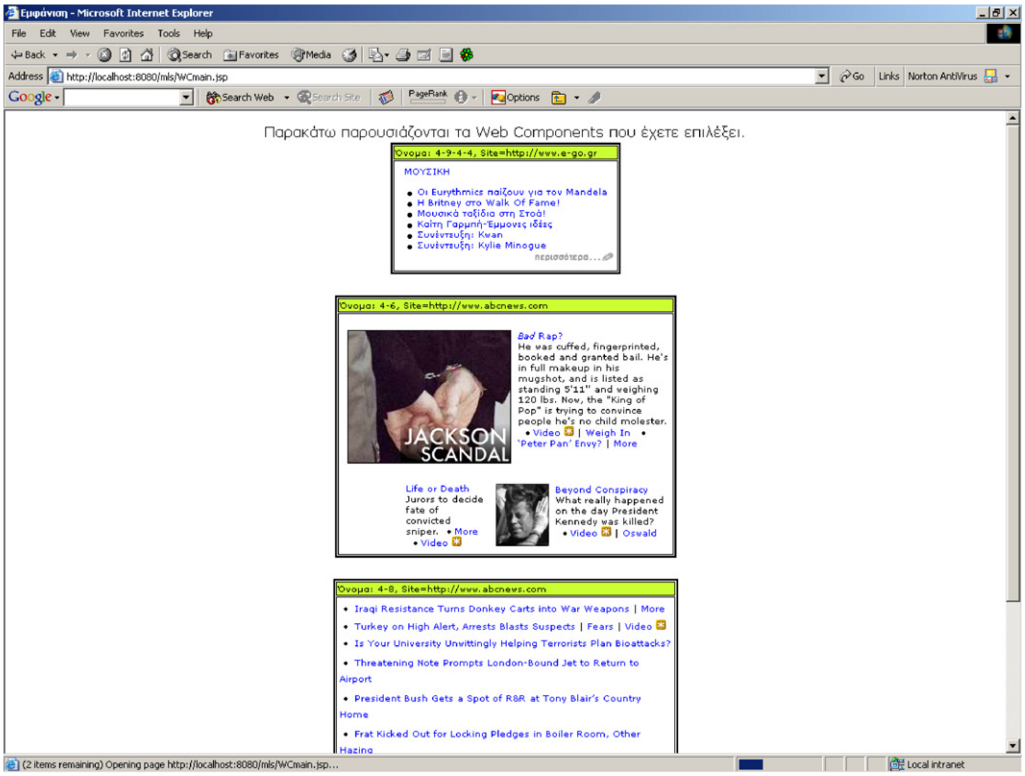
An example of a personal page is seen in Fig. 4. In this page there are three WCs selected, one from www.e-go.gr and two from www.abcnews.com.

11. Evaluation

In order to evaluate the training/update procedure we executed the respective algorithms with three news web sites. These web sites were: www.cnn.com, www.abcnews.com and www.cbsnews.com. The time interval between each parsing and analysis of the pages was 50 min. We allowed 10 parsings of each web page (we have to note here that for real use the number of parsings should be more and the monitoring period should be longer). The results are shown in Figs. 5 and 6, from which some interesting conclusions can be drawn:

It is clear that the assumption that no changes happen in the number of the WCs during the training procedure does not always hold. This can be seen in the graphs (in Fig. 5) for CBS News and CNN. However, we can see that there are only two and one fetches, respectively (out of 10) that differ from the majority. Therefore, rejecting these samples does not cause significant problems and loss of information.

Fig. 6 shows the size of the ICVs of the 25 WCs of ABC News. Nine WCs out of the 25 have weak ICVs. Five out of nine WCs with weak ICVs are also empty. However, almost all of them are weak not due to changes in their content, but due to small size. This shows


Εμφάνιση - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/mls/WCmain.jsp
Go Links Norton AntiVirus
Google Search Web Search Site PageRank Options
(2 items remaining) Opening page http://localhost:8080/mls/WCmain.jsp...
Local intranet
JACKSON SCANDAL

Παρακάτω παρουσιάζονται τα Web Components που έχετε επιλέξει.

Όνομα: 4-9-4-4, Site=http://www.e-go.gr

ΜΟΥΣΙΚΗ

- Οι Eurythmics παίζουν για τον Mandela
- Η Britney στο Walk Of Fame!
- Μουσικά ταξίδια στη Στοά!
- Καίτη Γαρμπή-Έμμονες ιδέες
- Συνέντευξη: Kwan
- Συνέντευξη: Kylie Minogue

περισσότερα...

Όνομα: 4-6, Site=http://www.abcnews.com

Bad Rap?
He was cuffed, fingerprinted, booked and granted bail. He's in full makeup in his mugshot, and is listed as standing 5'11" and weighing 120 lbs. Now, the "King of Pop" is trying to convince people he's no child molester.
• Video | Weigh In • 'Peter Pan' Envy? | More

Life or Death
Jurors to decide fate of convicted sniper. • More • Video

Beyond Conspiracy
What really happened on the day President Kennedy was killed?
• Video | Oswald

Όνομα: 4-8, Site=http://www.abcnews.com

- Iraqi Resistance Turns Donkey Carts into War Weapons | More
- Turkey on High Alert, Arrests Blasts Suspects | Fears | Video
- Is Your University Unwittingly Helping Terrorists Plan Bioattacks?
- Threatening Note Prompts London-Bound Jet to Return to Airport
- President Bush Gets a Spot of R&R at Tony Blair's Country Home
- Frat Kicked Out for Locking Pledges in Boiler Room, Other Hazing

Fig. 4. Personal page.

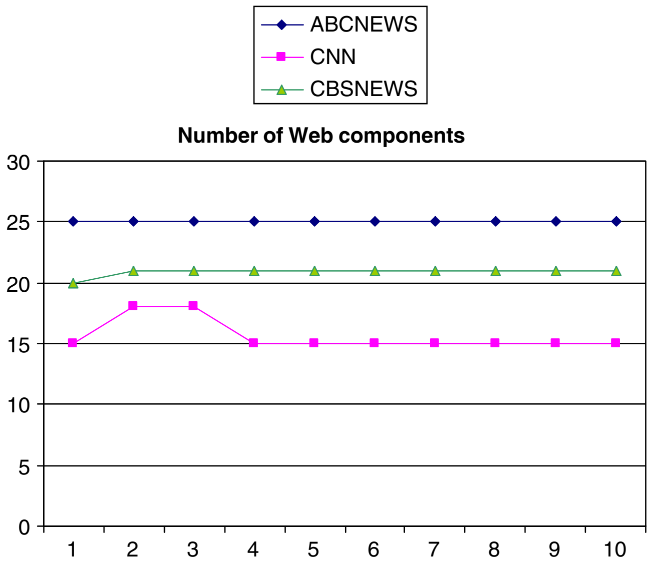


Fig. 5. Number of Web Components for 10 parsings.

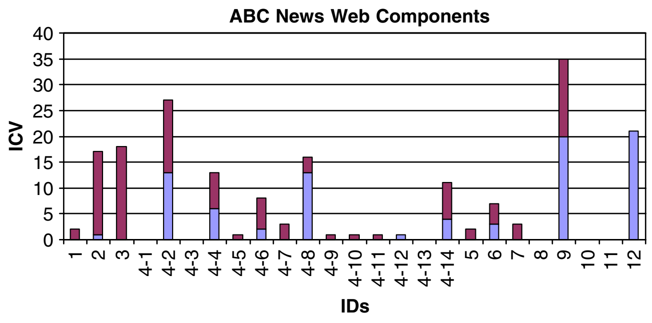


Fig. 6. Web Components of ABCNEWS.

that the whole technique could be enhanced (and simplified) by not allowing the fragmentation algorithm to select small WCs (it could merge them with others or just ignore them). Fig. 6 also shows that some sites (such as ABCNEWS) utilize a lot of images for building their layout and these images contribute a lot to the content of the WCs. For instance, Components with IDs 2 and 3 are almost completely composed of images. This leads to the conclusion that images should also be considered in the heuristics of the fragmentation algorithm.

Although the major goal of the technique described in this paper is to assist the Web users in their browsing sessions by collecting in a single web page all the WCs that they wish to see, it also has some performance advantages. Specifically, by using this technique the user's browser does not have to download all the content that comprise a web page (HTML code, images, etc), but only data belonging to the specific WCs that are used in the user's personal page. These data consists of HTML code, images, flash files and all other files that are used for presenting the WC in its originating page. Huge savings in downloaded data can be achieved by the proposed technique if users only show interest in small parts of web pages. Downloading less data also results in savings of the available bandwidth for other user activities and leads to the reduction of user perceived latency.

In order to demonstrate the amount of avoided data transfers to the users' personal computers by using this technique, some experiments were performed. Three popular sites were selected (CNN, BBC and Yahoo) and the fragmentation technique was applied to them. They were split in their respective WCs and the size of each component was recorded. The size of the components was calculated by adding its HTML code size to the size of the images included in it. Assuming that a user selects some WCs from each site and rejects all others, only the data of these components will be transferred to his/her personal computer. The rest of the data are not transferred since the fragmentation technique is executed centrally as a data source to the web server and the personalized page of the user is constructed in the web server of the service provider. For instance, if a site contains a WC with advertisements (which are usually big files relatively to the other files of a web page) and has not selected it for his/her personalized page, then all these data will not be downloaded to his/her personal computer.

The results of the fragmentation for the web sites of CNN, BBC and Yahoo are shown in Fig. 7. It shows the sizes of the WCs that comprise each page.

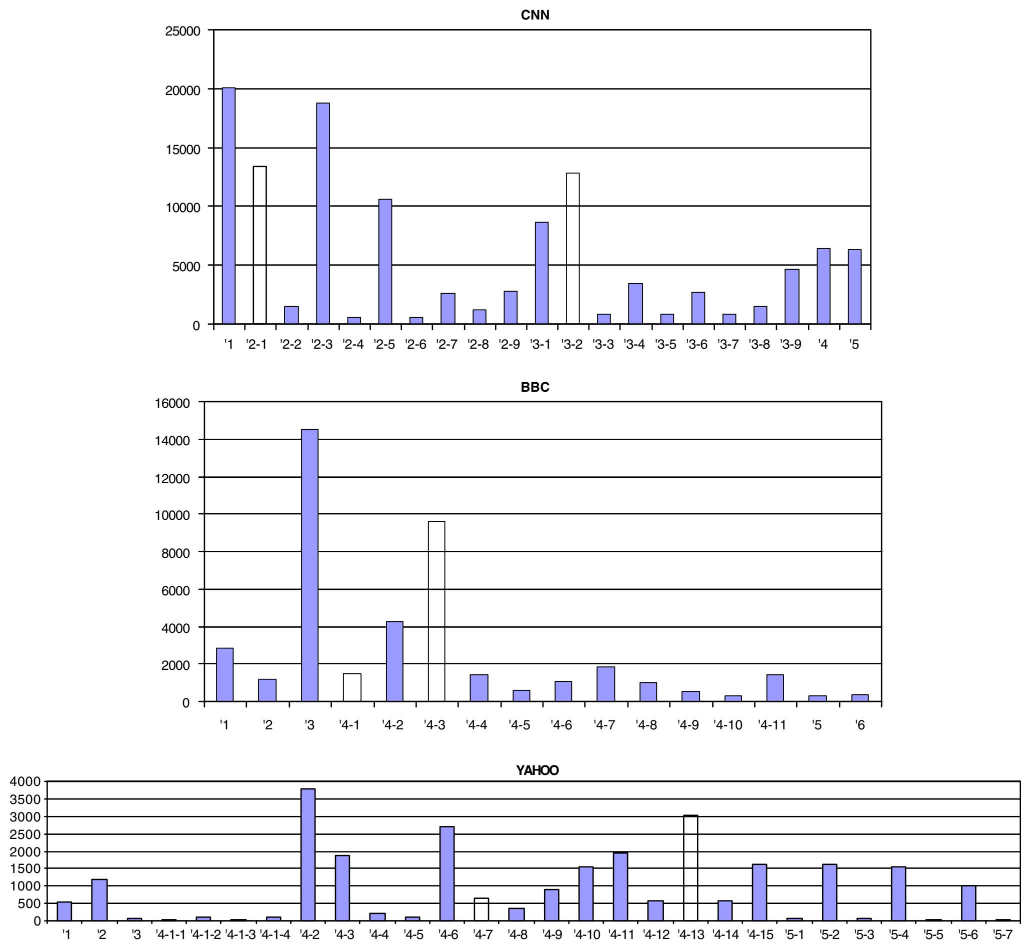


Fig. 7. Web fragmentation of three popular sites.

The percentage of downloaded data (D) and avoided data (A) over the whole page data size, which denote the performance gain from the technique, can be calculated by the formulas: $\mathrm{D} = \sum \mathrm{S}_{p,k} / \sum \mathrm{Total}_k$ and$A = 1 - \sum \mathrm{S}_{p,k} / \sum \mathrm{Total}_k$, where $S_{p,k}$ denotes the size of the pth component of the kth page and Total_k denotes the size of the whole page.

As an example of the performance gain of using the fragmentation technique, let's assume that a user selects to see in his/her personalized page only the news headlines and the 'general' links from the three sites presented above. These are included in the following Components: 2-1 and 3-2 for CNN, 4-1 and 4-3 for BBC and 4-7 and 4-13 for Yahoo (they are marked with different color in Fig. 7). Substituting the respective values in the formulas presented above we obtain a 78% gain for the user.

In conclusion, the result of these observations is that the fragmentation technique besides the convenience of presenting to the users all desired information in a single page, can also help towards the reduction of data transfers to their PCs and increase the perceived 'speed' of the Internet connection during browsing sessions.

12. Future work—enhancements

The technique presented in this paper can be further improved in many ways. Future work plans include testing the software that implements this technique with some users and gathering their feedback on bugs or improvements of the service. After evaluating users' feedback and using our own remarks we will implement the suggested improvements for the technique and the software.

Enhancements to the fragmentation algorithm are also included in our future work and research plans. There are some cases where small leaf nodes are selected as WCs or some areas of content are not included in any WC of a page. This is a consequence of only using the TABLE tags for defining page structure. However, this is not the only option for the algorithm. Index tree nodes can include all the children of a table tag such as TR or TD. This will increase the size of the index trees, since many more nodes will be added, but will offer greater flexibility, providing more options for the fragmentation. Some additional features should also be implemented in the fragmentation algorithm, such as the ability to combine different nodes to build the index tree. This will also reduce the problem of selecting small nodes as WCs, since even nodes that are not descendants of a common ancestor could be merged in a WC. In addition, we will investigate the possibility of implementing a second fragmentation algorithm that is not based on the table structure of HTML pages. This will allow the system to function with pages that do not contain table tags, but are created using CSS styles positioning. We believe such an algorithm will increase the usability of the proposed technique. However, computational complexity and the resources needed to support such a service will be increased.

The training and update procedures can be also enhanced. We will examine situations where the training or the update procedures fail and by inspecting the reasons of their failure we will improve the algorithms. Some simulations will also be made with 'mockup' sites, created just for testing the operation of the whole system. This is required in order to identify situations where the system fails because of changes that very rarely happen to real web sites.

The synthesis algorithm, which merges the WCs that are of interest to the user, can be improved by better handling of the javascript calls inside the WCs and the javascript code. In the current implementation, SCRIPT tags are treated just as all other tags and this can sometimes lead to problems.

Besides improving the proposed technique, which aims to provide a system for constructing personalized portals with content from external sites, future work plans include applying the 'WCs' concept for reducing redundant data transfers caused by the transfer of slightly changed web pages and implementing a software system towards this direction. This idea is explained in Bouras and Konidaris (2003). Although this application of WCs seems different from the one explained in the current paper, much of our research can be applied in a system that would reduce redundant data transfers by transferring only some parts of web pages and not the whole of them.

13. Conclusions

In this paper, we presented the concept of 'WCs' and its application in designing and implementing a software technique that can assist Web users in their browsing sessions, by presenting to them in a single web page only the parts of web sites that are of interest to them. Usage of this technique enhances their browsing experiences, since all information a user usually accesses in a single browsing session is gathered to his/her personalized page. Additionally, users save time, since they do not have to visit several sites to access the desired information and they avoid redundant data transfers, also reducing browsing cost. Reserved bandwidth also leads to a decrease of the user perceived latency. The first implemented prototype shows that the concept of 'WCs' can be implemented, without inducing heavy load on the web server. We believe that the final version of the software system will be a viable product in the market and that many users will embrace it.

References

Bickmore T, Schilit W. Digestor: device-independent access to the World Wide Web. Comput Networks ISDN Syst 1997;29(8):1075–82.

Bouras C, Konidaris A, Wcomponents: A Concept for improving personalization and reducing user perceived latency on the World Wide Web. In: Proceedings of the second international conference on internet computing (IC2001),vol.2, Las Vegas, Nv, USA, June 25–8th 2001. pp. 38–44.

Bouras C, Konidaris A. Performance Evaluation of a Hybrid Run-time Management Policy for Data Intensive Web Sites. World Wide Web J Internet Web Inform Syst 2003;6(1):23–47.

Bouras C, Konidaris A, Estimating and Eliminating Redundant Data Transfers Over the Web: A Fragment Based Approach. In: Proceedings of the third international conference on internet computing (IC2002), USA, 2002.

Bouras C, Kapoulas V, Misedakis I, A Web-page fragmentation technique for personalized browsing. In: 19th ACM Symposium on applied computing-track on inernet data management, Nicosia, Cyprus, March 14–17, 2004, pp. 1146–7.

Britton KH, et al. Transcoding: extending E-Business to new environments. IBM Syst J 2001;40(1).

Buttler D, Liu L, Pu C, A Fully Automated Object Extraction system for the World Wide Web. In: Proceedings of the 2001 international conference on distributed computing systems (IDCS '01).

Buyukkokten H, Garcia-Molina H, Paepcke A, Accordion summarization for End-Game Browsing on PDAs and Cellular Phones. In Proceedings of the conference on human factors in computing systems, CHI'01, 2001.

Caverlee J, Butler D, Liu L, Probe, Cluster, and Discover: Focused Extraction of QA-Pagelets fom the Deep Web. In: Proceedings of the 20th IEEE international conference on data engineeing (ICDE ′04), March 30–April 2, 2004, Boston.

Challenger J, Iyengar A, Witting K, Ferstat C, Reed P, A publishing system for efficiently creating dynamic web content. In: Proceedings of the IEEE conference on computer communications (INFOCOM'00), March 2000.

Chen J, Zhou B, Shi J, Zhang H, Fengwu Q. Function-Based Object Model Towards Web-site Adaptation. In: Proceedings of the 10th WWW Conference. New york: ACM Press; 2001. p. 587–96.

Chen Y, Ma W, Zhang H, Detecting Web Page Structure for Adaptive Viewing on small Form Factor Devices. In: Proceedings of WWW '03, May 20–24, 2003, Budapest, Hungary.

Freire J, Kumar B, Lieuwen D, WebViews: Accessing Personalized Web Content and Services. In: Proceedings of the 10th international conference on World Wide Web, Hong Kong, 2001.

Hori M, Kondoh G, Ono K, Hirose S, Singhal S, Annotation-based web content transcoding. In: Proceedings of the ninth international world wide web conference, Amsterdam, The Netherlands, May 2000.

Hwang E, Lee S, Web surfing assistant for improving web accessibility. In: international conference on internet computing (IC'03), Las Vegas, NV, USA, June 23–26, 2003.

Hwang Y, Jung C, Kim J, Chung S, WebAlchemist: a web transcoding system for mobile web access in handheld devices. In: Proceedings of ITCom, SPIE—The International Society for Optical Engineering, 2001. p. 37–46.

Hwang Y, Kim J, Seo E, Structure-aware web transcoding for mobile devices. IEEE Internet Comput J September–October 2003.

Kaasinen E, Aantonen M, Kolari J, Melakoski S, Laakko T, two approaches to bringing Internet services to WAP devices. In: Proceedings of the ninth International World Wide Web conference, 2000.

Liu Z, Keong Ng W, Lim E, Li F. Towards building logical views of websites. Data Knowledge Eng 2004;49(2):197–222.

Ramaswamy L, Iyengar A, Liu L, Douglis F. Automatic Fragment Detection in Dynamic Web Pages and Its Impact on Caching. IEEE Trans Knowledge Data Eng 2005;17(6):859–74.

Sugiura A, Koseki Y, Internet Scrapbook: creating personalized world wide web pages. CHI 97 Extended Abstract, 1997. pp. 343–4.

Wills CE, Mikhailov M, Studying the impact of more complete server information on Web caching. In: fifth International Web caching and Content delivery Workshop, Lisbon, Portugal, 22–24 May 2000.

Yu S, Cai D, Wen J, Ma W, Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In: Proceedings of WWW '03, May 20–24, 2003, Budapest, Hungary.